SOUND4 IMPACT.CL Library [1.2.3]

Generated by Doxygen 1.9.1

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

2 Deprecated List 3

3 Module Index 5
3.1 Modules . . . 5

4 Namespace Index 7
4.1 Namespace List . . . 7

5 Hierarchical Index 9
5.1 Class Hierarchy . . . 9

6 Class Index 11
6.1 Class List . . . 11

7 File Index 13
7.1 File List . . . 13

8 Module Documentation 15
8.1 Sample using C interface . . . 15
8.1.1 Sample to create the processing instance . . . 15
8.1.2 Sample processing loop . . . 16
8.1.3 Sample to stop the processing instance . . . 16
8.2 Environment variables and parameters . . . 17
8.2.1 Parameters defaulting from environment . . . 17
8.2.1.1 Webserver authentification setup : . . . 17
8.2.1.2 Webserver secure : . . . 17
8.2.1.3 Network access : . . . 17
8.2.2 Parameters . . . 17
8.2.3 Preset sharing . . . 18
8.3 Global setup . . . 18
8.3.1 Detailed Description . . . 18
8.3.2 Macro Definition Documentation . . . 19
8.3.2.1 IMPACT_AUDIO_INPUT_COUNT . . . 19
8.3.2.2 IMPACT_AUDIO_OUTPUT_COUNT . . . 19
8.3.2.3 IMPACT_AUDIOFRAME_COUNT . . . 19
8.3.2.4 IMPACT_CHANNEL_COUNT . . . 20
8.3.2.5 IMPACT_SAMPLE_RATE . . . 20
8.3.3 Typedef Documentation . . . 20
8.3.3.1 impact_loggerfn . . . 20
8.3.4 Enumeration Type Documentation . . . 20
8.3.4.1 impact_LogSeverity . . . 20

8.3.5 Function Documentation . . . 21
8.3.5.1 impact_GetAudioInputCount() . . . 21
8.3.5.2 impact_GetAudioOutputCount() . . . 21
8.3.5.3 impact_GetChannelCount() . . . 22
8.3.5.4 impact_GetChunkSizeInFrames() . . . 22
8.3.5.5 impact_GetPossibleChunkSizeInFrames() . . . 22
8.3.5.6 impact_GetSampleRate() . . . 23
8.3.5.7 impact_GetVersion() . . . 23
8.3.5.8 impact_SetLoggerCallback() . . . 23
8.3.5.9 impact_SetLogSeverity() . . . 24
8.4 Process creation/destruction . . . 24
8.4.1 Detailed Description . . . 24
8.4.2 Function Documentation . . . 24
8.4.2.1 impact_ExitProcess() . . . 24
8.4.2.2 impact_FreeParameters() . . . 25
8.4.2.3 impact_FreeParameterValue() . . . 25
8.4.2.4 impact_GetParameter() . . . 25
8.4.2.5 impact_GetProcessChunkFrames() . . . 25
8.4.2.6 impact_InitProcess() . . . 26
8.4.2.7 impact_InitProcess2() . . . 26
8.4.2.8 impact_InitProcess3() . . . 27
8.4.2.9 impact_NewParameters() . . . 28
8.4.2.10 impact_SetParameter() . . . 28
8.4.2.11 impact_TerminateProcess() . . . 28
8.5 Custom preset managagement . . . 29
8.5.1 Detailed Description . . . 30
8.5.2 Typedef Documentation . . . 30
8.5.2.1 fs_char . . . 30
8.5.2.2 impact_storage_exists . . . 30
8.5.2.3 impact_storage_getall . . . 31
8.5.2.4 impact_storage_getall_free . . . 31
8.5.2.5 impact_storage_reader . . . 31
8.5.2.6 impact_storage_reader_free . . . 32
8.5.2.7 impact_storage_remove . . . 32
8.5.2.8 impact_storage_rename . . . 33
8.5.2.9 impact_storage_writer . . . 33
8.5.3 Enumeration Type Documentation . . . 33
8.5.3.1 impact_PresetChange_Kind . . . 33
8.5.4 Function Documentation . . . 34
8.5.4.1 impact_PresetManager_InformChange() . . . 34
8.5.4.2 impact_SetPresetManager() . . . 34
8.6 Metadata Management . . . 35

8.6.1 Detailed Description . . . 35
8.6.2 Function Documentation . . . 35
8.6.2.1 impact_FreeMetadataInfos() . . . 35
8.6.2.2 impact_GetMetadataInfos() . . . 36
8.6.2.3 impact_SetMetadata() . . . 36
8.7 Update Management . . . 36
8.7.1 Detailed Description . . . 36
8.7.2 Function Documentation . . . 37
8.7.2.1 impact_StartUpdateThread() . . . 37
8.7.2.2 impact_StopUpdateThread() . . . 37
8.7.2.3 impact_WaitUpdateThreadReady() . . . 37
8.8 Direct processing . . . 38
8.8.1 Detailed Description . . . 38
8.8.2 Function Documentation . . . 38
8.8.2.1 impact_GetBufferIn() . . . 38
8.8.2.2 impact_GetBufferOut() . . . 39
8.8.2.3 impact_GetEstimatedDelay() . . . 39
8.8.2.4 impact_ProcessAudio() . . . 40
8.8.2.5 impact_ProcessAudio_Planar() . . . 40
8.9 Audio managing (any audio) . . . 41
8.9.1 Detailed Description . . . 41
8.9.2 Enumeration Type Documentation . . . 41
8.9.2.1 impact_SampleFormat . . . 41
8.9.3 Function Documentation . . . 42
8.9.3.1 impact_AddAudio() . . . 42
8.9.3.2 impact_AddPadAudio() . . . 43
8.9.3.3 impact_GetAudio() . . . 43
8.9.3.4 impact_GetBytesFromFormat() . . . 43
8.9.3.5 impact_GetFormatFromName() . . . 44
8.9.3.6 impact_GetFormatName() . . . 44
8.9.3.7 impact_GetMaxPacketFrame() . . . 44
8.9.3.8 impact_GetOutputCount() . . . 46
8.10 Audio convert helpers . . . 46
8.10.1 Detailed Description . . . 46
8.10.2 Function Documentation . . . 46
8.10.2.1 impact_AudioConvertFrom() . . . 47
8.10.2.2 impact_AudioConvertTo() . . . 47
8.10.2.3 impact_AudioMonoFromLiveStereo() . . . 47
8.10.2.4 impact_AudioMonoToLiveStereo() . . . 48
8.10.2.5 impact_MonoToStereo() . . . 48
8.10.2.6 impact_MonoToStereo_Planar() . . . 49
8.10.2.7 impact_StereoToMono() . . . 49

8.10.2.8 impact_StereoToMono_Planar() . . . 49
8.11 Client's JSon config interfacing . . . 50
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() . . . 51
8.11.2.3 impact_NewClient() . . . 51
8.11.2.4 impact_ProcessJson() . . . 51
8.11.2.5 impact_SaveState() . . . 52
8.12 JSON syntax . . . 52
8.12.1 Commands . . . 53
8.12.2 Answers . . . 53
8.12.3 Examples . . . 54
8.12.3.1 Get status . . . 54
8.12.3.2 Get all changes since last request . . . 54
8.12.3.3 Check license . . . 54
8.12.3.4 Set Processor name . . . 54
8.12.3.5 Get presetlist . . . 54
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 . . . 55
8.12.3.9 Set bypass . . . 55
8.12.3.10 Export a preset . . . 55
8.12.3.11 Import a preset . . . 55
8.13 JSON syntax for internal setup . . . 55
8.13.1 Get process and modules information . . . 56
8.13.2 Authentication management . . . 56
8.13.2.1 Difference between setup and admin account . . . 56
8.13.2.2 Access encoding . . . 56
8.13.2.3 Change setup account . . . 56
8.13.2.4 Get user info . . . 57
8.13.2.5 Add or modify a user . . . 57
8.13.2.6 Removes a user . . . 57
8.13.2.7 List users . . . 57
8.13.2.8 Get/set unlogged access . . . 57
8.13.2.9 Get realm . . . 57
8.13.2.10 Get API version . . . 58
8.13.2.11 Get setup is admin . . . 58
8.13.2.12 Get/set setup account . . . 58
8.13.2.13 Get/set custom storage . . . 58
8.13.3 Get process' interface description . . . 58
8.13.3.1 Get XML decription . . . 58

8.13.3.2 Get Process Image . . . 58
8.13.3.3 Get Process Ico . . . 59
8.14 Web server . . . 59
8.14.1 Detailed Description . . . 59
8.14.2 Macro Definition Documentation . . . 59
8.14.2.1 SOUND4_INVALID_WEBSERVER_ID . . . 59
8.14.2.2 SOUND4_WEBSERVER_HTTP_OK . . . 60
8.14.2.3 SOUND4_WEBSERVER_HTTPS_OK . . . 60
8.14.3 Function Documentation . . . 60
8.14.3.1 impact_Webserver() . . . 60
8.14.3.2 impact_Webserver_Status() . . . 61
8.14.3.3 impact_Webserver_Stop() . . . 61
8.14.3.4 impact_Webserver_tcp() . . . 62
8.14.3.5 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

9 Namespace Documentation **67**
9.1 dyn Namespace Reference . . . 67
9.1.1 Detailed Description . . . 67
9.2 sound4 Namespace Reference . . . 67
9.2.1 Detailed Description . . . 67
9.3 sound4::impact Namespace Reference . . . 67
9.3.1 Detailed Description . . . 68
9.3.2 Typedef Documentation . . . 68
9.3.2.1 log_cb_t . . . 68
9.3.2.2 LogSeverity . . . 68
9.3.2.3 PresetChange_Kind . . . 69
9.3.2.4 SampleFormat . . . 69

9.3.3 Variable Documentation . . . 69
9.3.3.1 ChunkFrames . . . 69
9.3.3.2 InputChannels . . . 69
9.3.3.3 InputSampleSize . . . 69
9.3.3.4 OutputChannels . . . 70
9.3.3.5 OutputSampleSize . . . 70
9.3.3.6 SampleRate . . . 70
9.4 sound4::impact::dyn Namespace Reference . . . 70
9.4.1 Typedef Documentation . . . 71
9.4.1.1 log_cb_t . . . 71
9.4.1.2 LogSeverity . . . 71
9.4.1.3 PresetChange_Kind . . . 71
9.4.1.4 SampleFormat . . . 71
9.4.2 Variable Documentation . . . 72
9.4.2.1 ChunkFrames . . . 72
9.4.2.2 InputChannels . . . 72
9.4.2.3 InputSampleSize . . . 72
9.4.2.4 OutputChannels . . . 72
9.4.2.5 OutputSampleSize . . . 72
9.4.2.6 SampleRate . . . 73
9.5 sound4::impact::dyn::helper Namespace Reference . . . 73
9.5.1 Detailed Description . . . 73
9.5.2 Typedef Documentation . . . 73
9.5.2.1 DynFunc_t . . . 73
9.5.2.2 DynLib_t . . . 74
9.6 sound4::impact::helper Namespace Reference . . . 74
9.6.1 Detailed Description . . . 74
9.7 std Namespace Reference . . . 74

10 Class Documentation **75**
10.1 sound4::impact::CInstance::CClient Class Reference . . . 75
10.1.1 Detailed Description . . . 75
10.1.2 Constructor & Destructor Documentation . . . 75
10.1.2.1 CClient() [1/2] . . . 75
10.1.2.2 CClient() [2/2] . . . 76
10.1.2.3 ~CClient() . . . 76
10.1.3 Member Function Documentation . . . 76
10.1.3.1 operator=() . . . 76
10.1.3.2 ProcessJson() . . . 76
10.2 sound4::impact::dyn::CInstance::CClient Class Reference . . . 77
10.2.1 Detailed Description . . . 77
10.2.2 Constructor & Destructor Documentation . . . 77

10.2.2.1 CClient() [1/2] . . . 77
10.2.2.2 CClient() [2/2] . . . 78
10.2.2.3 ~CClient() . . . 78
10.2.3 Member Function Documentation . . . 78
10.2.3.1 operator=() . . . 78
10.2.3.2 ProcessJson() . . . 78
10.3 sound4::impact::dyn::CDynLib Class Reference . . . 79
10.3.1 Detailed Description . . . 80
10.3.2 Constructor & Destructor Documentation . . . 80
10.3.2.1 CDynLib() . . . 80
10.3.3 Member Function Documentation . . . 80
10.3.3.1 IsOk() . . . 81
10.3.3.2 Load() . . . 81
10.3.4 Member Data Documentation . . . 81
10.3.4.1 AddAudio . . . 81
10.3.4.2 AddPadAudio . . . 81
10.3.4.3 AudioConvertFrom . . . 82
10.3.4.4 AudioConvertTo . . . 82
10.3.4.5 AudioMonoFromLiveStereo . . . 82
10.3.4.6 AudioMonoToLiveStereo . . . 82
10.3.4.7 DeleteClient . . . 82
10.3.4.8 ExitProcess . . . 82
10.3.4.9 FreeJsonAnswer . . . 83
10.3.4.10 FreeMetadataInfos . . . 83
10.3.4.11 FreeParameters . . . 83
10.3.4.12 FreeParameterValue . . . 83
10.3.4.13 GetAudio . . . 83
10.3.4.14 GetAudioInputCount . . . 83
10.3.4.15 GetAudioOutputCount . . . 84
10.3.4.16 GetBufferIn . . . 84
10.3.4.17 GetBufferOut . . . 84
10.3.4.18 GetBytesFromFormat . . . 84
10.3.4.19 GetChannelCount . . . 84
10.3.4.20 GetChunkSizeInFrames . . . 84
10.3.4.21 GetEstimatedDelay . . . 85
10.3.4.22 GetFormatFromName . . . 85
10.3.4.23 GetFormatName . . . 85
10.3.4.24 GetMaxPacketFrame . . . 85
10.3.4.25 GetMetadataInfos . . . 85
10.3.4.26 GetOutputCount . . . 85
10.3.4.27 GetParameter . . . 86
10.3.4.28 GetPossibleChunkSizeInFrames . . . 86

10.3.4.29 GetProcessChunkFrames . . . 86
10.3.4.30 GetSampleRate . . . 86
10.3.4.31 GetVersion . . . 86
10.3.4.32 InitProcess . . . 86
10.3.4.33 InitProcess2 . . . 87
10.3.4.34 InitProcess3 . . . 87
10.3.4.35 MonoToStereo . . . 87
10.3.4.36 MonoToStereo_Planar . . . 87
10.3.4.37 NewClient . . . 87
10.3.4.38 NewParameters . . . 87
10.3.4.39 PresetManager_InformChange . . . 88
10.3.4.40 ProcessAudio . . . 88
10.3.4.41 ProcessAudio_Planar . . . 88
10.3.4.42 ProcessJson . . . 88
10.3.4.43 SaveState . . . 88
10.3.4.44 SetLoggerCallback . . . 88
10.3.4.45 SetLogSeverity . . . 89
10.3.4.46 SetMetadata . . . 89
10.3.4.47 SetParameter . . . 89
10.3.4.48 SetPresetManager . . . 89
10.3.4.49 StartUpdateThread . . . 89
10.3.4.50 StereoToMono . . . 89
10.3.4.51 StereoToMono_Planar . . . 90
10.3.4.52 StopUpdateThread . . . 90
10.3.4.53 TerminateProcess . . . 90
10.3.4.54 WaitUpdateThreadReady . . . 90
10.3.4.55 Webserver . . . 90
10.3.4.56 Webserver_Status . . . 90
10.3.4.57 Webserver_Stop . . . 91
10.3.4.58 Webserver_tcp . . . 91
10.3.4.59 Webserver_tcp2 . . . 91
10.4 sound4::impact::dyn::helper::CDynLoader Class Reference . . . 91
10.4.1 Detailed Description . . . 92
10.4.2 Constructor & Destructor Documentation . . . 92
10.4.2.1 CDynLoader() [1/3] . . . 92
10.4.2.2 CDynLoader() [2/3] . . . 92
10.4.2.3 CDynLoader() [3/3] . . . 92
10.4.2.4 ~CDynLoader() . . . 92
10.4.3 Member Function Documentation . . . 92
10.4.3.1 Close() . . . 93
10.4.3.2 GetSymbol() . . . 93
10.4.3.3 GetSymbol_safe() . . . 93

10.4.3.4 GetThisLibraryPath() . . . 93
10.4.3.5 IsOk() . . . 93
10.4.3.6 Load() . . . 94
10.4.3.7 operator bool() . . . 94
10.4.3.8 operator=() . . . 94
10.5 sound4::impact::CInstance Class Reference . . . 94
10.5.1 Detailed Description . . . 95
10.5.2 Constructor & Destructor Documentation . . . 95
10.5.2.1 CInstance() [1/2] . . . 96
10.5.2.2 CInstance() [2/2] . . . 96
10.5.2.3 ~CInstance() . . . 96
10.5.3 Member Function Documentation . . . 96
10.5.3.1 AddAudio() . . . 96
10.5.3.2 Create() . . . 96
10.5.3.3 GetAudio() . . . 97
10.5.3.4 GetBufferIn() . . . 97
10.5.3.5 GetBufferOut() . . . 98
10.5.3.6 GetChunkFrames() . . . 98
10.5.3.7 GetEstimatedDelay() . . . 98
10.5.3.8 GetMetadataInfos() . . . 99
10.5.3.9 GetParam() . . . 99
10.5.3.10 IsOk() . . . 99
10.5.3.11 NewClient() . . . 99
10.5.3.12 OnUpdateThreadStart() . . . 100
10.5.3.13 OnUpdateThreadStop() . . . 100
10.5.3.14 operator=() . . . 100
10.5.3.15 PresetManager_InformChange() . . . 100
10.5.3.16 ProcessAnyAudio() . . . 100
10.5.3.17 ProcessAudio() . . . 101
10.5.3.18 ProcessAudio_Planar() . . . 101
10.5.3.19 SetMetadata() . . . 102
10.5.3.20 SetParam() . . . 102
10.5.3.21 SetPresetManager() . . . 103
10.5.3.22 StartWebServer() . . . 103
10.5.3.23 Stop() . . . 103
10.5.3.24 StopWebServer() . . . 104
10.5.3.25 Terminate() . . . 104
10.6 sound4::impact::dyn::CInstance Class Reference . . . 104
10.6.1 Detailed Description . . . 105
10.6.2 Constructor & Destructor Documentation . . . 105
10.6.2.1 CInstance() [1/2] . . . 106
10.6.2.2 CInstance() [2/2] . . . 106

10.6.2.3 ~CInstance() . . . 106
10.6.3 Member Function Documentation . . . 106
10.6.3.1 AddAudio() . . . 106
10.6.3.2 Create() . . . 106
10.6.3.3 GetAudio() . . . 107
10.6.3.4 GetBufferIn() . . . 107
10.6.3.5 GetBufferOut() . . . 108
10.6.3.6 GetChunkFrames() . . . 108
10.6.3.7 GetEstimatedDelay() . . . 108
10.6.3.8 GetMetadataInfos() . . . 109
10.6.3.9 GetParam() . . . 109
10.6.3.10 IsOk() . . . 109
10.6.3.11 NewClient() . . . 109
10.6.3.12 OnUpdateThreadStart() . . . 110
10.6.3.13 OnUpdateThreadStop() . . . 110
10.6.3.14 operator=() . . . 110
10.6.3.15 PresetManager_InformChange() . . . 110
10.6.3.16 ProcessAnyAudio() . . . 110
10.6.3.17 ProcessAudio() . . . 111
10.6.3.18 ProcessAudio_Planar() . . . 111
10.6.3.19 SetMetadata() . . . 112
10.6.3.20 SetParam() . . . 112
10.6.3.21 SetPresetManager() . . . 113
10.6.3.22 StartWebServer() . . . 113
10.6.3.23 Stop() . . . 113
10.6.3.24 StopWebServer() . . . 114
10.6.3.25 Terminate() . . . 114
10.7 sound4::impact::CPresetLoader Class Reference . . . 114
10.7.1 Detailed Description . . . 115
10.7.2 Constructor & Destructor Documentation . . . 115
10.7.2.1 CPresetLoader() . . . 115
10.7.2.2 ~CPresetLoader() . . . 115
10.7.3 Member Function Documentation . . . 115
10.7.3.1 Exists() . . . 115
10.7.3.2 GetAll() . . . 115
10.7.3.3 IsReadOnly() . . . 115
10.7.3.4 Read() . . . 116
10.7.3.5 Remove() . . . 116
10.7.3.6 Rename() . . . 116
10.7.3.7 Write() . . . 116
10.8 sound4::impact::dyn::CPresetLoader Class Reference . . . 116
10.8.1 Detailed Description . . . 117

10.8.2 Constructor & Destructor Documentation . . . 117
10.8.2.1 CPresetLoader() . . . 117
10.8.2.2 ∼CPresetLoader() . . . 117
10.8.3 Member Function Documentation . . . 117
10.8.3.1 Exists() . . . 117
10.8.3.2 GetAll() . . . 117
10.8.3.3 IsReadOnly() . . . 118
10.8.3.4 Read() . . . 118
10.8.3.5 Remove() . . . 118
10.8.3.6 Rename() . . . 118
10.8.3.7 Write() . . . 118
10.9 sound4::impact::dyn::helper::DynFuncHolder< T > Class Template Reference . . . 119
10.9.1 Detailed Description . . . 119
10.9.2 Constructor & Destructor Documentation . . . 119
10.9.2.1 DynFuncHolder() [1/2] . . . 119
10.9.2.2 DynFuncHolder() [2/2] . . . 119
10.9.3 Member Function Documentation . . . 120
10.9.3.1 IsOk() . . . 120
10.9.3.2 operator bool() . . . 120
10.9.3.3 operator T∗() . . . 120
10.10 impact_CClientInstance Struct Reference . . . 120
10.10.1 Detailed Description . . . 120
10.11 impact_CInstance Struct Reference . . . 121
10.11.1 Detailed Description . . . 121
10.12 impact_CParameters Struct Reference . . . 121
10.12.1 Detailed Description . . . 121
10.13 MyPresetManager Class Reference . . . 121
10.13.1 Detailed Description . . . 122
10.13.2 Constructor & Destructor Documentation . . . 122
10.13.2.1 MyPresetManager() . . . 122
10.13.2.2 ∼MyPresetManager() . . . 122
10.13.3 Member Function Documentation . . . 122
10.13.3.1 Exists() . . . 122
10.13.3.2 GetAll() . . . 122
10.13.3.3 IsReadOnly() . . . 123
10.13.3.4 Read() . . . 123
10.13.3.5 Remove() . . . 123
10.13.3.6 Rename() . . . 123
10.13.3.7 SetupPath() . . . 123
10.13.3.8 Write() . . . 124
10.14 sound4::impact::dyn::helper::SampleFormat< T > Struct Template Reference . . . 124
10.14.1 Detailed Description . . . 124

10.15 sound4::impact::helper::SampleFormat< T > Struct Template Reference 124
10.15.1 Detailed Description 124
10.16 sound4::impact::dyn::helper::SampleFormat< float > Struct Reference 125
10.16.1 Detailed Description 125
10.16.2 Member Data Documentation 125
10.16.2.1 format 125
10.17 sound4::impact::helper::SampleFormat< float > Struct Reference 125
10.17.1 Detailed Description 126
10.17.2 Member Data Documentation 126
10.17.2.1 format 126
10.18 sound4::impact::dyn::helper::SampleFormat< int16_t > Struct Reference 126
10.18.1 Detailed Description 126
10.18.2 Member Data Documentation 126
10.18.2.1 format 127
10.19 sound4::impact::helper::SampleFormat< int16_t > Struct Reference 127
10.19.1 Detailed Description 127
10.19.2 Member Data Documentation 127
10.19.2.1 format 127
10.20 sound4::impact::dyn::helper::SampleFormat< int32_t > Struct Reference 127
10.20.1 Detailed Description 128
10.20.2 Member Data Documentation 128
10.20.2.1 format 128
10.21 sound4::impact::helper::SampleFormat< int32_t > Struct Reference 128
10.21.1 Detailed Description 128
10.21.2 Member Data Documentation 128
10.21.2.1 format 128

11 File Documentation **129**
11.1 sound4.impact.cl-pipe-cxx.cpp File Reference 129
11.1.1 Function Documentation 129
11.1.1.1 GetEnv() 129
11.1.1.2 main() 130
11.1.1.3 MyLogger() 130
11.1.1.4 ProcessLoop() 130
11.1.1.5 Usage() 130
11.2 sound4.impact.cl-pipe-cxx.cpp 131
11.3 sound4.impact.cl-pipe-dyn.cpp File Reference 132
11.3.1 Function Documentation 133
11.3.1.1 GetEnv() 133
11.3.1.2 main() 134
11.3.1.3 MyLogger() 134
11.3.1.4 ProcessLoop() 134

11.3.1.5 Usage() . . . 135
11.4 sound4.impact.cl-pipe-dyn.cpp . . . 135
11.5 sound4.impact.cl-pipe.cpp File Reference . . . 138
11.5.1 Macro Definition Documentation . . . 138
11.5.1.1 CHANNEL_COUNT . . . 139
11.5.1.2 CHUNK_SIZE . . . 139
11.5.1.3 SAMPLE_RATE . . . 139
11.5.2 Function Documentation . . . 139
11.5.2.1 main() . . . 139
11.5.2.2 MyLogger() . . . 140
11.5.2.3 ProcessLoop() . . . 140
11.5.2.4 Usage() . . . 140
11.6 sound4.impact.cl-pipe.cpp . . . 140
11.7 sound4.impact.cl.h File Reference . . . 143
11.7.1 Macro Definition Documentation . . . 146
11.7.1.1 IMPACT_DECL . . . 146
11.7.1.2 IMPACT_HAS_CLOUDBUS . . . 146
11.7.1.3 IMPACT_HAS_WEBSERVER . . . 146
11.8 sound4.impact.cl.h . . . 146
11.9 sound4.impact.cl.hpp File Reference . . . 150
11.9.1 Macro Definition Documentation . . . 151
11.9.1.1 fs_strdup . . . 151
11.10 sound4.impact.cl.hpp . . . 152
11.11 sound4.impact.cl_dyn.hpp File Reference . . . 159
11.11.1 Detailed Description . . . 160
11.11.2 Macro Definition Documentation . . . 160
11.11.2.1 fs_strdup . . . 160
11.12 sound4.impact.cl_dyn.hpp . . . 160

Index **173**

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 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 ∗AWS_Access_ID, const char ∗AWS_Access_Secret, const char ∗save_path)

Use impact_InitProcess3

Member impact_InitProcess2 (const char ∗LoginKey, const char ∗RadioName, const char ∗AWS_Access↩ _ID, const char ∗AWS_Access_Secret, const char ∗save_path, const struct impact_CParameters ∗parameters)

Use impact_InitProcess3

Chapter 3

Module Index

3.1 Modules

Here is a list of all modules:

Sample using C interface 15
Environment variables and parameters 17
Global setup 18
Process creation/destruction 24
Custom preset managagement 29
Metadata Management 35
Update Management 36
Direct processing 38
Audio managing (any audio) 41
Audio convert helpers 46
Client's JSon config interfacing 50
JSON syntax 52
JSON syntax for internal setup 55
Web server 59
Sample using C++ interface wrapper 63
Sample using C++ interface wrapper with dynamic library loading 64

Chapter 4

Namespace Index

4.1 Namespace List

Here is a list of all namespaces with brief descriptions:

dyn 67
sound4 67
sound4::impact 67
sound4::impact::dyn 70
sound4::impact::dyn::helper 73
sound4::impact::helper 74
std 74

Chapter 5

Hierarchical Index

5.1 Class Hierarchy

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

sound4::impact::CInstance::CClient . . . 75
sound4::impact::dyn::CInstance::CClient . . . 77
sound4::impact::dyn::CDynLib . . . 79
sound4::impact::dyn::helper::CDynLoader . . . 91
sound4::impact::CInstance . . . 94
sound4::impact::dyn::CInstance . . . 104
sound4::impact::CPresetLoader . . . 114
sound4::impact::dyn::CPresetLoader . . . 116
MyPresetManager . . . 121
sound4::impact::dyn::helper::DynFuncHolder< T > . . . 119
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_AddAudio) > . . . 119
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_AddPadAudio) > . . . 119
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_AudioConvertFrom) > . . . 119
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_AudioConvertTo) > . . . 119
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_AudioMonoFromLiveStereo) > . . . 119
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_AudioMonoToLiveStereo) > . . . 119
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_DeleteClient) > . . . 119
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_ExitProcess) > . . . 119
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_FreeJsonAnswer) > . . . 119
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_FreeMetadataInfos) > . . . 119
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_FreeParameters) > . . . 119
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_FreeParameterValue) > . . . 119
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetAudio) > . . . 119
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetAudioInputCount) > . . . 119
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetAudioOutputCount) > . . . 119
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetBufferIn) > . . . 119
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetBufferOut) > . . . 119
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetBytesFromFormat) > . . . 119
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetChannelCount) > . . . 119
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetChunkSizeInFrames) > . . . 119
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetEstimatedDelay) > . . . 119
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetFormatFromName) > . . . 119
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetFormatName) > . . . 119
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetMaxPacketFrame) > . . . 119
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetMetadataInfos) > . . . 119

sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetOutputCount) > 119
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetParameter) > 119
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetPossibleChunkSizeInFrames) > . . 119
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetProcessChunkFrames) > 119
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetSampleRate) > 119
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_GetVersion) > 119
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_InitProcess) > 119
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_InitProcess2) > 119
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_InitProcess3) > 119
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_MonoToStereo) > 119
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_MonoToStereo_Planar) > 119
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_NewClient) > 119
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_NewParameters) > 119
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_PresetManager_InformChange) > . . . 119
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_ProcessAudio) > 119
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_ProcessAudio_Planar) > 119
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_ProcessJson) > 119
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_SaveState) > 119
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_SetLoggerCallback) > 119
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_SetLogSeverity) > 119
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_SetMetadata) > 119
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_SetParameter) > 119
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_SetPresetManager) > 119
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_StartUpdateThread) > 119
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_StereoToMono) > 119
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_StereoToMono_Planar) > 119
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_StopUpdateThread) > 119
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_TerminateProcess) > 119
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_WaitUpdateThreadReady) > 119
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_Webserver) > 119
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_Webserver_Status) > 119
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_Webserver_Stop) > 119
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_Webserver_tcp) > 119
sound4::impact::dyn::helper::DynFuncHolder< decltype(impact_Webserver_tcp2) > 119
impact_CClientInstance 120
impact_CInstance 121
impact_CParameters 121
sound4::impact::dyn::helper::SampleFormat< T > 124
sound4::impact::helper::SampleFormat< T > 124
sound4::impact::dyn::helper::SampleFormat< float > 125
sound4::impact::helper::SampleFormat< float > 125
sound4::impact::dyn::helper::SampleFormat< int16_t > 126
sound4::impact::helper::SampleFormat< int16_t > 127
sound4::impact::dyn::helper::SampleFormat< int32_t > 127
sound4::impact::helper::SampleFormat< int32_t > 128

Chapter 6

Class Index

6.1 Class List

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

sound4::impact::CInstance::CClient 75
sound4::impact::dyn::CInstance::CClient 77
sound4::impact::dyn::CDynLib
Dynamic library interface 79
sound4::impact::dyn::helper::CDynLoader
Helper to load dynamic library 91
sound4::impact::CInstance 94
sound4::impact::dyn::CInstance 104
sound4::impact::CPresetLoader 114
sound4::impact::dyn::CPresetLoader 116
sound4::impact::dyn::helper::DynFuncHolder< T >
Helper to wrap a function pointer 119
impact_CClientInstance 120
impact_CInstance 121
impact_CParameters 121
MyPresetManager
[Custom preset manager] 121
sound4::impact::dyn::helper::SampleFormat< T >
Helper for Sample format types 124
sound4::impact::helper::SampleFormat< T >
Helper for Sample format types 124
sound4::impact::dyn::helper::SampleFormat< float > 125
sound4::impact::helper::SampleFormat< float > 125
sound4::impact::dyn::helper::SampleFormat< int16_t > 126
sound4::impact::helper::SampleFormat< int16_t > 127
sound4::impact::dyn::helper::SampleFormat< int32_t > 127
sound4::impact::helper::SampleFormat< int32_t > 128

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 129
sound4.impact.cl-pipe-dyn.cpp
C++ sample using C++ interface wrapper with dynamic library loading 132
sound4.impact.cl-pipe.cpp
C++ sample using only C interface 138
sound4.impact.cl.h
C interface for library 143
sound4.impact.cl.hpp
C++ wrapper for library 150
sound4.impact.cl_dyn.hpp
C++ wrapper for dynamic loading library 159

Chapter 8

Module Documentation

8.1 Sample using C interface

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

8.1.1 Sample to create the processing instance

see Process creation/destruction and Update Management

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

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

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

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

8.1.2 Sample processing loop

see Direct processing

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

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

8.1.3 Sample to stop the processing instance

see Process creation/destruction

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

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

8.2 Environment variables and parameters

8.2.1 Parameters defaulting from environment

The following environment variables are used by software library:

Note

Those variables can be passed as parameters with impact_InitProcess2

8.2.1.1 Webserver authentification setup :

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

8.2.1.2 Webserver secure :

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

8.2.1.3 Network access :

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

8.2.2 Parameters

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

Run mode:

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

8.2.3 Preset sharing

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

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

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

8.3 Global setup

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 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 166 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 172 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 154 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 160 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 178 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 271 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 252 of file sound4.impact.cl.h.

8.3.5 Function Documentation

8.3.5.1 impact_GetAudioInputCount()

```
unsigned int impact_GetAudioInputCount ( )
```

Get the number of inputs for this process

Remarks

All inputs have impact_GetChannelCount() channels

Fixed to 1, but should be checked for backward compatibility

Returns

The process input count

8.3.5.2 impact_GetAudioOutputCount()

```
unsigned int impact_GetAudioOutputCount ( )
```

Get the number of outputs for this process

Remarks

All outputs have impact_GetChannelCount() channels

Fixed to 1, but should be checked for backward compatibility

Returns

The process outputs count

8.3.5.3 impact_GetChannelCount()

```
unsigned int impact_GetChannelCount ( )
```

Get the number of channels needed by this process

Remarks

Fixed to 2, but should be checked for backward compatibility

Returns

The process channels count

8.3.5.4 impact_GetChunkSizeInFrames()

```
unsigned int impact_GetChunkSizeInFrames ( )
```

Get the number of frames in a process chunk.

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

Note

A Frame is a set of impact_GetChannelCount samples

Remarks

Fixed to 64, but should be checked for backward compatibility

Returns

The process chunk size in frame

Deprecated Use impact_GetPossibleChunkSizeInFrames

8.3.5.5 impact_GetPossibleChunkSizeInFrames()

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

Get the possibles number of frames in a process chunk.

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

Note

A Frame is a set of impact_GetChannelCount samples

Returns

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

8.3.5.6 impact_GetSampleRate()

```
unsigned int impact_GetSampleRate ( )
```

Get the processing sample-rate

Remarks

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

Returns

The process sample-rate in Hz

8.3.5.7 impact_GetVersion()

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

Get the process version string

Returns

version string

8.3.5.8 impact_SetLoggerCallback()

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

Set a callback for logging

If not set, a default logger prints to stderr.

Warning

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

Parameters

logger	A C callback called for logging.

8.3.5.9 impact_SetLogSeverity()

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

Change the global log severity

Logs above this level will not be printed.

Parameters

severity	Log severity

8.4 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 ∗AWS_Access_ID, const char ∗AWS_Access_Secret, const char ∗save_path)
- struct impact_CInstance ∗ impact_InitProcess2 (const char ∗LoginKey, const char ∗RadioName, const char ∗AWS_Access_ID, const char ∗AWS_Access_Secret, const char ∗save_path, const struct impact_CParameters ∗parameters)
- struct impact_CInstance ∗ impact_InitProcess3 (const char ∗LoginKey, const char ∗RadioName, const char ∗AWS_Access_ID, const char ∗AWS_Access_Secret, const char ∗save_path, const struct impact_CParameters ∗parameters, unsigned int frames_per_chunk)
- unsigned int impact_GetProcessChunkFrames (struct impact_CInstance ∗instance)
- int impact_TerminateProcess (struct impact_CInstance ∗instance)
- void impact_ExitProcess (struct impact_CInstance ∗instance)

8.4.1 Detailed Description

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

8.4.2 Function Documentation

8.4.2.1 impact_ExitProcess()

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

Quit and destroy the instance

Parameters

instance	the instance to destroy

8.4.2.2 impact_FreeParameters()

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

Free the parameters created with impact_NewParameters.

8.4.2.3 impact_FreeParameterValue()

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

Free a parameter value returned by impact_GetParameter

Parameters

value	the returned value

8.4.2.4 impact_GetParameter()

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

Get current parameter value.

Returned value must be freed with impact_FreeParameterValue

8.4.2.5 impact_GetProcessChunkFrames()

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

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

The value reflects the frames_per_chunk parameter passed to impact_InitProcess3

Parameters

instance	the instance to destroy

Returns

the number of audio frames in each chunk of processing

Note

A Frame is a set of impact_GetChannelCount samples

8.4.2.6 impact_InitProcess()

```
struct impact_CInstance* impact_InitProcess (
            const char * LoginKey,
            const char * RadioName,
            const char * AWS_Access_ID,
            const char * AWS_Access_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 * AWS_Access_ID,
            const char * AWS_Access_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 * AWS_Access_ID,
            const char * AWS_Access_Secret,
            const char * save_path,
            const struct impact_CParameters * parameters,
            unsigned int frames_per_chunk )
```

Create a process instance

The output will have to be freed with impact_ExitProcess

Attention

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

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

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

Web login initialization

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

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

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

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

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

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

License access

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

Parameters

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

Returns

A new process instance

8.4.2.9 impact_NewParameters()

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

Create parameters to use with impact_InitProcess2.

The output will have to be freed with impact_FreeParameters

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

8.4.2.10 impact_SetParameter()

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

Set a new parameter.

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

See also Custom preset managagement

8.4.2.11 impact_TerminateProcess()

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

Terminate the processing

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

Parameters

instance	the instance to terminate

Returns

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

8.5 Custom preset managagement

This allows to provide a customized preset management.

Typedefs

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

Enumerations

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

Functions

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

8.5.1 Detailed Description

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

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

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

Known files:

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

Attention

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

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

8.5.2 Typedef Documentation

8.5.2.1 fs_char

```
typedef wchar_t fs_char
```

Definition at line 539 of file sound4.impact.cl.h.

8.5.2.2 impact_storage_exists

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

Function to call to test if a stored file exists.

Parameters

filename	the name of the file to test.
userdata	the userdata passed to impact_SetPresetManager

Returns

-1 if not exists, 0 if it exists

Definition at line 578 of file sound4.impact.cl.h.

8.5.2.3 impact_storage_getall

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

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

Parameters

userdata	the userdata passed to impact_SetPresetManager

Returns

A NULL terminated array of file names

Definition at line 587 of file sound4.impact.cl.h.

8.5.2.4 impact_storage_getall_free

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

Function to call to free the result of getall.

Parameters

all	the returned value from impact_storage_getall
userdata	the userdata passed to impact_SetPresetManager

Definition at line 595 of file sound4.impact.cl.h.

8.5.2.5 impact_storage_reader

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

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

Parameters

filename	the name of the file to read.
userdata	the userdata passed to impact_SetPresetManager

Returns

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

Definition at line 551 of file sound4.impact.cl.h.

8.5.2.6 impact_storage_reader_free

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

Function to call to free the result of impact_storage_reader.

Parameters

content	the returned value from impact_storage_reader
userdata	the userdata passed to impact_SetPresetManager

Definition at line 559 of file sound4.impact.cl.h.

8.5.2.7 impact_storage_remove

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

Function to call to remove a stored file.

Parameters

filename	the name of the file to remove.
userdata	the userdata passed to impact_SetPresetManager

Returns

-1 if failed, 0 if OK

Definition at line 604 of file sound4.impact.cl.h.

8.5.2.8 impact_storage_rename

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

Function to call to rename a stored file.

Parameters

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

Returns

-1 if failed, 0 if OK

Definition at line 614 of file sound4.impact.cl.h.

8.5.2.9 impact_storage_writer

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

Function to call to write a stored file.

Parameters

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

Returns

-1 if failed, 0 if OK

Definition at line 569 of file sound4.impact.cl.h.

8.5.3 Enumeration Type Documentation

8.5.3.1 impact_PresetChange_Kind

```
enum impact_PresetChange_Kind
```

Kind of change on a storage path

Enumerator

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

Definition at line 647 of file sound4.impact.cl.h.

8.5.4 Function Documentation

8.5.4.1 impact_PresetManager_InformChange()

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

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

Parameters

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

8.5.4.2 impact_SetPresetManager()

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

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

To dynamically update when storage change, see impact_PresetManager_InformChange

Parameters

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

8.6 Metadata Management

Functions

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

8.6.1 Detailed Description

Metadata handling.

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

8.6.2 Function Documentation

8.6.2.1 impact_FreeMetadataInfos()

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

Frees the information returned by impact_GetMetadataInfos.

Parameters

instance	the instance
infos	the infos

8.6.2.2 impact_GetMetadataInfos()

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

Get information on known metadata keys.

Parameters

instance	the instance

Returns

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

8.6.2.3 impact_SetMetadata()

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

Adds the specific metadata to the process.

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

Parameters

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

8.7 Update Management

Functions

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

8.7.1 Detailed Description

Manage the process update.

8.7.2 Function Documentation

8.7.2.1 impact_StartUpdateThread()

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

Update function

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

Remarks

This function does not return until it is stopped with impact_StopUpdateThread

Parameters

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

8.7.2.2 impact_StopUpdateThread()

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

Stop the update thread

Parameters

instance	the processing instance

8.7.2.3 impact_WaitUpdateThreadReady()

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

Wait for the update thread to be ready

Parameters

instance	the processing instance
milliseconds	maximum time to wait in milliseconds

Returns

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

8.8 Direct processing

Functions

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

8.8.1 Detailed Description

Optimized processing with direct in->out.

8.8.2 Function Documentation

8.8.2.1 impact_GetBufferIn()

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

Get preallocated aligned buffer for audio input for impact_ProcessAudio()

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

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

Note

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

Parameters

instance	the processing instance

Returns

An audio input dedicated buffer.

8.8.2.2 impact_GetBufferOut()

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

Get preallocated aligned buffer for audio output for impact_ProcessAudio()

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

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

Note

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

Parameters

instance	the processing instance

Returns

An audio output dedicated buffer.

8.8.2.3 impact_GetEstimatedDelay()

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

Get the delay (estimated) of current processing algorithm

Remarks

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

Returns

Estimated delay in frames

8.8.2.4 impact_ProcessAudio()

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

Process an audio chunk

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

Input has impact_GetAudioInputCount() ∗ impact_GetChannelCount() channels

Ouput has impact_GetAudioOutputCount() ∗ impact_GetChannelCount() channels

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

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

Warning

Buffers must be 64-byte aligned

Note

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

Parameters

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

8.8.2.5 impact_ProcessAudio_Planar()

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

Process an audio chunk of planar audio

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

Input has impact_GetAudioInputCount() ∗ impact_GetChannelCount() channels

Ouput has impact_GetAudioOutputCount() ∗ impact_GetChannelCount() channels

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

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

Parameters

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

8.9 Audio managing (any audio)

Enumerations

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

Functions

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

8.9.1 Detailed Description

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

Remarks

It is more optimized to use Direct processing.

8.9.2 Enumeration Type Documentation

8.9.2.1 impact_SampleFormat

`enum impact_SampleFormat`

Supported formats for audio I/O

Enumerator

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

Definition at line 855 of file sound4.impact.cl.h.

8.9.3 Function Documentation

8.9.3.1 impact_AddAudio()

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

Convert and process the payload.

Parameters

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

Returns

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

8.9.3.2 impact_AddPadAudio()

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

Add silence to finish the current audio chunk and process it

Parameters

instance	the processing instance

Returns

number of added silence frame

8.9.3.3 impact_GetAudio()

```
unsigned int impact_GetAudio (
            struct impact_CInstance * instance,
            uint8_t * payload,
            unsigned int max_nFrame,
            enum impact_SampleFormat fmt )
```

Consume the output

Parameters

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

Returns

number of frames put in the buffer

8.9.3.4 impact_GetBytesFromFormat()

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

Get the format byte size

Parameters

fmt	the format

Returns

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

8.9.3.5 impact_GetFormatFromName()

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

Convert string to format

Parameters

name	the format string

Returns

the matching format, or INVALID_FORMAT if invalid.

8.9.3.6 impact_GetFormatName()

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

Convert format to string

Parameters

fmt	the format

Returns

the string name of the format

8.9.3.7 impact_GetMaxPacketFrame()

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

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

Parameters

instance	The process instance

Returns

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

8.9.3.8 impact_GetOutputCount()

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

Get the available output size

Parameters

instance	the processing instance

Returns

number of available frames at output

8.10 Audio convert helpers

Functions

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

8.10.1 Detailed Description

8.10.2 Function Documentation

8.10.2.1 impact_AudioConvertFrom()

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

Convert the payload to floating-point array

Warning

Output buffer must be 64-byte aligned

Parameters

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

8.10.2.2 impact_AudioConvertTo()

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

Convert floating-point array to payload

Warning

Input buffer must be 64-byte aligned

Parameters

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

8.10.2.3 impact_AudioMonoFromLiveStereo()

```
void impact_AudioMonoFromLiveStereo (
```

```
            const uint8_t * payload,
            float * output )
```

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

Parameters

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

8.10.2.4 impact_AudioMonoToLiveStereo()

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

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

Parameters

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

8.10.2.5 impact_MonoToStereo()

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

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

Warning

Buffers must be 64-byte aligned

Parameters

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

8.10.2.6 impact_MonoToStereo_Planar()

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

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

Warning

Buffers must be 64-byte aligned

Parameters

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

8.10.2.7 impact_StereoToMono()

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

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

Warning

Buffers must be 64-byte aligned

Parameters

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

8.10.2.8 impact_StereoToMono_Planar()

```
void impact_StereoToMono_Planar (
            const float * inputL,
```

```
const float * inputR,
float * output,
size_t nFrame )
```

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

Warning

Buffers must be 64-byte aligned

Parameters

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

8.11 Client's JSon config interfacing

Classes

- struct impact_CClientInstance

Functions

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

8.11.1 Detailed Description

Allow low-level JSON configuration. see also JSON syntax

8.11.2 Function Documentation

8.11.2.1 impact_DeleteClient()

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

Deletes a client

Parameters

client	the client instance

8.11.2.2 impact_FreeJsonAnswer()

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

Free a JSON answer returned by impact_ProcessJson

Parameters

json_str	the JSON request

8.11.2.3 impact_NewClient()

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

Creates a new client

Remarks

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

Parameters

instance	the processing instance

Returns

A new client instance

8.11.2.4 impact_ProcessJson()

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

Process a JSON request

For syntax, see JSON syntax

Note

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

Attention

the return value needs to be freed with impact_FreeJsonAnswer

Parameters

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

Returns

a JSON answer (to be freed with impact_FreeJsonAnswer)

8.11.2.5 impact_SaveState()

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

Save the current state

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

Remarks

Works only when update thread is running

Parameters

instance	the processing instance

Returns

0 if success, <0 on error

8.12 JSON syntax

JSON protocol to configure the process

8.12.1 Commands

```
{ "get" : `object` }
```

get values.
If `object` is null, get all parameters, and so show the possibilities

```
{ "set" : `object` }
```

set values.
`object` can be any subpart, only those will be changed

```
{ "change" : `object` }
```

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

Note

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

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

load the preset

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

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

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

delete the preset.
can also be an array of presetnames

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

Create a preset from factory and load it

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

Rename a preset

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

Import a preset

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

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

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

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

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

Undo last(s) `count` preset change

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

Redo last(s) `count` preset change

See also JSON syntax for internal setup

8.12.2 Answers

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

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

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

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

If there is an error, answer has the syntax

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

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

8.12.3 Examples

8.12.3.1 Get status

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

answer:

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

8.12.3.2 Get all changes since last request

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

answer:

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

8.12.3.3 Check license

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

answer:

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

8.12.3.4 Set Processor name

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

answer:

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

8.12.3.5 Get presetlist

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

answer:

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

8.12.3.6 Load a preset

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

answer:

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

8.12.3.7 Get information on preset on air

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

answer:

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

8.12.3.8 Set input gain

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

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

8.12.3.9 Set bypass

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

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

8.12.3.10 Export a preset

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

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

Note

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

8.12.3.11 Import a preset

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

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

8.13 JSON syntax for internal setup

JSON commands for internal setup and information.

This is used internally for the web server.

This API might change in the future.

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

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

8.13.1 Get process and modules information

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

answer:

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

8.13.2 Authentication management

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

Secret stored is

```
MD5(username+":"+realm+":"+password)
```

8.13.2.1 Difference between setup and admin account

There are 2 modes:

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

From JSON, changing setup_is_admin can change the mode.

8.13.2.2 Access encoding

Access is a combinaison of letters.

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

8.13.2.3 Change setup account

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

8.13.2.3.1 Set the setup/admin account

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

answer:

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

8.13.2.3.2 Set the setup/admin account with clear password

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

answer:

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

8.13.2.3.3 Remove the setup/admin account

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

answer:

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

8.13.2.4 Get user info

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

answer:

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

8.13.2.5 Add or modify a user

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

answer:

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

8.13.2.6 Removes a user

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

answer:

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

8.13.2.7 List users

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

answer:

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

8.13.2.8 Get/set unlogged access

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

answer:

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

8.13.2.9 Get realm

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

answer:

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

8.13.2.10 Get API version

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

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

8.13.2.11 Get setup is admin

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

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

8.13.2.12 Get/set setup account

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

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

8.13.2.13 Get/set custom storage

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

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

8.13.3 Get process' interface description

8.13.3.1 Get XML decription

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

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

Attention

The answer is XML, not JSON !

8.13.3.2 Get Process Image

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

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

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

8.13.3.3 Get Process Ico

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

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

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

8.14 Web server

Macros

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

Functions

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

8.14.1 Detailed Description

Run a web server to configure the processor

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

8.14.2 Macro Definition Documentation

8.14.2.1 SOUND4_INVALID_WEBSERVER_ID

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

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

Definition at line 1502 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 1507 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 1511 of file sound4.impact.cl.h.

8.14.3 Function Documentation

8.14.3.1 impact_Webserver()

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

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

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

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

Warning

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

Parameters

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

Returns

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

8.14.3.2 impact_Webserver_Status()

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

Get the web server status

Warning

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

Parameters

id	The web server Id, returned by impact_Webserver_tcp or impact_Webserver

Returns

a combinaison of SOUND4_WEBSERVER_HTTP_OK and SOUND4_WEBSERVER_HTTPS_OK depending on the server status

8.14.3.3 impact_Webserver_Stop()

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

Stop a web server started with impact_Webserver_tcp

Warning

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

Parameters

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

Returns

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

8.14.3.4 impact_Webserver_tcp()

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

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

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

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

Warning

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

Parameters

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

Returns

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

8.14.3.5 impact_Webserver_tcp2()

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

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

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

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

Warning

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

Parameters

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

Returns

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

8.15 Sample using C++ interface wrapper

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

8.15.1 Setup and sanity checks

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

8.15.2 Sample to create the processing instance

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

8.15.3 Communicating with the instance

See also

JSON syntax

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

8.15.4 Starting a web-server

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

8.15.5 Sample processing loop

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

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

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

8.16.1 Dynamic loading

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

8.16.2 Setup and sanity checks

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

8.16.3 Sample to create the processing instance

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

8.16.4 Communicating with the instance

See also

JSON syntax

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

8.16.5 Starting a web-server

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

8.16.6 Sample processing loop

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

8.16.7 Custom preset manager

```
class MyPresetManager : public sound4::impact::dyn::CPresetLoader {
public:
    MyPresetManager() = default;
    virtual ~MyPresetManager() = default;
    bool SetupPath(const std::filesystem::path& a_preset_path)
        {
            m_preset_path=a_preset_path;
            if ( !std::filesystem::exists(m_preset_path ) ) {
                return false;
            }
            std::error_code ec;
            if ( !std::filesystem::is_directory(m_preset_path, ec) ) {
                return false;
            }
            return true;
        }
    virtual bool IsReadOnly() override { return false; };
    virtual bool Exists(const std::filesystem::path &name) override
        {
            std::error_code ec;
            return std::filesystem::exists(GetRealPath(name), ec);
        }
    virtual bool Remove(const std::filesystem::path &name) override
        {
            std::error_code ec;
            return std::filesystem::remove(GetRealPath(name), ec);
        }
    virtual bool Rename(const std::filesystem::path &from, const std::filesystem::path &to) override
        {
            std::error_code ec;
            std::filesystem::rename(GetRealPath(from), GetRealPath(to), ec);
            return (!ec);
        }
    virtual std::vector<std::filesystem::path> GetAll() override
        {
            std::vector<std::filesystem::path> list;
            for (const auto & entry : std::filesystem::directory_iterator(m_preset_path.c_str(),
   std::filesystem::directory_options::skip_permission_denied)) {
                if (entry.is_regular_file()) {
                    list.push_back(std::filesystem::relative(entry.path(),m_preset_path).native());
                }
            }
            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);
    }
```

Chapter 9

Namespace Documentation

9.1 dyn Namespace Reference

9.1.1 Detailed Description

Dynamic loading namespace

9.2 sound4 Namespace Reference

Namespaces

- impact

9.2.1 Detailed Description

SOUND4 namespace

9.3 sound4::impact Namespace Reference

Namespaces

- dyn
- helper

Classes

- class CPresetLoader
- class CInstance

Typedefs

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

Variables

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

9.3.1 Detailed Description

impact namespace

9.3.2 Typedef Documentation

9.3.2.1 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.

9.3.2.2 LogSeverity

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

Log Severity levels

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

9.3.2.3 PresetChange_Kind

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

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

9.3.2.4 SampleFormat

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

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

9.3.3 Variable Documentation

9.3.3.1 ChunkFrames

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

Number of frames in a chunk

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

9.3.3.2 InputChannels

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

Number of float values in an input frame

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

9.3.3.3 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.

9.3.3.4 OutputChannels

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

Number of float values in an output frame

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

9.3.3.5 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.

9.3.3.6 SampleRate

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

Sample rate of library

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

9.4 sound4::impact::dyn Namespace Reference

Namespaces

- helper

Classes

- class CDynLib

  Dynamic library interface.
- class CPresetLoader
- class CInstance

Typedefs

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

Variables

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

9.4.1 Typedef Documentation

9.4.1.1 log_cb_t

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

Log callback type

Definition at line 697 of file sound4.impact.cl_dyn.hpp.

9.4.1.2 LogSeverity

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

Log Severity levels

Definition at line 693 of file sound4.impact.cl_dyn.hpp.

9.4.1.3 PresetChange_Kind

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

Definition at line 765 of file sound4.impact.cl_dyn.hpp.

9.4.1.4 SampleFormat

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

Definition at line 426 of file sound4.impact.cl_dyn.hpp.

9.4.2 Variable Documentation

9.4.2.1 ChunkFrames

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

Number of frames in a chunk

Definition at line 430 of file sound4.impact.cl_dyn.hpp.

9.4.2.2 InputChannels

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

Number of float values in an input frame

Definition at line 434 of file sound4.impact.cl_dyn.hpp.

9.4.2.3 InputSampleSize

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

Number of float values for an input chunk

Definition at line 442 of file sound4.impact.cl_dyn.hpp.

9.4.2.4 OutputChannels

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

Number of float values in an output frame

Definition at line 438 of file sound4.impact.cl_dyn.hpp.

9.4.2.5 OutputSampleSize

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

Number of float values for an output chunk

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

9.4.2.6 SampleRate

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

Sample rate of library

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

9.5 sound4::impact::dyn::helper Namespace Reference

Classes

- class DynFuncHolder

  Helper to wrap a function pointer.
- class CDynLoader

  Helper to load dynamic library.
- struct SampleFormat

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

Typedefs

- using DynLib_t = HMODULE
- using DynFunc_t = FARPROC

9.5.1 Detailed Description

helper functions for unicode strings in Windows

9.5.2 Typedef Documentation

9.5.2.1 DynFunc_t

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

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

9.5.2.2 DynLib_t

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

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

9.6 sound4::impact::helper Namespace Reference

Classes

- struct SampleFormat

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

9.6.1 Detailed Description

helper functions for unicode strings in Windows

9.7 std Namespace Reference

Chapter 10

Class Documentation

10.1 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.

10.1.1 Detailed Description

Low-level JSON client.

See also

JSON syntax

Definition at line 770 of file sound4.impact.cl.hpp.

10.1.2 Constructor & Destructor Documentation

10.1.2.1 CClient() [1/2]

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

Definition at line 772 of file sound4.impact.cl.hpp.

10.1.2.2 CClient() [2/2]

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

10.1.2.3 ~CClient()

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

Definition at line 781 of file sound4.impact.cl.hpp.

10.1.3 Member Function Documentation

10.1.3.1 operator=()

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

10.1.3.2 ProcessJson()

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

Parameters

request	the JSON request.

See also

JSON syntax

Parameters

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

Returns

the JSON answer

Definition at line 795 of file sound4.impact.cl.hpp.

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

- sound4.impact.cl.hpp

10.2 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.

10.2.1 Detailed Description

Low-level JSON client.

See also

JSON syntax

Definition at line 1170 of file sound4.impact.cl_dyn.hpp.

10.2.2 Constructor & Destructor Documentation

10.2.2.1 CClient() [1/2]

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

Definition at line 1172 of file sound4.impact.cl_dyn.hpp.

10.2.2.2 CClient() [2/2]

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

10.2.2.3 ~CClient()

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

Definition at line 1182 of file sound4.impact.cl_dyn.hpp.

10.2.3 Member Function Documentation

10.2.3.1 operator=()

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

10.2.3.2 ProcessJson()

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

Parameters

request	the JSON request.

See also

JSON syntax

Parameters

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

Returns

the JSON answer

Definition at line 1196 of file sound4.impact.cl_dyn.hpp.

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

- sound4.impact.cl_dyn.hpp

10.3 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=std::filesystem::current_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

10.3.1 Detailed Description

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

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

10.3.2 Constructor & Destructor Documentation

10.3.2.1 CDynLib()

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

10.3.3 Member Function Documentation

10.3.3.1 IsOk()

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

Returns

true if OK.

false if not OK.

Definition at line 416 of file sound4.impact.cl_dyn.hpp.

10.3.3.2 Load()

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

Parameters

path	the path of the library to load.

Returns

true if library was loaded correctly.

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

Definition at line 302 of file sound4.impact.cl_dyn.hpp.

10.3.4 Member Data Documentation

10.3.4.1 AddAudio

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

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

10.3.4.2 AddPadAudio

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

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

10.3.4.3 AudioConvertFrom

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

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

10.3.4.4 AudioConvertTo

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

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

10.3.4.5 AudioMonoFromLiveStereo

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

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

10.3.4.6 AudioMonoToLiveStereo

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

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

10.3.4.7 DeleteClient

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

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

10.3.4.8 ExitProcess

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

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

10.3.4.9 FreeJsonAnswer

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

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

10.3.4.10 FreeMetadataInfos

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

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

10.3.4.11 FreeParameters

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

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

10.3.4.12 FreeParameterValue

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

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

10.3.4.13 GetAudio

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

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

10.3.4.14 GetAudioInputCount

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

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

10.3.4.15 GetAudioOutputCount

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

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

10.3.4.16 GetBufferIn

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

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

10.3.4.17 GetBufferOut

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

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

10.3.4.18 GetBytesFromFormat

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

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

10.3.4.19 GetChannelCount

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

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

10.3.4.20 GetChunkSizeInFrames

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

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

10.3.4.21 GetEstimatedDelay

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

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

10.3.4.22 GetFormatFromName

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

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

10.3.4.23 GetFormatName

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

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

10.3.4.24 GetMaxPacketFrame

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

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

10.3.4.25 GetMetadataInfos

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

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

10.3.4.26 GetOutputCount

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

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

10.3.4.27 GetParameter

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

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

10.3.4.28 GetPossibleChunkSizeInFrames

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

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

10.3.4.29 GetProcessChunkFrames

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

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

10.3.4.30 GetSampleRate

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

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

10.3.4.31 GetVersion

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

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

10.3.4.32 InitProcess

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

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

10.3.4.33 InitProcess2

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

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

10.3.4.34 InitProcess3

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

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

10.3.4.35 MonoToStereo

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

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

10.3.4.36 MonoToStereo_Planar

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

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

10.3.4.37 NewClient

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

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

10.3.4.38 NewParameters

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

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

10.3.4.39 PresetManager_InformChange

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

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

10.3.4.40 ProcessAudio

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

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

10.3.4.41 ProcessAudio_Planar

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

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

10.3.4.42 ProcessJson

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

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

10.3.4.43 SaveState

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

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

10.3.4.44 SetLoggerCallback

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

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

10.3.4.45 SetLogSeverity

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

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

10.3.4.46 SetMetadata

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

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

10.3.4.47 SetParameter

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

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

10.3.4.48 SetPresetManager

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

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

10.3.4.49 StartUpdateThread

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

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

10.3.4.50 StereoToMono

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

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

10.3.4.51 StereoToMono_Planar

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

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

10.3.4.52 StopUpdateThread

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

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

10.3.4.53 TerminateProcess

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

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

10.3.4.54 WaitUpdateThreadReady

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

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

10.3.4.55 Webserver

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

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

10.3.4.56 Webserver_Status

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

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

10.3.4.57 Webserver_Stop

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

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

10.3.4.58 Webserver_tcp

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

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

10.3.4.59 Webserver_tcp2

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

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

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

- sound4.impact.cl_dyn.hpp

10.4 sound4::impact::dyn::helper::CDynLoader Class Reference

Helper to load dynamic library.

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

Public Member Functions

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

Static Public Member Functions

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

10.4.1 Detailed Description

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

10.4.2 Constructor & Destructor Documentation

10.4.2.1 CDynLoader() [1/3]

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

10.4.2.2 CDynLoader() [2/3]

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

10.4.2.3 CDynLoader() [3/3]

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

10.4.2.4 ~CDynLoader()

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

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

10.4.3 Member Function Documentation

10.4.3.1 Close()

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

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

10.4.3.2 GetSymbol()

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

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

10.4.3.3 GetSymbol_safe()

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

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

10.4.3.4 GetThisLibraryPath()

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

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

10.4.3.5 IsOk()

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

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

10.4.3.6 Load()

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

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

10.4.3.7 operator bool()

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

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

10.4.3.8 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

10.5 sound4::impact::CInstance Class Reference

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

Classes

- class CClient

Public Member Functions

- CInstance ()
- CInstance (const CInstance &)=delete
- CInstance & operator= (const CInstance &)=delete
- ~CInstance ()
- bool IsOk () const
- void SetParam (const std::string &name, const std::string &value)
- std::string GetParam (const std::string &name)
- void SetPresetManager (CPresetLoader *preset_manager)
- bool Create (const std::string &LoginKey, const std::string &RadioName, const std::string &AWS_Access_ID, const std::string &AWS_Access_Secret, const std::filesystem::path &save_path, int json_port=0, unsigned int frames_per_chunk=64)
- unsigned int GetChunkFrames ()
- void PresetManager_InformChange (const std::filesystem::path &relative_path, PresetChange_Kind change_kind)
- void SetMetadata (const std::string &key, const char *value)
- std::vector< std::tuple< std::string, std::string > > GetMetadataInfos ()
- virtual void OnUpdateThreadStart ()
- virtual void OnUpdateThreadStop ()
- bool StartWebServer (int http_port, int https_port=0)
- void StopWebServer ()
- int Terminate ()
- void Stop ()
- unsigned int GetEstimatedDelay ()
- std::array< float, InputSampleSize > & GetBufferIn ()
- std::array< float, OutputSampleSize > & GetBufferOut ()
- void ProcessAudio ()
- void ProcessAudio_Planar (float const *const *input, float *const *output)
- template<typename T >
  std::vector< T > ProcessAnyAudio (const std::vector< T > input)
- std::shared_ptr< CClient > NewClient ()

Protected Member Functions

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

10.5.1 Detailed Description

Instance handling class

Definition at line 385 of file sound4.impact.cl.hpp.

10.5.2 Constructor & Destructor Documentation

10.5.2.1 CInstance() [1/2]

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

Creates the instance, with default parameters

Definition at line 390 of file sound4.impact.cl.hpp.

10.5.2.2 CInstance() [2/2]

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

10.5.2.3 ~CInstance()

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

Destructor, stopping any running tasks

Definition at line 401 of file sound4.impact.cl.hpp.

10.5.3 Member Function Documentation

10.5.3.1 AddAudio()

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

Definition at line 826 of file sound4.impact.cl.hpp.

10.5.3.2 Create()

```
bool sound4::impact::CInstance::Create (
            const std::string & LoginKey,
            const std::string & RadioName,
            const std::string & AWS_Access_ID,
            const std::string & AWS_Access_Secret,
            const std::filesystem::path & save_path,
            int json_port = 0,
            unsigned int frames_per_chunk = 64 ) [inline]
```

Creates and initialize the instance, and starts its update thread

See also

impact_InitProcess2

Parameters

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

Returns

false if failed

Definition at line 487 of file sound4.impact.cl.hpp.

10.5.3.3 GetAudio()

```
template<typename T >
unsigned int sound4::impact::CInstance::GetAudio (
            T * payload,
            unsigned int max_nFrame )  [inline], [protected]
```

Definition at line 832 of file sound4.impact.cl.hpp.

10.5.3.4 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 681 of file sound4.impact.cl.hpp.

10.5.3.5 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 698 of file sound4.impact.cl.hpp.

10.5.3.6 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.

10.5.3.7 GetEstimatedDelay()

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

Gets an estimated delay of current processing

See also

impact_GetEstimatedDelay

Definition at line 665 of file sound4.impact.cl.hpp.

10.5.3.8 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 568 of file sound4.impact.cl.hpp.

10.5.3.9 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.

10.5.3.10 IsOk()

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

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

Definition at line 409 of file sound4.impact.cl.hpp.

10.5.3.11 NewClient()

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

Creates a low-level JSON client

See also

JSON syntax

Returns

a new client to communicate with the processing

Definition at line 818 of file sound4.impact.cl.hpp.

10.5.3.12 OnUpdateThreadStart()

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

Definition at line 588 of file sound4.impact.cl.hpp.

10.5.3.13 OnUpdateThreadStop()

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

Definition at line 590 of file sound4.impact.cl.hpp.

10.5.3.14 operator=()

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

10.5.3.15 PresetManager_InformChange()

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

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

Parameters

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

Definition at line 538 of file sound4.impact.cl.hpp.

10.5.3.16 ProcessAnyAudio()

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

Process audio (different formats, less optimized)

Supports for int16_t, int32_t and float (32) in native endianness

Attention

the return count is independent from the input count due to IMPACT_AUDIOFRAME_COUNT

Template Parameters

T	the C type for one audio sample

Parameters

input	audio to process

Returns

processed audio

Definition at line 746 of file sound4.impact.cl.hpp.

10.5.3.17 ProcessAudio()

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

Process one audio chunk, from GetBufferIn() to GetBufferOut()

See also

impact_ProcessAudio

Definition at line 709 of file sound4.impact.cl.hpp.

10.5.3.18 ProcessAudio_Planar()

```
void sound4::impact::CInstance::ProcessAudio_Planar (
            float const *const * input,
            float *const * output ) [inline]
```

Process one audio chunk, but with planar audio

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

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

See also

impact_ProcessAudio_Planar

Parameters

input	has GetAudioInputCount() ∗ GetChannelCount() channels
output	has GetAudioOutputCount() ∗ GetChannelCount() channels

Definition at line 728 of file sound4.impact.cl.hpp.

10.5.3.19 SetMetadata()

```
void sound4::impact::CInstance::SetMetadata (
            const std::string & key,
            const char * value )  [inline]
```

Adds the specific metadata to the process.

It can come from any thread, and will be used as soon as possible. Setting a null value erase the metadata, which is different from setting an empty string.

Parameters

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

Definition at line 554 of file sound4.impact.cl.hpp.

10.5.3.20 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.

10.5.3.21 SetPresetManager()

```
void sound4::impact::CInstance::SetPresetManager (
            CPresetLoader * preset_manager )  [inline]
```

Sets a custom preset manager Optional, this overrides the `save_path` in Create

See also

impact_SetPresetManager

Must be done before Create

Parameters

preset_manager	The custom preset manager

Definition at line 445 of file sound4.impact.cl.hpp.

10.5.3.22 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 598 of file sound4.impact.cl.hpp.

10.5.3.23 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 646 of file sound4.impact.cl.hpp.

10.5.3.24 StopWebServer()

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

Stops the Web server

See also

impact_Webserver_Stop

Definition at line 620 of file sound4.impact.cl.hpp.

10.5.3.25 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 635 of file sound4.impact.cl.hpp.

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

- sound4.impact.cl.hpp

10.6 sound4::impact::dyn::CInstance Class Reference

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

Classes

- class CClient

Public Member Functions

- CInstance (CDynLib &dynlib)
- CInstance (const CInstance &)=delete
- CInstance & operator= (const CInstance &)=delete
- ~CInstance ()
- bool IsOk () const
- void SetParam (const std::string &name, const std::string &value)
- std::string GetParam (const std::string &name)
- void SetPresetManager (CPresetLoader *preset_manager)
- bool Create (const std::string &LoginKey, const std::string &RadioName, const std::string &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)
- std::vector< std::tuple< std::string, std::string > > GetMetadataInfos ()
- virtual void OnUpdateThreadStart ()
- virtual void OnUpdateThreadStop ()
- bool StartWebServer (int http_port, int https_port=0)
- void StopWebServer ()
- int Terminate ()
- void Stop ()
- unsigned int GetEstimatedDelay ()
- std::array< float, InputSampleSize > & GetBufferIn ()
- std::array< float, OutputSampleSize > & GetBufferOut ()
- void ProcessAudio ()
- void ProcessAudio_Planar (float const *const *input, float *const *output)
- template<typename T >
  std::vector< T > ProcessAnyAudio (const std::vector< T > input)
- std::shared_ptr< CClient > NewClient ()

Protected Member Functions

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

10.6.1 Detailed Description

Instance handling class

Definition at line 770 of file sound4.impact.cl_dyn.hpp.

10.6.2 Constructor & Destructor Documentation

10.6.2.1 CInstance() [1/2]

```
sound4::impact::dyn::CInstance::CInstance (
            CDynLib & dynlib )  [inline]
```

Creates the instance, with default parameters

Definition at line 775 of file sound4.impact.cl_dyn.hpp.

10.6.2.2 CInstance() [2/2]

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

10.6.2.3 ~CInstance()

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

Destructor, stopping any running tasks

Definition at line 787 of file sound4.impact.cl_dyn.hpp.

10.6.3 Member Function Documentation

10.6.3.1 AddAudio()

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

Definition at line 1228 of file sound4.impact.cl_dyn.hpp.

10.6.3.2 Create()

```
bool sound4::impact::dyn::CInstance::Create (
            const std::string & LoginKey,
            const std::string & RadioName,
            const std::string & AWS_Access_ID,
            const std::string & AWS_Access_Secret,
            const std::filesystem::path & save_path,
            int json_port = 0,
            unsigned int frames_per_chunk = 64 )  [inline]
```

Creates and initialize the instance, and starts its update thread

See also

impact_InitProcess2

Parameters

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

Returns

false if failed

Definition at line 875 of file sound4.impact.cl_dyn.hpp.

10.6.3.3 GetAudio()

```
template<typename T >
unsigned int sound4::impact::dyn::CInstance::GetAudio (
            T * payload,
            unsigned int max_nFrame )  [inline], [protected]
```

Definition at line 1234 of file sound4.impact.cl_dyn.hpp.

10.6.3.4 GetBufferIn()

```
std::array<float,InputSampleSize>& sound4::impact::dyn::CInstance::GetBufferIn ( )  [inline]
```

Gets the input buffer used in ProcessAudio

Input has GetAudioInputCount() ∗ GetChannelCount() channels

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

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

Definition at line 1081 of file sound4.impact.cl_dyn.hpp.

10.6.3.5 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 1098 of file sound4.impact.cl_dyn.hpp.

10.6.3.6 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 918 of file sound4.impact.cl_dyn.hpp.

10.6.3.7 GetEstimatedDelay()

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

Gets an estimated delay of current processing

See also

impact_GetEstimatedDelay

Definition at line 1065 of file sound4.impact.cl_dyn.hpp.

10.6.3.8 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 966 of file sound4.impact.cl_dyn.hpp.

10.6.3.9 GetParam()

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

Gets a parameter value

See also

impact_GetParameter

Definition at line 813 of file sound4.impact.cl_dyn.hpp.

10.6.3.10 IsOk()

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

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

Definition at line 795 of file sound4.impact.cl_dyn.hpp.

10.6.3.11 NewClient()

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

Creates a low-level JSON client

See also

JSON syntax

Returns

a new client to communicate with the processing

Definition at line 1220 of file sound4.impact.cl_dyn.hpp.

10.6.3.12 OnUpdateThreadStart()

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

Definition at line 988 of file sound4.impact.cl_dyn.hpp.

10.6.3.13 OnUpdateThreadStop()

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

Definition at line 990 of file sound4.impact.cl_dyn.hpp.

10.6.3.14 operator=()

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

10.6.3.15 PresetManager_InformChange()

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

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

Parameters

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

Definition at line 932 of file sound4.impact.cl_dyn.hpp.

10.6.3.16 ProcessAnyAudio()

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

Process audio (different formats, less optimized)

Supports for int16_t, int32_t and float (32) in native endianness

Attention

the return count is independent from the input count due to IMPACT_AUDIOFRAME_COUNT

Template Parameters

T	the C type for one audio sample

Parameters

input	audio to process

Returns

processed audio

Definition at line 1146 of file sound4.impact.cl_dyn.hpp.

10.6.3.17 ProcessAudio()

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

Process one audio chunk, from GetBufferIn() to GetBufferOut()

See also

impact_ProcessAudio

Definition at line 1109 of file sound4.impact.cl_dyn.hpp.

10.6.3.18 ProcessAudio_Planar()

```
void sound4::impact::dyn::CInstance::ProcessAudio_Planar (
            float const *const * input,
            float *const * output ) [inline]
```

Process one audio chunk, but with planar audio

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

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

See also

impact_ProcessAudio_Planar

Parameters

input	has GetAudioInputCount() ∗ GetChannelCount() channels
output	has GetAudioOutputCount() ∗ GetChannelCount() channels

Definition at line 1128 of file sound4.impact.cl_dyn.hpp.

10.6.3.19 SetMetadata()

```
void sound4::impact::dyn::CInstance::SetMetadata (
            const std::string & key,
            const char * value )  [inline]
```

Adds the specific metadata to the process.

It can come from any thread, and will be used as soon as possible. Setting a null value erase the metadata, which is different from setting an empty string.

Parameters

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

Definition at line 950 of file sound4.impact.cl_dyn.hpp.

10.6.3.20 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 803 of file sound4.impact.cl_dyn.hpp.

10.6.3.21 SetPresetManager()

```
void sound4::impact::dyn::CInstance::SetPresetManager (
            CPresetLoader * preset_manager )  [inline]
```

Sets a custom preset manager Optional, this overrides the `save_path` in Create

See also

impact_SetPresetManager

Must be done before Create

Parameters

preset_manager	The custom preset manager

Definition at line 842 of file sound4.impact.cl_dyn.hpp.

10.6.3.22 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 998 of file sound4.impact.cl_dyn.hpp.

10.6.3.23 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 1046 of file sound4.impact.cl_dyn.hpp.

10.6.3.24 StopWebServer()

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

Stops the Web server

See also

impact_Webserver_Stop

Definition at line 1020 of file sound4.impact.cl_dyn.hpp.

10.6.3.25 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 1035 of file sound4.impact.cl_dyn.hpp.

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

- sound4.impact.cl_dyn.hpp

10.7 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

10.7.1 Detailed Description

Custom preset handler helper Derive from this class and call CInstance::SetPresetManager

Definition at line 367 of file sound4.impact.cl.hpp.

10.7.2 Constructor & Destructor Documentation

10.7.2.1 CPresetLoader()

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

10.7.2.2 ~CPresetLoader()

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

10.7.3 Member Function Documentation

10.7.3.1 Exists()

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

10.7.3.2 GetAll()

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

10.7.3.3 IsReadOnly()

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

10.7.3.4 Read()

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

10.7.3.5 Remove()

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

10.7.3.6 Rename()

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

10.7.3.7 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

10.8 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

10.8.1 Detailed Description

Custom preset handler helper Derive from this class and call CInstance::SetPresetManager

Definition at line 752 of file sound4.impact.cl_dyn.hpp.

10.8.2 Constructor & Destructor Documentation

10.8.2.1 CPresetLoader()

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

10.8.2.2 ~CPresetLoader()

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

10.8.3 Member Function Documentation

10.8.3.1 Exists()

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

Implemented in MyPresetManager.

10.8.3.2 GetAll()

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

Implemented in MyPresetManager.

10.8.3.3 IsReadOnly()

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

Implemented in MyPresetManager.

10.8.3.4 Read()

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

Implemented in MyPresetManager.

10.8.3.5 Remove()

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

Implemented in MyPresetManager.

10.8.3.6 Rename()

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

Implemented in MyPresetManager.

10.8.3.7 Write()

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

Implemented in MyPresetManager.

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

- sound4.impact.cl_dyn.hpp

10.9 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

10.9.1 Detailed Description

template<typename T>
class sound4::impact::dyn::helper::DynFuncHolder< T >

Parameters

T	is the function type, you may use decltype(fn)

Definition at line 86 of file sound4.impact.cl_dyn.hpp.

10.9.2 Constructor & Destructor Documentation

10.9.2.1 DynFuncHolder() [1/2]

```
template<typename T >
sound4::impact::dyn::helper::DynFuncHolder< T >::DynFuncHolder ( ) [default]
```

10.9.2.2 DynFuncHolder() [2/2]

```
template<typename T >
sound4::impact::dyn::helper::DynFuncHolder< T >::DynFuncHolder (
            DynFunc_t a_ptr ) [inline]
```

Definition at line 91 of file sound4.impact.cl_dyn.hpp.

10.9.3 Member Function Documentation

10.9.3.1 IsOk()

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

Definition at line 93 of file sound4.impact.cl_dyn.hpp.

10.9.3.2 operator bool()

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

Definition at line 92 of file sound4.impact.cl_dyn.hpp.

10.9.3.3 operator T∗()

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

Definition at line 94 of file sound4.impact.cl_dyn.hpp.

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

- sound4.impact.cl_dyn.hpp

10.10 impact_CClientInstance Struct Reference

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

10.10.1 Detailed Description

Abstract pointer for a client instance

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

- sound4.impact.cl.h

10.11 impact_CInstance Struct Reference

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

10.11.1 Detailed Description

Abstract holder for a process instance

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

- sound4.impact.cl.h

10.12 impact_CParameters Struct Reference

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

10.12.1 Detailed Description

Abstract holder for a process instance parameters

Definition at line 307 of file sound4.impact.cl.h.

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

- sound4.impact.cl.h

10.13 MyPresetManager Class Reference

[Custom preset manager]

Inherits sound4::impact::dyn::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

10.13.1 Detailed Description

Definition at line 119 of file sound4.impact.cl-pipe-dyn.cpp.

10.13.2 Constructor & Destructor Documentation

10.13.2.1 MyPresetManager()

```
MyPresetManager::MyPresetManager ( ) [default]
```

10.13.2.2 ~MyPresetManager()

```
virtual MyPresetManager::~MyPresetManager ( ) [virtual], [default]
```

10.13.3 Member Function Documentation

10.13.3.1 Exists()

```
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.

10.13.3.2 GetAll()

```
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.

10.13.3.3 IsReadOnly()

```
virtual bool MyPresetManager::IsReadOnly ( )  [inline], [override], [virtual]
```

Implements sound4::impact::dyn::CPresetLoader.

Definition at line 136 of file sound4.impact.cl-pipe-dyn.cpp.

10.13.3.4 Read()

```
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.

10.13.3.5 Remove()

```
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.

10.13.3.6 Rename()

```
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.

10.13.3.7 SetupPath()

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

Definition at line 124 of file sound4.impact.cl-pipe-dyn.cpp.

10.13.3.8 Write()

```
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.

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

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

10.14 sound4::impact::dyn::helper::SampleFormat< T > Struct Template Reference

Helper for Sample format types.

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

10.14.1 Detailed Description

template<typename T>
struct sound4::impact::dyn::helper::SampleFormat< T >

Template Parameters

T	the C type for one audio sample

Definition at line 489 of file sound4.impact.cl_dyn.hpp.

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

- sound4.impact.cl_dyn.hpp

10.15 sound4::impact::helper::SampleFormat< T > Struct Template Reference

Helper for Sample format types.

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

10.15.1 Detailed Description

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

Template Parameters

T	the C type for one audio sample

Definition at line 116 of file sound4.impact.cl.hpp.

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

- sound4.impact.cl.hpp

10.16 sound4::impact::dyn::helper::SampleFormat< float > Struct Reference

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

Public Attributes

- const impact_SampleFormat format = F32_NATIVE

10.16.1 Detailed Description

Definition at line 500 of file sound4.impact.cl_dyn.hpp.

10.16.2 Member Data Documentation

10.16.2.1 format

```
const impact_SampleFormat sound4::impact::dyn::helper::SampleFormat< float >::format = F32_NATIVE
```

Definition at line 501 of file sound4.impact.cl_dyn.hpp.

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

- sound4.impact.cl_dyn.hpp

10.17 sound4::impact::helper::SampleFormat< float > Struct Reference

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

Public Attributes

- const impact_SampleFormat format = F32_NATIVE

10.17.1 Detailed Description

Definition at line 127 of file sound4.impact.cl.hpp.

10.17.2 Member Data Documentation

10.17.2.1 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

10.18 sound4::impact::dyn::helper::SampleFormat< int16_t > Struct Reference

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

Public Attributes

- const impact_SampleFormat format = S16_NATIVE

10.18.1 Detailed Description

Definition at line 492 of file sound4.impact.cl_dyn.hpp.

10.18.2 Member Data Documentation

10.18.2.1 format

```
const impact_SampleFormat sound4::impact::dyn::helper::SampleFormat< int16_t >::format =
S16_NATIVE
```

Definition at line 493 of file sound4.impact.cl_dyn.hpp.

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

- sound4.impact.cl_dyn.hpp

10.19 sound4::impact::helper::SampleFormat< int16_t > Struct Reference

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

Public Attributes

- const impact_SampleFormat format = S16_NATIVE

10.19.1 Detailed Description

Definition at line 119 of file sound4.impact.cl.hpp.

10.19.2 Member Data Documentation

10.19.2.1 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

10.20 sound4::impact::dyn::helper::SampleFormat< int32_t > Struct Reference

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

Public Attributes

- const impact_SampleFormat format = S32_NATIVE

10.20.1 Detailed Description

Definition at line 496 of file sound4.impact.cl_dyn.hpp.

10.20.2 Member Data Documentation

10.20.2.1 format

```
const impact_SampleFormat sound4::impact::dyn::helper::SampleFormat< int32_t >::format =
S32_NATIVE
```

Definition at line 497 of file sound4.impact.cl_dyn.hpp.

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

- sound4.impact.cl_dyn.hpp

10.21 sound4::impact::helper::SampleFormat< int32_t > Struct Reference

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

Public Attributes

- const impact_SampleFormat format = S32_NATIVE

10.21.1 Detailed Description

Definition at line 123 of file sound4.impact.cl.hpp.

10.21.2 Member Data Documentation

10.21.2.1 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

Chapter 11

File Documentation

11.1 sound4.impact.cl-pipe-cxx.cpp File Reference

C++ sample using C++ interface wrapper.

```
#include <unistd.h>
#include <iostream>
#include <strings.h>
#include <sys/types.h>
#include <filesystem>
#include <io.h>
#include <fcntl.h>
#include "sound4.impact.cl.hpp"
```

Functions

- std::string GetEnv (const std::string &name)
- void ProcessLoop (sound4::impact::CInstance &instance)
- void Usage (const char *progname)
- void MyLogger (sound4::impact::LogSeverity severity, const std::string &msg)
- int main (int argc, const char **argv)

11.1.1 Function Documentation

11.1.1.1 GetEnv()

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

Definition at line 39 of file sound4.impact.cl-pipe-cxx.cpp.

11.1.1.2 main()

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

[Setup and sanity checks]

[Setup and sanity checks]

[Create the processing instance]

[Create the processing instance]

[Communicating with the instance]

[Communicating with the instance]

Definition at line 104 of file sound4.impact.cl-pipe-cxx.cpp.

11.1.1.3 MyLogger()

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

Definition at line 99 of file sound4.impact.cl-pipe-cxx.cpp.

11.1.1.4 ProcessLoop()

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

[Processing Loop]

[Processing Loop]

Definition at line 48 of file sound4.impact.cl-pipe-cxx.cpp.

11.1.1.5 Usage()

```
void Usage (
            const char * progname )
```

Definition at line 79 of file sound4.impact.cl-pipe-cxx.cpp.

11.2 sound4.impact.cl-pipe-cxx.cpp

```
00001
00004 #include <unistd.h>
00005 #include <iostream>
00006 #include <strings.h>
00007 #include <sys/types.h>
00008 #include <filesystem>
00009 #if defined(_WIN32) || defined(WIN32)
00010     #include <io.h>
00011     #include <fcntl.h>
00012 #endif
00013
00014 #include "sound4.impact.cl.hpp"
00015
00016 // ------------------------------------------------------------
00039 std::string GetEnv(const std::string& name)
00040 {
00041     char *data=getenv(name.c_str());
00042     if (!data) return {};
00043     return std::string(data);
00044 }
00045
00046
00047 // Reads from stdin, write to stdout
00048 void ProcessLoop(sound4::impact::CInstance &instance)
00049 {
00050 bool stopthread=false;
00051
00052     #if defined(_WIN32) || defined(WIN32)
00053         // Windows: The stdin, stdout, and stderr streams always open in text mode by default
00054         _setmode(_fileno(stdin), _O_BINARY);
00055         _setmode(_fileno(stdout), _O_BINARY);
00056     #endif
00057
00059     while (!stopthread)
00060     {
00061         // Read a chunk from stdin
00062         if (std::fread(instance.GetBufferIn().data(), sizeof(float), sound4::impact::InputSampleSize,
       stdin) != sound4::impact::InputSampleSize) {
00063             std::cerr « "ProcessLoop: input stopped"« std::endl;
00064             stopthread=true;
00065             break;
00066         }
00067         if (stopthread) break;
00068         // Process it
00069         instance.ProcessAudio();
00070         // Write it to stdout
00071         if (fwrite(instance.GetBufferOut().data(), sizeof(float), sound4::impact::OutputSampleSize,
       stdout) != sound4::impact::OutputSampleSize) {
00072             std::cerr « "Failed to write audio output" « std::endl;
00073             stopthread=true;
00074         }
00075     }
00077 }
00078
00079 void Usage(const char *progname)
00080 {
00081     std::cerr « "Usage: " « progname « std::endl;;
00082     std::cerr « "  Reads audio from stdin and return processed audio in stdout" « std::endl;
00083     std::cerr « "Environment:"« std::endl;;
00084     std::cerr « " - S4STORE_PATH            : path to store config [default: (no storage)]"«
       std::endl;;
00085     std::cerr « " - S4LOGINKEY              : SOUND4 Login key (or S4USERID)"« std::endl;;
00086     std::cerr « " - RADIO_NAME              : SOUND4 Radio Name"« std::endl;;
00087     std::cerr « " - S4_AWS_ACCESS_KEY_ID    : SOUND4 Access Key ID"« std::endl;;
00088     std::cerr « " - S4_AWS_SECRET_ACCESS_KEY: SOUND4 Access Key Secret"« std::endl;;
00089     std::cerr « std::endl;
00090     std::cerr « "Example:"« std::endl;
00091     std::cerr « "ffmpeg -re -i file.wav -ar "«sound4::impact::GetSampleRate()«" -ac
       "«sound4::impact::GetChannelCount()«" -f f32le - | "«progname«" | ffplay -f f32le -ar
       "«sound4::impact::GetSampleRate()«" -ac "«sound4::impact::GetChannelCount()«" -i -" « std::endl;;
00092     #if defined(_WIN32) || defined(WIN32)
00093         std::cerr « "ffmpeg -f dshow -i audio=\"Microphone\" -ar "«sound4::impact::GetSampleRate()«"
       -ac "«sound4::impact::GetChannelCount()«" -f f32le - | "«progname«" | ffplay -f f32le -ar
       "«sound4::impact::GetSampleRate()«" -ac "«sound4::impact::GetChannelCount()«" -i -" « std::endl;;
00094     #else
00095         std::cerr « "ffmpeg -f pulse -i default -ar "«sound4::impact::GetSampleRate()«" -ac
       "«sound4::impact::GetChannelCount()«" -f f32le - | "«progname«" | ffplay -f f32le -ar
       "«sound4::impact::GetSampleRate()«" -ac "«sound4::impact::GetChannelCount()«" -i -" « std::endl;;
00096     #endif
00097 }
00098
00099 void MyLogger(sound4::impact::LogSeverity severity,const std::string& msg)
00100 {
00101     std::cerr « msg « std::endl;
00102 }
```

```
int main(int argc, const char **argv)
{
#if (IMPACT_HAS_WEBSERVER)
    int listen_port = 8080;
#endif // (IMPACT_HAS_WEBSERVER)
    auto store_path_str=GetEnv("S4STORE_PATH");

    // Get credentials from environment
    auto LoginKey=GetEnv("S4LOGINKEY");
    if (LoginKey.empty()) {
        LoginKey=GetEnv("S4USERID");
    }
    auto RadioName=GetEnv("RADIO_NAME");
    auto KeyId=GetEnv("S4_AWS_ACCESS_KEY_ID");
    auto AccessKey=GetEnv("S4_AWS_SECRET_ACCESS_KEY");

    const char *progname=argv[0];

    if (argc>1) {
        if ((strcasecmp(argv[1],"-h")==0) || (strcasecmp(argv[1],"--help")==0)) {
            Usage(progname);
            return 0;
        }
    }
    if (LoginKey.empty() || RadioName.empty() || KeyId.empty() || AccessKey.empty()) {
        std::cerr « "Missing credentials in environment" « std::endl;
        Usage(progname);
        return 1;
    }
    std::filesystem::path store_path;
    if (!store_path_str.empty()) {
        store_path = std::filesystem::path(store_path_str);
        if (!std::filesystem::exists(store_path) || !std::filesystem::is_directory(store_path)) {
            std::cerr « "Storage folder '"«store_path.c_str()«"' is not a valid directory" «
std::endl;
            Usage(progname);
            return 1;
        }
    }

    sound4::impact::SetLogSeverity(verbose5);
    sound4::impact::SetLoggerCallback(MyLogger);

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

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

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

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

    ProcessLoop(instance);

    std::cerr « "Stopping now" « std::endl;
    instance.Stop();

    std::cerr « "End" « std::endl;
    return 0;
}
```

11.3 sound4.impact.cl-pipe-dyn.cpp File Reference

C++ sample using C++ interface wrapper with dynamic library loading.

```
#include <unistd.h>
#include <iostream>
#include <strings.h>
#include <sys/types.h>
#include <filesystem>
#include <fstream>
#include <sstream>
#include <io.h>
#include <fcntl.h>
#include "sound4.impact.cl_dyn.hpp"
```

Classes

- class MyPresetManager

  [Custom preset manager]

Functions

- std::string GetEnv (const std::string &name)
- void ProcessLoop (sound4::impact::dyn::CInstance &instance)
- void Usage (const char *progname, sound4::impact::dyn::CDynLib &dynlib)
- void MyLogger (sound4::impact::dyn::LogSeverity severity, const std::string &msg)
- int main (int argc, const char **argv)

  [Custom preset manager]

11.3.1 Function Documentation

11.3.1.1 GetEnv()

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

Definition at line 51 of file sound4.impact.cl-pipe-dyn.cpp.

11.3.1.2 main()

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

[Dynamic loading]

[Dynamic loading]

[Setup and sanity checks]

[Setup and sanity checks]

[Custom preset manager loading]

[Custom preset manager loading]

[Create the processing instance]

[Create the processing instance]

[Communicating with the instance]

[Communicating with the instance]

Definition at line 194 of file sound4.impact.cl-pipe-dyn.cpp.

11.3.1.3 MyLogger()

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

Definition at line 113 of file sound4.impact.cl-pipe-dyn.cpp.

11.3.1.4 ProcessLoop()

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

[Processing Loop]

[Processing Loop]

Definition at line 60 of file sound4.impact.cl-pipe-dyn.cpp.

11.3.1.5 Usage()

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

Definition at line 91 of file sound4.impact.cl-pipe-dyn.cpp.

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

```
00001
00004 #include <unistd.h>
00005 #include <iostream>
00006 #include <strings.h>
00007 #include <sys/types.h>
00008 #include <filesystem>
00009 #include <fstream>
00010 #include <sstream>
00011 #if defined(_WIN32) || defined(WIN32)
00012     #include <io.h>
00013     #include <fcntl.h>
00014 #endif
00015
00016 #include "sound4.impact.cl_dyn.hpp"
00017
00018 // ------------------------------------------------------------
00051 std::string GetEnv(const std::string& name)
00052 {
00053     char *data=getenv(name.c_str());
00054     if (!data) return {};
00055     return std::string(data);
00056 }
00057
00058
00059 // Reads from stdin, write to stdout
00060 void ProcessLoop(sound4::impact::dyn::CInstance &instance)
00061 {
00062 bool stopthread=false;
00063
00064     #if defined(_WIN32) || defined(WIN32)
00065         // Windows: The stdin, stdout, and stderr streams always open in text mode by default
00066         _setmode(_fileno(stdin), _O_BINARY);
00067         _setmode(_fileno(stdout), _O_BINARY);
00068     #endif
00069
00071     while (!stopthread)
00072     {
00073         // Read a chunk from stdin
00074         if (std::fread(instance.GetBufferIn().data(), sizeof(float),
       sound4::impact::dyn::InputSampleSize, stdin) != sound4::impact::dyn::InputSampleSize) {
00075             std::cerr « "ProcessLoop: input stopped"« std::endl;
00076             stopthread=true;
00077             break;
00078         }
00079         if (stopthread) break;
00080         // Process it
00081         instance.ProcessAudio();
00082         // Write it to stdout
00083         if (fwrite(instance.GetBufferOut().data(), sizeof(float),
       sound4::impact::dyn::OutputSampleSize, stdout) != sound4::impact::dyn::OutputSampleSize) {
00084             std::cerr « "Failed to write audio output" « std::endl;
00085             stopthread=true;
00086         }
00087     }
00089 }
00090
00091 void Usage(const char *progname, sound4::impact::dyn::CDynLib &dynlib)
00092 {
00093     std::cerr « "Usage: " « progname « std::endl;;
00094     std::cerr « "  Reads audio from stdin and return processed audio in stdout" « std::endl;
00095     std::cerr « "Environment:"« std::endl;;
00096     std::cerr « " - S4STORE_PATH            : path to store config [default: (no storage)]"«
       std::endl;;
00097     std::cerr « " - S4LOGINKEY              : SOUND4 Login key (or S4USERID)"« std::endl;;
00098     std::cerr « " - RADIO_NAME              : SOUND4 Radio Name"« std::endl;;
00099     std::cerr « " - S4_AWS_ACCESS_KEY_ID    : SOUND4 Access Key ID"« std::endl;;
00100     std::cerr « " - S4_AWS_SECRET_ACCESS_KEY: SOUND4 Access Key Secret"« std::endl;;
00101     if (dynlib.IsOk()) {
00102         std::cerr « std::endl;
```

```
        std::cerr « "Example:"« std::endl;
        std::cerr « "ffmpeg -re -i file.wav -ar "«dynlib.GetSampleRate()«" -ac "«dynlib.GetChannelCount()«" -f f32le - | "«progname«" | ffplay -f f32le -ar "«dynlib.GetSampleRate()«" -ac "«dynlib.GetChannelCount()«" -i -" « std::endl;;
        #if defined(_WIN32) || defined(WIN32)
            std::cerr « "ffmpeg -f dshow -i audio=\"Microphone\" -ar "«dynlib.GetSampleRate()«" -ac "«dynlib.GetChannelCount()«" -f f32le - | "«progname«" | ffplay -f f32le -ar "«dynlib.GetSampleRate()«" -ac "«dynlib.GetChannelCount()«" -i -" « std::endl;;
        #else
            std::cerr « "ffmpeg -f pulse -i default -ar "«dynlib.GetSampleRate()«" -ac "«dynlib.GetChannelCount()«" -f f32le - | "«progname«" | ffplay -f f32le -ar "«dynlib.GetSampleRate()«" -ac "«dynlib.GetChannelCount()«" -i -" « std::endl;;
        #endif
    }
}

void MyLogger(sound4::impact::dyn::LogSeverity severity,const std::string& msg)
{
    std::cerr « msg « std::endl;
}

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

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

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

int main(int argc, const char **argv)
{
#if (IMPACT_HAS_WEBSERVER)
    int listen_port = 8080;
#endif // (IMPACT_HAS_WEBSERVER)
    auto store_path_str=GetEnv("S4STORE_PATH");

    // Get credentials from environment
    auto LoginKey=GetEnv("S4LOGINKEY");
    if (LoginKey.empty()) {
        LoginKey=GetEnv("S4USERID");
    }
    auto RadioName=GetEnv("RADIO_NAME");
    auto KeyId=GetEnv("S4_AWS_ACCESS_KEY_ID");
    auto AccessKey=GetEnv("S4_AWS_SECRET_ACCESS_KEY");

    const char *progname=argv[0];


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

    if (argc>1) {
        if ((strcasecmp(argv[1],"-h")==0) || (strcasecmp(argv[1],"--help")==0)) {
            Usage(progname,dynlib);
            return 0;
        }
    }
    if (LoginKey.empty() || RadioName.empty() || KeyId.empty() || AccessKey.empty()) {
        std::cerr « "Missing credentials in environment" « std::endl;
        Usage(progname,dynlib);
        return 1;
    }
    std::filesystem::path store_path;
    if (!store_path_str.empty()) {
        store_path = std::filesystem::path(store_path_str);
        if (!std::filesystem::exists(store_path) || !std::filesystem::is_directory(store_path)) {
            std::cerr « "Storage folder '"«store_path.c_str()«"' is not a valid directory" «
std::endl;
            Usage(progname,dynlib);
            return 1;
        }
    }

    sound4::impact::dyn::SetLogSeverity(dynlib, verbose5);
    sound4::impact::dyn::SetLoggerCallback(dynlib, MyLogger);

    sound4::impact::dyn::CInstance instance(dynlib);
    instance.SetParam("ADMIN_USER","admin");
    instance.SetParam("ADMIN_SECRET","admin");

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

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

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

```

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

    ProcessLoop(instance);

    std::cerr « "Stopping now" « std::endl;
    instance.Stop();

    std::cerr « "End" « std::endl;
    return 0;
}
```

11.5 sound4.impact.cl-pipe.cpp File Reference

C++ sample using only C interface.

```
#include <unistd.h>
#include <iostream>
#include <strings.h>
#include <sys/types.h>
#include <thread>
#include <experimental/filesystem>
#include <io.h>
#include <fcntl.h>
#include <stringapiset.h>
#include <sound4.impact.cl.h>
```

Namespaces

- std

Macros

- #define SAMPLE_RATE 48000
- #define CHUNK_SIZE 64
- #define CHANNEL_COUNT 2

Functions

- void ProcessLoop (struct impact_CInstance ∗pInstance)
- void Usage (const char ∗progname)
- void MyLogger (impact_LogSeverity severe[[maybe_unused]], const char ∗str)
- int main (int argc, char ∗const argv[ ])

11.5.1 Macro Definition Documentation

11.5.1.1 CHANNEL_COUNT

```
#define CHANNEL_COUNT 2
```

Definition at line 28 of file sound4.impact.cl-pipe.cpp.

11.5.1.2 CHUNK_SIZE

```
#define CHUNK_SIZE 64
```

Definition at line 27 of file sound4.impact.cl-pipe.cpp.

11.5.1.3 SAMPLE_RATE

```
#define SAMPLE_RATE 48000
```

Definition at line 26 of file sound4.impact.cl-pipe.cpp.

11.5.2 Function Documentation

11.5.2.1 main()

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

[Setup and sanity checks]

[Setup and sanity checks]

[Create the processing instance]

[Create the processing instance]

[Stopping the processing instance]

[Stopping the processing instance]

Definition at line 90 of file sound4.impact.cl-pipe.cpp.

11.5.2.2 MyLogger()

```
void MyLogger (
            impact_LogSeverity severe] [[maybe_unused],
            const char * str )
```

Definition at line 84 of file sound4.impact.cl-pipe.cpp.

11.5.2.3 ProcessLoop()

```
void ProcessLoop (
            struct impact_CInstance * pInstance )
```

[Processing Loop]

[Processing Loop]

Definition at line 31 of file sound4.impact.cl-pipe.cpp.

11.5.2.4 Usage()

```
void Usage (
            const char * progname )
```

Definition at line 64 of file sound4.impact.cl-pipe.cpp.

11.6 sound4.impact.cl-pipe.cpp

```
00001
00004 #include <unistd.h>
00005 #include <iostream>
00006 #include <strings.h>
00007 #include <sys/types.h>
00008 #include <thread>           // NOTE: needs pthread link
00009 // Choose good filesystem version depending on compiler.
00010 #if __has_include(<filesystem>)
00011     #include <filesystem>
00012 #else //if __cpp_lib_experimental_filesystem
00013     // We need the alias from std::experimental::filesystem to std::filesystem
00014     #include <experimental/filesystem>
00015     namespace std {
00016         namespace filesystem = experimental::filesystem;
00017     }
00018 #endif
00019 #if defined(_WIN32) || defined(WIN32)
00020     #include <io.h>
00021     #include <fcntl.h>
00022     #include <stringapiset.h>   // UTF-8 conversion
00023 #endif
00024 #include <sound4.impact.cl.h>
00025
00026 #define SAMPLE_RATE       48000
00027 #define CHUNK_SIZE        64
00028 #define CHANNEL_COUNT     2
00029
00030 // Reads from stdin, write to stdout
00031 void ProcessLoop(struct impact_CInstance *pInstance)
00032 {
00034 bool stopthread=false;
```

```
alignas(64) float audio_in[CHUNK_SIZE*CHANNEL_COUNT];
alignas(64) float audio_out[CHUNK_SIZE*CHANNEL_COUNT];

    #if defined(_WIN32) || defined(WIN32)
        // Windows: The stdin, stdout, and stderr streams always open in text mode by default
        _setmode(_fileno(stdin), _O_BINARY);
        _setmode(_fileno(stdout), _O_BINARY);
    #endif

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

void Usage(const char *progname)
{
    std::cerr « "Usage: " « progname « std::endl;;
    std::cerr « "  Reads audio from stdin and return processed audio in stdout" « std::endl;
    std::cerr « "Environment:"« std::endl;;
    std::cerr « " - S4STORE_PATH            : path to store config [default: ./state]"« std::endl;;
    std::cerr « " - S4LOGINKEY              : SOUND4 Login key (or S4USERID)"« std::endl;;
    std::cerr « " - RADIO_NAME              : SOUND4 Radio Name"« std::endl;;
    std::cerr « " - S4_AWS_ACCESS_KEY_ID    : SOUND4 Access Key ID"« std::endl;;
    std::cerr « " - S4_AWS_SECRET_ACCESS_KEY: SOUND4 Access Key Secret"« std::endl;;
    std::cerr « std::endl;
    std::cerr « "Example:"« std::endl;
    std::cerr « "ffmpeg -re -i file.wav -ar "«SAMPLE_RATE«" -ac "«CHANNEL_COUNT«" -f f32le - | "«progname«" | ffplay -f f32le -ar "«SAMPLE_RATE«" -ac "«CHANNEL_COUNT«" -i -" « std::endl;;
    #if defined(_WIN32) || defined(WIN32)
        std::cerr « "ffmpeg -f dshow -i audio=\"Microphone\" -ar "«SAMPLE_RATE«" -ac "«CHANNEL_COUNT«" -f f32le - | "«progname«" | ffplay -f f32le -ar "«SAMPLE_RATE«" -ac "«CHANNEL_COUNT«" -i -" « std::endl;;
    #else
        std::cerr « "ffmpeg -f pulse -i default -ar "«SAMPLE_RATE«" -ac "«CHANNEL_COUNT«" -f f32le - | "«progname«" | ffplay -f f32le -ar "«SAMPLE_RATE«" -ac "«CHANNEL_COUNT«" -i -" « std::endl;;
    #endif
}

void MyLogger(impact_LogSeverity severe [[maybe_unused]], const char *str)
{
    std::cerr « str « std::endl;
}


int main(int argc, char *const argv[])
{
    struct impact_CInstance *pInstance = NULL;
    const char *store_path_str=getenv("S4STORE_PATH");
    // Get credentials from environment
    char *pLoginKey=getenv("S4LOGINKEY");
    if (!pLoginKey) {
        pLoginKey=getenv("S4USERID");
    }
    char *pRadioName=getenv("RADIO_NAME");
    char *pKeyId=getenv("S4_AWS_ACCESS_KEY_ID");
    char *pAccessKey=getenv("S4_AWS_SECRET_ACCESS_KEY");

    const char *progname=argv[0];

    if (argc>1) {
        if ((strcasecmp(argv[1],"-h")==0) || (strcasecmp(argv[1],"--help")==0)) {
            Usage(progname);
            return 0;
        }
    }
    if (!pLoginKey || !pRadioName || !pKeyId || !pAccessKey) {
        std::cerr « "Missing credentials in environment" « std::endl;
        Usage(progname);
        return 1;
    }
    std::filesystem::path fs_store_path;
```

```
    if (!store_path_str || store_path_str[0]==0) {
        fs_store_path=std::filesystem::current_path() / "state";
    } else {
        fs_store_path = std::filesystem::path(store_path_str);
    }
    if (!std::filesystem::exists(fs_store_path) || !std::filesystem::is_directory(fs_store_path)) {
        std::cerr « "Storage folder '"«fs_store_path.c_str()«"' is not a valid directory" « std::endl;
        Usage(progname);
        return 1;
    }
    std::string store_path;
    #ifdef _WIN32
    {
        // Convert path to UTF-8
        std::wstring wstr(fs_store_path.wstring());
        if (wstr.empty()) return {};
        int size = WideCharToMultiByte(CP_UTF8, 0, &wstr[0], (int)wstr.size(), NULL, 0, NULL, NULL);
        std::string str(size, 0);
        WideCharToMultiByte(CP_UTF8, 0, &wstr[0], (int)wstr.size(), &str[0], size, NULL, NULL);
        store_path=str;
    }
    #else // !_WIN32
        store_path=fs_store_path;
    #endif // !_WIN32

    // Optional: Use our own logging, instead of default color output to stderr.
    //impact_SetLoggerCallback(&MyLogger);

    //impact_SetLogSeverity(impact_LogSeverity::info);
    impact_SetLogSeverity(impact_LogSeverity::verbose5);

    std::cerr « "Process version: " « impact_GetVersion() « std::endl;
    // Do some sanity checks
    if (impact_GetSampleRate()!=SAMPLE_RATE) {
      std::cerr « "Bad library sampling rate, expecting '"«SAMPLE_RATE«"' but has "«
impact_GetSampleRate() « std::endl;
      return 1;
    }
    if (impact_GetChunkSizeInFrames()!=CHUNK_SIZE) {
      std::cerr « "Bad library frame size, expecting '"«CHUNK_SIZE«"' but has "«
impact_GetChunkSizeInFrames() « std::endl;
      return 1;
    }
    if (impact_GetChannelCount()!=CHANNEL_COUNT) {
      std::cerr « "Bad library channel count, expecting '"«CHANNEL_COUNT«"' but has "«
impact_GetChannelCount() « std::endl;
      return 1;
    }

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

    // Create the processing instance
    pInstance=impact_InitProcess2(pLoginKey,pRadioName,pKeyId,pAccessKey,store_path.c_str(), params);
    impact_FreeParameters(params);
    if (!pInstance) {
      std::cerr « "Failed to create the process instance"« std::endl;
      return 1;
    }

    // Start the update thread, without JSON server (low priority)
    std::thread updater([pInstance](){
       // will returned when stopped or destroyed
       std::cerr « "impact_StartUpdateThread (no json port)" « std::endl;
       impact_StartUpdateThread(pInstance, 0);
       std::cerr « "impact_StartUpdateThread END" « std::endl;
     });

    // This is not mandatory, but better be sure we are ready right now
    std::cerr « "Waiting update thread" « std::endl;
    if (impact_WaitUpdateThreadReady(pInstance, 5000)<0) {
        std::cerr « "Timeout to create the process instance"« std::endl;
    }
    std::cerr « "Waiting update thread DONE" « std::endl;

#if (IMPACT_HAS_WEBSERVER)
    // Starts a web server, listening on port http://localhost:8080, https://localhost:8443
    std::cerr « "Starting web server" « std::endl;
    int listenport= 8080;
    int listenport_secure= 8443;
    uint64_t webserver= SOUND4_INVALID_WEBSERVER_ID;
    if (listenport || listenport_secure) {
        webserver = impact_Webserver(listenport,listenport_secure,pInstance);
```

```
        if (webserver==SOUND4_INVALID_WEBSERVER_ID) {
            std::cerr « "Starting web server FAILED" « std::endl;
        } else {
            int web_status=impact_Webserver_Status(webserver);
            if (listenport) {
                if (web_status & SOUND4_WEBSERVER_HTTP_OK) {
                    std::cerr « "Web server listening on port " « listenport;
                } else {
                    std::cerr « "Web server failed to listen on port " « listenport;
                }
            }
            if (listenport_secure) {
                if (web_status & SOUND4_WEBSERVER_HTTPS_OK) {
                    std::cerr « "Web server listening on secure port " « listenport_secure;
                } else {
                    std::cerr « "Web server failed to listen on secure port " « listenport_secure;
                }
            }
            std::cerr « "Starting web server DONE" « std::endl;
        }
    }
#endif // (IMPACT_HAS_WEBSERVER)

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

    std::cerr « "Estimated delay " « ( impact_GetEstimatedDelay(pInstance)*1000.0/SAMPLE_RATE ) « " ms" « std::endl;

    ProcessLoop(pInstance);

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

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

    updater.join();

    impact_ExitProcess(pInstance);

    return 0;
}
```

11.7 sound4.impact.cl.h File Reference

C interface for library.

```
#include <stdint.h>
```

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 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 *AWS_Access_ID, const char *AWS_Access_Secret, const char *save_path)

- struct impact_CInstance * impact_InitProcess2 (const char *LoginKey, const char *RadioName, const char *AWS_Access_ID, const char *AWS_Access_Secret, const char *save_path, const struct impact_CParameters *parameters)
- struct impact_CInstance * impact_InitProcess3 (const char *LoginKey, const char *RadioName, const char *AWS_Access_ID, const char *AWS_Access_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)
- const char ** impact_GetMetadataInfos (struct impact_CInstance *instance)
- void impact_FreeMetadataInfos (struct impact_CInstance *instance, const char **infos)
- void impact_StartUpdateThread (struct impact_CInstance *instance, unsigned int port)
- void impact_StopUpdateThread (struct impact_CInstance *instance)
- int impact_WaitUpdateThreadReady (struct impact_CInstance *instance, int milliseconds)
- void impact_ProcessAudio (struct impact_CInstance *instance, const float *input, float *output)
- void impact_ProcessAudio_Planar (struct impact_CInstance *instance, float const *const *input, float *const *output)
- float * impact_GetBufferIn (struct impact_CInstance *instance)
- float * impact_GetBufferOut (struct impact_CInstance *instance)
- unsigned int impact_GetEstimatedDelay (struct impact_CInstance *instance)
- const char * impact_GetFormatName (const enum impact_SampleFormat fmt)
- enum impact_SampleFormat impact_GetFormatFromName (const char *name)
- unsigned int impact_GetBytesFromFormat (const enum impact_SampleFormat fmt)
- unsigned int impact_GetMaxPacketFrame (struct impact_CInstance *instance)
- unsigned int impact_AddAudio (struct impact_CInstance *instance, const uint8_t *payload, unsigned int nFrame, enum impact_SampleFormat fmt)
- unsigned int impact_AddPadAudio (struct impact_CInstance *instance)
- unsigned int impact_GetOutputCount (struct impact_CInstance *instance)
- unsigned int impact_GetAudio (struct impact_CInstance *instance, uint8_t *payload, unsigned int max_nFrame, enum impact_SampleFormat fmt)
- void impact_AudioConvertFrom (const uint8_t *payload, float *output, size_t nSpl, enum impact_SampleFormat fmt)
- void impact_AudioConvertTo (const float *input, uint8_t *payload, size_t nSpl, enum impact_SampleFormat fmt)
- void impact_StereoToMono (const float *input, float *output, size_t nFrame)
- void impact_MonoToStereo (const float *input, float *output, size_t nFrame)
- void impact_StereoToMono_Planar (const float *inputL, const float *inputR, float *output, size_t nFrame)
- void impact_MonoToStereo_Planar (const float *input, float *outputL, float *outputR, size_t nFrame)
- void impact_AudioMonoFromLiveStereo (const uint8_t *payload, float *output)
- void impact_AudioMonoToLiveStereo (const float *input, uint8_t *payload)
- struct impact_CClientInstance * impact_NewClient (struct impact_CInstance *instance)
- void impact_DeleteClient (struct impact_CClientInstance *client)
- const char * impact_ProcessJson (struct impact_CClientInstance *client, const char *json_str, int *need_save)
- void impact_FreeJsonAnswer (const char *json_str)
- int impact_SaveState (struct impact_CInstance *instance)
- uint64_t impact_Webserver_tcp (unsigned int listenport, unsigned int listenport_secure, const char *socket_ip, unsigned int socket_port)

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

11.7.1 Macro Definition Documentation

11.7.1.1 IMPACT_DECL

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

Definition at line 14 of file sound4.impact.cl.h.

11.7.1.2 IMPACT_HAS_CLOUDBUS

```
#define IMPACT_HAS_CLOUDBUS 0
```

Definition at line 19 of file sound4.impact.cl.h.

11.7.1.3 IMPACT_HAS_WEBSERVER

```
#define IMPACT_HAS_WEBSERVER 1
```

Definition at line 18 of file sound4.impact.cl.h.

11.8 sound4.impact.cl.h

```
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 #endif
00017
00018 #define IMPACT_HAS_WEBSERVER 1
00019 #define IMPACT_HAS_CLOUDBUS 0
00020
00021 #ifdef __cplusplus
00022 extern "C" {
00023 #endif // __cplusplus
00024
```

```
// ------------------------------------------------------------
// ------------------------------------------------------------
// ------------------------------------------------------------

// ------------------------------------------------------------
#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

IMPACT_DECL const char *impact_GetVersion();

IMPACT_DECL unsigned int impact_GetChunkSizeInFrames();

IMPACT_DECL unsigned int* impact_GetPossibleChunkSizeInFrames();

IMPACT_DECL unsigned int impact_GetChannelCount();

IMPACT_DECL unsigned int impact_GetAudioInputCount();

IMPACT_DECL unsigned int impact_GetAudioOutputCount();

IMPACT_DECL unsigned int impact_GetSampleRate();

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

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

IMPACT_DECL void impact_SetLoggerCallback( impact_loggerfn logger );

IMPACT_DECL void impact_SetLogSeverity( enum impact_LogSeverity severity );

// ------------------------------------------------------------
// ------------------------------------------------------------
struct impact_CInstance;

struct impact_CParameters;

IMPACT_DECL struct impact_CParameters *impact_NewParameters();

IMPACT_DECL void impact_FreeParameters(struct impact_CParameters *params);

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

IMPACT_DECL const char *impact_GetParameter(struct impact_CParameters *params, const char *name);

IMPACT_DECL void impact_FreeParameterValue(const char *value);

IMPACT_DECL struct impact_CInstance *impact_InitProcess(
        const char *LoginKey,
        const char *RadioName,
        const char *AWS_Access_ID,
        const char *AWS_Access_Secret,
        const char *save_path
    );

IMPACT_DECL struct impact_CInstance *impact_InitProcess2(
        const char *LoginKey,
        const char *RadioName,
        const char *AWS_Access_ID,
        const char *AWS_Access_Secret,
        const char *save_path,
        const struct impact_CParameters *parameters
    );

IMPACT_DECL struct impact_CInstance *impact_InitProcess3(
        const char *LoginKey,
        const char *RadioName,
        const char *AWS_Access_ID,
        const char *AWS_Access_Secret,
        const char *save_path,
        const struct impact_CParameters *parameters,
        unsigned int frames_per_chunk
    );
```

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

IMPACT_DECL int impact_TerminateProcess(struct impact_CInstance *instance);

IMPACT_DECL void impact_ExitProcess(struct impact_CInstance *instance);

#if IMPACT_HAS_CLOUDBUS
// ------------------------------------------------------------
struct impact_CBus;

IMPACT_DECL struct impact_CBus* impact_NewBus();

IMPACT_DECL void impact_FreeBus(struct impact_CBus *bus);

IMPACT_DECL void impact_SetInstanceBus(struct impact_CParameters *params, struct impact_CBus *bus);


#endif // IMPACT_HAS_CLOUDBUS

// ------------------------------------------------------------
// ------------------------------------------------------------

// OS dependent filesystem path char type
#ifdef _WIN32
    typedef wchar_t fs_char;
#else // !_WIN32
    typedef char fs_char;
#endif // !_WIN32

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

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

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

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

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

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

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

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

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

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

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

// ------------------------------------------------------------
// ------------------------------------------------------------

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

IMPACT_DECL const char** impact_GetMetadataInfos(struct impact_CInstance *instance);

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

// ------------------------------------------------------------
// ------------------------------------------------------------

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

IMPACT_DECL void impact_StopUpdateThread(struct impact_CInstance *instance);

IMPACT_DECL int impact_WaitUpdateThreadReady(struct impact_CInstance *instance, int milliseconds);

// ------------------------------------------------------------
// ------------------------------------------------------------
```

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

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

IMPACT_DECL float *impact_GetBufferIn(struct impact_CInstance *instance);

IMPACT_DECL float *impact_GetBufferOut(struct impact_CInstance *instance);

IMPACT_DECL unsigned int impact_GetEstimatedDelay(struct impact_CInstance *instance);

// ------------------------------------------------------------
// ------------------------------------------------------------
    #ifdef _MSC_VER
        #define __BYTE_ORDER__  REG_DWORD
        #define __ORDER_LITTLE_ENDIAN__ REG_DWORD_LITTLE_ENDIAN
        #define __ORDER_BIG_ENDIAN__    REG_DWORD_BIG_ENDIAN
    #endif

enum impact_SampleFormat {
    INVALID_FORMAT=0,
    S16_LE,
    S16_BE,
    S24_LE,
    S24_BE,
    S32_LE,
    S32_BE,
    F32_LE,
    F32_BE,

    // Native definitions, only links to the physical ones
    #if __BYTE_ORDER__ == __ORDER_LITTLE_ENDIAN__
        S16_NATIVE=S16_LE,
        S24_NATIVE=S24_LE,
        S32_NATIVE=S32_LE,
        F32_NATIVE=F32_LE,
    #else
        S16_NATIVE=S16_BE,
        S24_NATIVE=S24_BE,
        S32_NATIVE=S32_BE,
        F32_NATIVE=F32_BE,
    #endif
};

IMPACT_DECL const char *impact_GetFormatName(const enum impact_SampleFormat fmt);

IMPACT_DECL enum impact_SampleFormat impact_GetFormatFromName(const char *name);

IMPACT_DECL unsigned int impact_GetBytesFromFormat(const enum impact_SampleFormat fmt);


IMPACT_DECL unsigned int impact_GetMaxPacketFrame(struct impact_CInstance *instance);

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

IMPACT_DECL unsigned int impact_AddPadAudio(struct impact_CInstance *instance);

IMPACT_DECL unsigned int impact_GetOutputCount(struct impact_CInstance *instance);

IMPACT_DECL unsigned int impact_GetAudio(struct impact_CInstance *instance, uint8_t *payload, unsigned int max_nFrame, enum impact_SampleFormat fmt );

// ------------------------------------------------------------
// ------------------------------------------------------------
IMPACT_DECL void impact_AudioConvertFrom(const uint8_t *payload, float *output, size_t nSpl, enum impact_SampleFormat fmt);

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

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

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

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

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

IMPACT_DECL void impact_AudioMonoFromLiveStereo(const uint8_t *payload, float *output);

IMPACT_DECL void impact_AudioMonoToLiveStereo(const float *input, uint8_t *payload);

```

```
// ------------------------------------------------------------
// ------------------------------------------------------------
struct impact_CClientInstance;

IMPACT_DECL struct impact_CClientInstance *impact_NewClient(struct impact_CInstance *instance);

IMPACT_DECL void impact_DeleteClient(struct impact_CClientInstance *client);

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

IMPACT_DECL void impact_FreeJsonAnswer(const char *json_str);

IMPACT_DECL int impact_SaveState(struct impact_CInstance *instance);

#if IMPACT_HAS_WEBSERVER
// ------------------------------------------------------------
// ------------------------------------------------------------
IMPACT_DECL uint64_t impact_Webserver_tcp(unsigned int listenport, unsigned int listenport_secure,
 const char *socket_ip, unsigned int socket_port);

// ------------------------------------------------------------
IMPACT_DECL uint64_t impact_Webserver_tcp2(unsigned int listenport, unsigned int listenport_secure,
 const char *socket_ip, unsigned int socket_port, const struct impact_CParameters *parameters);

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

IMPACT_DECL int impact_Webserver_Stop(uint64_t id, int timeout_ms);

IMPACT_DECL int impact_Webserver_Status(uint64_t id);

#define SOUND4_INVALID_WEBSERVER_ID            ((uint64_t)-1)

#define SOUND4_WEBSERVER_HTTP_OK               (1«0)
#define SOUND4_WEBSERVER_HTTPS_OK              (1«1)

#endif // IMPACT_HAS_WEBSERVER

#ifdef __cplusplus
} // extern "C"
#endif // __cplusplus
```

11.9 sound4.impact.cl.hpp File Reference

C++ wrapper for library.

```
#include <string>
#include <exception>
#include <thread>
#include <functional>
#include <memory>
#include <array>
#include <assert.h>
#include <stdexcept>
#include <filesystem>
#include <string.h>
#include "sound4.impact.cl.h"
#include <stringapiset.h>
#include <wchar.h>
```

Classes

- struct sound4::impact::helper::SampleFormat< T >

  Helper for Sample format types.
- struct sound4::impact::helper::SampleFormat< int16_t >
- struct sound4::impact::helper::SampleFormat< int32_t >

- struct sound4::impact::helper::SampleFormat< float >
- class sound4::impact::CPresetLoader
- class sound4::impact::CInstance
- class sound4::impact::CInstance::CClient

Namespaces

- sound4
- sound4::impact
- sound4::impact::helper

Macros

- #define fs_strdup wcsdup

  helper for OS dependant file name copy

Typedefs

- using sound4::impact::SampleFormat = impact_SampleFormat
- using sound4::impact::LogSeverity = impact_LogSeverity
- using sound4::impact::log_cb_t = std::function< void(LogSeverity, const std::string &)>
- using sound4::impact::PresetChange_Kind = impact_PresetChange_Kind

Variables

- const size_t sound4::impact::ChunkFrames =IMPACT_AUDIOFRAME_COUNT
- const size_t sound4::impact::InputChannels =IMPACT_CHANNEL_COUNT∗IMPACT_AUDIO_INPUT_COUNT
- const size_t sound4::impact::OutputChannels =IMPACT_CHANNEL_COUNT∗IMPACT_AUDIO_OUTPUT_COUNT
- const size_t sound4::impact::InputSampleSize =ChunkFrames∗InputChannels
- const size_t sound4::impact::OutputSampleSize =ChunkFrames∗OutputChannels
- const size_t sound4::impact::SampleRate =IMPACT_SAMPLE_RATE

11.9.1 Macro Definition Documentation

11.9.1.1 fs_strdup

```
#define fs_strdup wcsdup
```

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

11.10 sound4.impact.cl.hpp

```
00001
00004 #pragma once
00005
00006 #include <string>
00007 #include <exception>
00008 #include <thread>
00009 #include <functional>
00010 #include <memory>
00011 #include <array>
00012 #include <assert.h>
00013 #include <stdexcept>
00014 #include <filesystem>
00015 #include <string.h>
00016
00017 #include "sound4.impact.cl.h"
00018
00019
00020 // Filesystem path char type
00021 #ifdef _WIN32
00022     #include <stringapiset.h>
00023     #include <wchar.h>
00025     #define fs_strdup wcsdup
00026 #else // !_WIN32
00028     #define fs_strdup strdup
00029 #endif // !_WIN32
00030
00031 // bridge C callbacks to CPresetLoader
00032 extern "C" {
00033     static char *impact_custom_reader(const fs_char *filename, void* userdata);
00034     static void impact_custom_reader_free(char *content, void* userdata);
00035     static int impact_custom_writer(const fs_char *filename, const char *content, void* userdata);
00036     static int impact_custom_exists(const fs_char *filename, void* userdata);
00037     static fs_char** impact_custom_getall(void* userdata);
00038     static void impact_custom_getall_free(fs_char** all, void* userdata);
00039     static int impact_custom_remove(const fs_char *filename, void* userdata);
00040     static int impact_custom_rename(const fs_char *from, const fs_char *to, void* userdata);
00041 };
00042
00046 namespace sound4 {
00050 namespace impact {
00051     static constexpr const char *process_name = "SOUND4 IMPACT.CL";
00052     static constexpr const char *process_shortname = "impact";
00053     using SampleFormat = impact_SampleFormat;
00057     const size_t ChunkFrames=IMPACT_AUDIOFRAME_COUNT;
00061     const size_t InputChannels=IMPACT_CHANNEL_COUNT*IMPACT_AUDIO_INPUT_COUNT;
00065     const size_t OutputChannels=IMPACT_CHANNEL_COUNT*IMPACT_AUDIO_OUTPUT_COUNT;
00069     const size_t InputSampleSize=ChunkFrames*InputChannels;
00073     const size_t OutputSampleSize=ChunkFrames*OutputChannels;
00077     const size_t SampleRate=IMPACT_SAMPLE_RATE;
00078
00082     namespace helper {
00083         #ifdef _WIN32
00089         static inline std::string WStringToUTF8(const std::wstring& wstr) {
00090             if (wstr.empty()) return {};
00091             int size = WideCharToMultiByte(CP_UTF8, 0, &wstr[0], (int)wstr.size(), NULL, 0, NULL,
      NULL);
00092             std::string str(size, 0);
00093             WideCharToMultiByte(CP_UTF8, 0, &wstr[0], (int)wstr.size(), &str[0], size, NULL, NULL);
00094             return str;
00095         }
00101         static inline std::wstring UTF8ToWString(const std::string& str) {
00102             if (str.empty()) return std::wstring();
00103             int size = MultiByteToWideChar(CP_UTF8, 0, &str[0], (int)str.size(), NULL, 0);
00104             std::wstring wstr(size, 0);
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
```

```

        static inline void AudioConvertFrom(const uint8_t *payload, float *output, size_t nSpl, impact_SampleFormat fmt) {
            impact_AudioConvertFrom(payload, output, nSpl, fmt);
        }
        static inline void AudioConvertTo(const float *input, uint8_t *payload, size_t nSpl, impact_SampleFormat fmt) {
            impact_AudioConvertTo(input, payload, nSpl, fmt);
        }
        static inline void StereoToMono(const float *input, float *output, size_t nSpl) {
            impact_StereoToMono(input, output, nSpl);
        }
        static inline void MonoToStereo(const float *input, float *output, size_t nSpl) {
            impact_MonoToStereo(input, output, nSpl);
        }
        static inline void StereoToMono_Planar(const float *inputL, const float *inputR, float *output, size_t nSpl) {
            impact_StereoToMono_Planar(inputL, inputR, output, nSpl);
        }
        static inline void MonoToStereo_Planar(const float *input, float *outputL, float *outputR, size_t nSpl) {
            impact_MonoToStereo_Planar(input, outputL, outputR, nSpl);
        }
        static inline void AudioMonoFromLiveStereo(const uint8_t *payload, float *output) {
            impact_AudioMonoFromLiveStereo(payload, output);
        }
        static inline void AudioMonoToLiveStereo(const float *input, uint8_t *payload) {
            impact_AudioMonoToLiveStereo(input, payload);
        }

    }; // namespace helper

    static inline std::string GetVersion()
        {
            return std::string(impact_GetVersion());
        }

    static inline unsigned int GetChunkSizeInFrames()
        {
            return impact_GetChunkSizeInFrames();
        }

    static inline std::vector<unsigned int> GetPossibleChunkSizeInFrames()
        {
            std::vector<unsigned int> list;
            for (unsigned int* src=impact_GetPossibleChunkSizeInFrames();*src;src++) {
                list.push_back(*src);
            }
            return list;
        }

    static inline unsigned int GetChannelCount()
        {
            return impact_GetChannelCount();
        }

    static inline unsigned int GetAudioInputCount()
        {
            return impact_GetAudioInputCount();
        }

    static inline unsigned int GetAudioOutputCount()
        {
            return impact_GetAudioOutputCount();
        }

    static inline unsigned int GetSampleRate()
        {
            return impact_GetSampleRate();
        }

    static inline void SanityCheck(bool a_bCheckFrames = true)
    {
        // Do some sanity checks
        if (impact_GetSampleRate()!=IMPACT_SAMPLE_RATE) {
            throw std::runtime_error("Bad library sampling rate");
        }
        if (a_bCheckFrames && impact_GetChunkSizeInFrames()!=ChunkFrames) {
            throw std::runtime_error("Bad library frame size");
        }
        if (impact_GetChannelCount()!=IMPACT_CHANNEL_COUNT) {
            throw std::runtime_error("Bad library channel count");
        }
        if (impact_GetAudioInputCount()!=IMPACT_AUDIO_INPUT_COUNT) {
            throw std::runtime_error("Bad library input count");
        }
        if (impact_GetAudioOutputCount()!=IMPACT_AUDIO_OUTPUT_COUNT) {
```

```
            throw std::runtime_error("Bad library output count");
        }
    }

    static inline std::string GetFormatName(const SampleFormat fmt)
        {
            return std::string(impact_GetFormatName(fmt));
        }

    static inline SampleFormat GetFormatFromName(const std::string& name)
        {
            return impact_GetFormatFromName(name.c_str());
        }

    static inline unsigned int GetBytesFromFormat(const SampleFormat fmt)
        {
            return impact_GetBytesFromFormat(fmt);
        }

    using LogSeverity = impact_LogSeverity;
    using log_cb_t = std::function<void(LogSeverity,const std::string&)>;

    static log_cb_t _log_cb;
    extern "C" {
        static inline void _log_cb_c(impact_LogSeverity severity, const char *c_msg) {
            _log_cb(severity, std::string(c_msg));
        }
    }
    static inline void SetLogSeverity(LogSeverity severity)
        {
            impact_SetLogSeverity(severity);
        }
    static inline void SetLoggerCallback(log_cb_t cb)
        {
            _log_cb=cb;
            impact_SetLoggerCallback(&_log_cb_c);
        }

#if IMPACT_HAS_CLOUDBUS
    class CBus {
        public:
            CBus()
                : m_bus(impact_NewBus()) {}
            ~CBus()
                {
                    impact_FreeBus(m_bus);
                    m_bus = nullptr;
                }
            impact_CBus *Get() const { return m_bus; }
        private:
            impact_CBus *m_bus;
    };
#endif // IMPACT_HAS_CLOUDBUS

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

        virtual bool IsReadOnly() = 0;
        virtual bool Exists(const std::filesystem::path &name) = 0;
        virtual bool Remove(const std::filesystem::path &name) = 0;
        virtual bool Rename(const std::filesystem::path &from, const std::filesystem::path &to) = 0;
        virtual std::vector<std::filesystem::path> GetAll() = 0;
        virtual std::string Read(const std::filesystem::path &filename) = 0;
        virtual bool Write(const std::filesystem::path &filename, const std::string &content) =0;
    };
    using PresetChange_Kind = impact_PresetChange_Kind;

    class CInstance {
    public:
        CInstance()
            {
                params = impact_NewParameters();
                if (!params) throw std::bad_alloc();
            }
        // non-copyable
        CInstance( const CInstance& ) = delete;
        CInstance& operator=( const CInstance& ) = delete;
        ~CInstance()
            {
                Stop();
                if (params) impact_FreeParameters(params);
            }
       bool IsOk() const { return instance!=nullptr; }
        void SetParam(const std::string& name, const std::string& value)
            {
```

```
                assert(params);
                impact_SetParameter(params, name.c_str(), value.c_str());
            }
        std::string GetParam(const std::string& name)
            {
                assert(params);
                auto c_value=impact_GetParameter(params, name.c_str());
                std::string ret(c_value);
                impact_FreeParameterValue(c_value);
                return ret;
            }

        void SetPresetManager(CPresetLoader *preset_manager)
            {
                if (!preset_manager) return;
                assert(params);
                impact_SetPresetManager(params,
                    impact_custom_reader,
                    impact_custom_reader_free,
                    impact_custom_writer,
                    impact_custom_exists,
                    impact_custom_getall,
                    impact_custom_getall_free,
                    impact_custom_remove,
                    impact_custom_rename,
                    preset_manager->IsReadOnly(),
                    preset_manager
                );
            }

#if IMPACT_HAS_CLOUDBUS
            void SetBus(const CBus& bus) {
                assert(params);
                impact_SetInstanceBus(params,bus.Get());
            }
#endif // IMPACT_HAS_CLOUDBUS

        bool Create(const std::string& LoginKey, const std::string& RadioName, const std::string& AWS_Access_ID, const std::string& AWS_Access_Secret, const std::filesystem::path& save_path, int json_port = 0, unsigned int frames_per_chunk=64)
            {
                assert(!instance);
                std::string l_save_path;
                if (!save_path.empty()) {
                    l_save_path = save_path.u8string();
                }
                instance = impact_InitProcess3(LoginKey.c_str(), RadioName.c_str(), AWS_Access_ID.c_str(), AWS_Access_Secret.c_str(), l_save_path.c_str(), params, frames_per_chunk);
                if (!instance) return false;
                if (!init_metadata.empty()) {
                    for (auto&& [key,value]: init_metadata) {
                        impact_SetMetadata(instance, key.c_str(), value.c_str());
                    }
                    init_metadata.clear();
                }
                update_thread = std::thread([this, json_port](){
                    OnUpdateThreadStart();
                    impact_StartUpdateThread(instance, json_port);
                    OnUpdateThreadStop();
                });
                int timeout=5000;
                #ifdef DEBUG
                    timeout=60*1000; // For debugger pauses
                #endif
                if (impact_WaitUpdateThreadReady(instance, timeout)<0) {
                    return false;
                }
                return true;
            }

        unsigned int GetChunkFrames() {
            if (instance) {
                return impact_GetProcessChunkFrames(instance);
            }
            return 0;
        }

        void PresetManager_InformChange(const std::filesystem::path& relative_path, PresetChange_Kind change_kind)
            {
                impact_PresetManager_InformChange(instance, relative_path.c_str(), (impact_PresetChange_Kind) change_kind);
            }

        void SetMetadata(const std::string& key, const char* value)
            {
                if (instance) {
```

```
                    impact_SetMetadata(instance, key.c_str(), value);
                } else {
                    init_metadata.push_back( {key, value} );
                }
            }

        std::vector< std::tuple<std::string,std::string> > GetMetadataInfos()
            {
                std::vector< std::tuple<std::string,std::string> > values;
                const char** c_values = impact_GetMetadataInfos(instance);
                if (c_values) {
                    for (const char** c_value=c_values; *c_value; ) {
                        std::string key(*c_value);
                        c_value++;
                        if (*c_value) {
                            std::string descr(*c_value);
                            c_value++;
                            values.push_back( {key,descr} );
                        }
                    }
                    impact_FreeMetadataInfos(instance, c_values);
                }
                return values;
            }

        // Can be overloaded, called at start of UpdateThread in the thread
        virtual void OnUpdateThreadStart() {}
        // Can be overloaded to call at end of UpdateThread in the thread
        virtual void OnUpdateThreadStop() {}

#if IMPACT_HAS_WEBSERVER
        bool StartWebServer(int http_port, int https_port=0)
            {
                assert(instance);
                assert(webserver==SOUND4_INVALID_WEBSERVER_ID);
                webserver = impact_Webserver(http_port,https_port,instance);
                if (webserver==SOUND4_INVALID_WEBSERVER_ID) {
                    return false;
                }
                int web_status=impact_Webserver_Status(webserver);
                if (http_port && (web_status & SOUND4_WEBSERVER_HTTP_OK) == 0) {
                    return false;
                }
                if (https_port && (web_status & SOUND4_WEBSERVER_HTTPS_OK) == 0) {
                    return false;
                }
                return true;
            }
        void StopWebServer()
            {
                if (webserver != SOUND4_INVALID_WEBSERVER_ID) {
                    impact_Webserver_Stop(webserver, 1000);
                    webserver = SOUND4_INVALID_WEBSERVER_ID;
                }
            }
#endif // IMPACT_HAS_WEBSERVER

        int Terminate()
            {
                if (!instance) return 0;
                return impact_TerminateProcess(instance);
            }

        void Stop()
            {
                if (!instance) return;
#if IMPACT_HAS_WEBSERVER
                StopWebServer();
#endif // IMPACT_HAS_WEBSERVER
                if (update_thread.joinable()) {
                    impact_StopUpdateThread(instance); // this will exit the update thread
                    update_thread.join();
                }
                impact_ExitProcess(instance);
                instance=nullptr;
            }

        unsigned int GetEstimatedDelay()
            {
                assert(instance);
                return impact_GetEstimatedDelay(instance);
            }

        std::array<float,InputSampleSize>& GetBufferIn()
            {
                assert(instance);
                float *buf=impact_GetBufferIn(instance);
```

```
                return reinterpret_cast< std::array<float,InputSampleSize>& >(*buf);

            }
        std::array<float,OutputSampleSize>& GetBufferOut()
            {
                assert(instance);
                float *buf= impact_GetBufferOut(instance);
                return reinterpret_cast< std::array<float,OutputSampleSize>& >(*buf);
            }
        void ProcessAudio()
            {
                assert(instance);
                impact_ProcessAudio(instance, impact_GetBufferIn(instance),
    impact_GetBufferOut(instance));
            }
        // void ProcessAudio_Planar(const std::array< const std::array<const float,ChunkFrames>,
    InputChannels>& input, std::array< std::array<float,ChunkFrames>, OutputChannels>& output)
        void ProcessAudio_Planar(float const * const *input, float * const *output)
            {
                assert(instance);
                impact_ProcessAudio_Planar(instance, input, output);
            }

            template<typename T>
        std::vector<T> ProcessAnyAudio(const std::vector<T> input)
            {
                assert(instance);
                std::vector<T> output;
                unsigned int out_offset=0;
                unsigned int in_offset=0;
                unsigned int todo = input.size();
                while (todo>0) {
                    unsigned int left = AddAudio(&input[in_offset], todo);
                    unsigned int out_avail = impact_GetOutputCount(instance);
                    output.resize(out_offset + out_avail);
                    GetAudio(&output[out_offset], out_avail);
                    out_offset+=out_avail;
                    in_offset += todo-left;
                    todo=left;
                }
                return output;
            }

        class CClient {
            public:
                CClient(impact_CInstance* instance)
                    {
                        assert(instance);
                        client=impact_NewClient(instance);
                        if (!client) throw std::bad_alloc();
                    }
                // non-copyable
                CClient( const CInstance& ) = delete;
                CClient& operator=( const CInstance& ) = delete;
                ~CClient()
                    {
                        if (client) {
                            impact_DeleteClient(client);
                            client=nullptr;
                        }
                    }
                std::string ProcessJson(const std::string &request, bool *NeedSave = nullptr)
                    {
                        assert(client);
                        int need_save=0;
                        const char *canswer = impact_ProcessJson(client, request.c_str(), &need_save);
                        if (!canswer) return {};
                        std::string answer(canswer);
                        impact_FreeJsonAnswer (canswer);
                        if (NeedSave) {
                            *NeedSave=(need_save!=0);
                        }
                        return answer;
                    }
            private:
                impact_CClientInstance *client = nullptr;
        };
        std::shared_ptr<CClient> NewClient()
            {
                assert(instance);
                return std::make_shared<CClient>(instance);
            }

    protected:
            template<typename T>
        unsigned int AddAudio(const T* payload, unsigned int nFrame)
            {
```

```
                assert(instance);
                return impact_AddAudio(instance, reinterpret_cast<const T*>(payload), nFrame, helper::SampleFormat<T>::format);
            }
            template<typename T>
        unsigned int GetAudio(T* payload, unsigned int max_nFrame)
            {
                assert(instance);
                return impact_GetAudio(instance, reinterpret_cast<T*>(payload), max_nFrame, helper::SampleFormat<T>::format);
            }

    private:
        impact_CParameters* params = nullptr;
        impact_CInstance* instance = nullptr;
        std::thread update_thread;
        std::vector< std::pair<std::string,std::string> > init_metadata;
#if IMPACT_HAS_WEBSERVER
        uint64_t webserver = SOUND4_INVALID_WEBSERVER_ID;
#endif // IMPACT_HAS_WEBSERVER
    };

}; // namespace impact
}; // namespace sound4

// bridge C callbacks to CPresetLoader
extern "C" {
    static char *impact_custom_reader(const fs_char *filename, void* userdata) {
        sound4::impact::CPresetLoader *preset_loader=reinterpret_cast<sound4::impact::CPresetLoader*>(userdata);
        std::string content=preset_loader->Read(filename);
        return strdup(content.c_str());
    }
    static void impact_custom_reader_free(char *content, void* userdata) {
        if (!content) return;
        free(content);
    }
    static int impact_custom_writer(const fs_char *filename, const char *content, void* userdata) {
        sound4::impact::CPresetLoader *preset_loader=reinterpret_cast<sound4::impact::CPresetLoader*>(userdata);
        auto res=preset_loader->Write(filename,content);
        return res?0:-1;
    }
    static int impact_custom_exists(const fs_char *filename, void* userdata) {
        sound4::impact::CPresetLoader *preset_loader=reinterpret_cast<sound4::impact::CPresetLoader*>(userdata);
        auto res=preset_loader->Exists(filename);
        return res?0:-1;
    }
    static fs_char** impact_custom_getall(void* userdata) {
        sound4::impact::CPresetLoader *preset_loader=reinterpret_cast<sound4::impact::CPresetLoader*>(userdata);
        auto res=preset_loader->GetAll();
        fs_char**all=(fs_char**)malloc((res.size()+1)*sizeof(const fs_char*));
        for (size_t n=0;n<res.size();n++) {
            all[n]=fs_strdup(res[n].c_str());
        }
        all[res.size()]=nullptr;
        return all;
    }
    static void impact_custom_getall_free(fs_char** all, void* userdata) {
        if (!all) return;
        for (fs_char** one=all;*one;one++) {
            free(*one);
        }
        free(all);
    }
    static int impact_custom_remove(const fs_char *filename, void* userdata) {
        sound4::impact::CPresetLoader *preset_loader=reinterpret_cast<sound4::impact::CPresetLoader*>(userdata);
        auto res=preset_loader->Remove(filename);
        return res?0:-1;
    }
    static int impact_custom_rename(const fs_char *from, const fs_char *to, void* userdata) {
        sound4::impact::CPresetLoader *preset_loader=reinterpret_cast<sound4::impact::CPresetLoader*>(userdata);
        auto res=preset_loader->Rename(from,to);
        return res?0:-1;
    }

};
```

11.11 sound4.impact.cl_dyn.hpp File Reference

C++ wrapper for dynamic loading library.

```
#include <string>
#include <exception>
#include <thread>
#include <functional>
#include <memory>
#include <array>
#include <assert.h>
#include <stdexcept>
#include <filesystem>
#include <string.h>
#include <libloaderapi.h>
#include "sound4.impact.cl.h"
#include <stringapiset.h>
#include <wchar.h>
```

Classes

- class sound4::impact::dyn::helper::DynFuncHolder< T >

  Helper to wrap a function pointer.
- class sound4::impact::dyn::helper::CDynLoader

  Helper to load dynamic library.
- class sound4::impact::dyn::CDynLib

  Dynamic library interface.
- struct sound4::impact::dyn::helper::SampleFormat< T >

  Helper for Sample format types.
- struct sound4::impact::dyn::helper::SampleFormat< int16_t >
- struct sound4::impact::dyn::helper::SampleFormat< int32_t >
- struct sound4::impact::dyn::helper::SampleFormat< float >
- class sound4::impact::dyn::CPresetLoader
- class sound4::impact::dyn::CInstance
- class sound4::impact::dyn::CInstance::CClient

Namespaces

- sound4
- sound4::impact
- dyn
- sound4::impact::dyn
- sound4::impact::dyn::helper

Macros

- #define fs_strdup wcsdup

  helper for OS dependant file name copy

Typedefs

- using sound4::impact::dyn::helper::DynLib_t = HMODULE
- using sound4::impact::dyn::helper::DynFunc_t = FARPROC
- using sound4::impact::dyn::SampleFormat = impact_SampleFormat
- using sound4::impact::dyn::LogSeverity = impact_LogSeverity
- using sound4::impact::dyn::log_cb_t = std::function< void(LogSeverity, const std::string &)>
- using sound4::impact::dyn::PresetChange_Kind = impact_PresetChange_Kind

Variables

- const size_t sound4::impact::dyn::ChunkFrames =IMPACT_AUDIOFRAME_COUNT
- const size_t sound4::impact::dyn::InputChannels =IMPACT_CHANNEL_COUNT∗IMPACT_AUDIO_INPUT_COUNT
- const size_t sound4::impact::dyn::OutputChannels =IMPACT_CHANNEL_COUNT∗IMPACT_AUDIO_OUTPUT_COUNT
- const size_t sound4::impact::dyn::InputSampleSize =ChunkFrames∗InputChannels
- const size_t sound4::impact::dyn::OutputSampleSize =ChunkFrames∗OutputChannels
- const size_t sound4::impact::dyn::SampleRate =IMPACT_SAMPLE_RATE

11.11.1 Detailed Description

You need to load the dynamic library, and if it succeed you can use it.

Definition in file sound4.impact.cl_dyn.hpp.

11.11.2 Macro Definition Documentation

11.11.2.1 fs_strdup

```
#define fs_strdup wcsdup
```

Definition at line 36 of file sound4.impact.cl_dyn.hpp.

11.12 sound4.impact.cl_dyn.hpp

```
00001
00007 #pragma once
00008
00009 #include <string>
00010 #include <exception>
00011 #include <thread>
00012 #include <functional>
00013 #include <memory>
00014 #include <array>
00015 #include <assert.h>
00016 #include <stdexcept>
00017 #include <filesystem>
00018 #include <string.h>
00019
00020 #ifdef __unix__
00021     #include <dlfcn.h>
00022 #elif defined(_WIN32)
00023     #include <libloaderapi.h>
00024 #else
00025     #error "Unsupported OS"
```

```
#endif

// Needed to have function declarations
#include "sound4.impact.cl.h"


#ifdef _WIN32
    #include <stringapiset.h>
    #include <wchar.h>
    #define fs_strdup wcsdup
#else // !_WIN32
    #define fs_strdup strdup
#endif // !_WIN32

// bridge C callbacks to CPresetLoader
extern "C" {
    static char *impact_custom_reader(const fs_char *filename, void* userdata);
    static void impact_custom_reader_free(char *content, void* userdata);
    static int impact_custom_writer(const fs_char *filename, const char *content, void* userdata);
    static int impact_custom_exists(const fs_char *filename, void* userdata);
    static fs_char** impact_custom_getall(void* userdata);
    static void impact_custom_getall_free(fs_char** all, void* userdata);
    static int impact_custom_remove(const fs_char *filename, void* userdata);
    static int impact_custom_rename(const fs_char *from, const fs_char *to, void* userdata);
};

namespace sound4 {
namespace impact {
namespace dyn {

    namespace helper {
        // Storage definition, depending on OS
        #ifdef __unix__
            using DynLib_t = void *;
            using DynFunc_t = void *;
        #else
            using DynLib_t = HMODULE;
            using DynFunc_t = FARPROC;
        #endif

            template <typename T>
        class DynFuncHolder {
        private:
            DynFunc_t m_ptr = NULL;
        public:
            DynFuncHolder() = default;
            DynFuncHolder(DynFunc_t a_ptr) : m_ptr(a_ptr) {}
            operator bool() const { return (m_ptr!=NULL); }
            bool IsOk() const { return m_ptr!=NULL; }
            operator T* () const { return reinterpret_cast<T*>(m_ptr); }
        };

        class CDynLoader {
            private:
                DynLib_t m_lib {};
            public:
                CDynLoader() = default;
                CDynLoader(CDynLoader&&) = default;
                // No copy allowed
                CDynLoader(const CDynLoader&) = delete;
                CDynLoader& operator=(CDynLoader const&) = delete;
                operator bool() const {
                    return (m_lib!=NULL);
                }
                bool IsOk() const {
                    return (m_lib!=NULL);
                }
                #ifdef __unix__
                    void Close() {
                        if (IsOk()) {
                            #ifndef __SANITIZE_ADDRESS__ // avoid unloading to keep call stack
                            dlclose(m_lib);
                            #endif // __SANITIZE_ADDRESS__
                            m_lib=NULL;
                        }
                    }
                    bool Load(const std::filesystem::path& dynlib)
                    {
                        Close();
                        m_lib=dlopen(dynlib.c_str(), RTLD_NOW|RTLD_LOCAL);
                        return IsOk();
                    }
                        template <typename T>
                    DynFuncHolder<T> GetSymbol(const std::string& name) {
                        auto ptr=dlsym(m_lib, name.c_str());
                        if (!ptr) {
                            throw std::runtime_error("Missing function in library");
```

```
                        }
                        return DynFuncHolder<T>(ptr);
                    }
                        template <typename T>
                    DynFuncHolder<T> GetSymbol_safe(const std::string& name) {
                        auto ptr=dlsym(m_lib, name.c_str());
                        return DynFuncHolder<T>(ptr);
                    }
                    static std::filesystem::path GetThisLibraryPath(void) {
                        static Dl_info info;
                        if (! dladdr((void *) GetThisLibraryPath, & info)) return {};
                        auto rp=realpath(info.dli_fname, NULL);
                        std::filesystem::path p(rp);
                        free(rp);
                        return p;
                    }
                #else
                    bool Load(const std::filesystem::path& dynlib, uint32_t loadflags = LOAD_LIBRARY_SEARCH_DLL_LOAD_DIR|LOAD_LIBRARY_SEARCH_APPLICATION_DIR|LOAD_LIBRARY_SEARCH_USER_DIRS|LOAD_LIBRARY_SEARCH_
                    {
                        Close();
                        m_lib=LoadLibraryExW(dynlib.c_str(), NULL, loadflags);
                        return IsOk();
                    }
                    void Close() {
                        if (IsOk()) {
                            // NOTE: On Windows, FreeLibrary may create a lot of troubles, should use FreeLibraryAndExitThread but...
                            // So define SOUND4_CALL_FREELIBRARY before including this file if you really want to unload the library dynamically
                            #if defined(SOUND4_CALL_FREELIBRARYANDEXITTHREAD)
                                FreeLibraryAndExitThread(m_lib, 0);
                            #elif defined(SOUND4_CALL_FREELIBRARY)
                                FreeLibrary(m_lib);
                            #endif // SOUND4_CALL_FREELIBRARY
                            m_lib=NULL;
                        }
                    }
                        template <typename T>
                    DynFuncHolder<T> GetSymbol(const std::string& name) {
                        auto ptr=GetProcAddress(m_lib, name.c_str());
                        if (!ptr) {
                            throw std::runtime_error("Missing function in library");
                        }
                        return DynFuncHolder<T>(ptr);
                    }
                        template <typename T>
                    DynFuncHolder<T> GetSymbol_safe(const std::string& name) {
                        auto ptr=GetProcAddress(m_lib, name.c_str());
                        return DynFuncHolder<T>(ptr);
                    }
                    static std::filesystem::path GetThisLibraryPath(void) {
                        static wchar_t path[MAX_PATH]={};
                        HMODULE hm = NULL;
                        if (GetModuleHandleExW(GET_MODULE_HANDLE_EX_FLAG_FROM_ADDRESS |
                                GET_MODULE_HANDLE_EX_FLAG_UNCHANGED_REFCOUNT,
                                (LPCWSTR) &GetThisLibraryPath, &hm))
                        {
                            if (GetModuleFileNameW(hm, path, MAX_PATH) > 0) {
                                return std::filesystem::path(path);
                            }
                        }
                        return {};
                    }
                #endif
                virtual ~CDynLoader() {
                    Close();
                }
        }; // class CDynLoader
    }; // namespace helper

    class CDynLib {
    private:
        helper::CDynLoader m_lib;
        bool m_bOK = false;
    public:
        CDynLib() = default;

        // All those wrapped functions have the same signature as there C impact_XXX
        helper::DynFuncHolder< decltype(impact_GetVersion              ) > GetVersion              ;
        helper::DynFuncHolder< decltype(impact_GetChunkSizeInFrames    ) > GetChunkSizeInFrames    ;
        helper::DynFuncHolder< decltype(impact_GetChannelCount         ) > GetChannelCount         ;
        helper::DynFuncHolder< decltype(impact_GetAudioInputCount      ) > GetAudioInputCount      ;
        helper::DynFuncHolder< decltype(impact_GetAudioOutputCount     ) > GetAudioOutputCount     ;
        helper::DynFuncHolder< decltype(impact_GetSampleRate           ) > GetSampleRate           ;
        helper::DynFuncHolder< decltype(impact_SetLoggerCallback       ) > SetLoggerCallback       ;
        helper::DynFuncHolder< decltype(impact_SetLogSeverity          ) > SetLogSeverity          ;
```

```
        helper::DynFuncHolder< decltype(impact_NewParameters          ) > NewParameters          ;
        helper::DynFuncHolder< decltype(impact_FreeParameters         ) > FreeParameters         ;
        helper::DynFuncHolder< decltype(impact_SetParameter           ) > SetParameter           ;
        helper::DynFuncHolder< decltype(impact_GetParameter           ) > GetParameter           ;
        helper::DynFuncHolder< decltype(impact_FreeParameterValue     ) > FreeParameterValue     ;
        helper::DynFuncHolder< decltype(impact_InitProcess            ) > InitProcess            ;
        helper::DynFuncHolder< decltype(impact_InitProcess2           ) > InitProcess2           ;
        helper::DynFuncHolder< decltype(impact_TerminateProcess       ) > TerminateProcess       ;
        helper::DynFuncHolder< decltype(impact_ExitProcess            ) > ExitProcess            ;
        helper::DynFuncHolder< decltype(impact_StartUpdateThread      ) > StartUpdateThread      ;
        helper::DynFuncHolder< decltype(impact_StopUpdateThread       ) > StopUpdateThread       ;
        helper::DynFuncHolder< decltype(impact_WaitUpdateThreadReady  ) > WaitUpdateThreadReady  ;
        helper::DynFuncHolder< decltype(impact_ProcessAudio           ) > ProcessAudio           ;
        helper::DynFuncHolder< decltype(impact_ProcessAudio_Planar    ) > ProcessAudio_Planar    ;
        helper::DynFuncHolder< decltype(impact_GetBufferIn            ) > GetBufferIn            ;
        helper::DynFuncHolder< decltype(impact_GetBufferOut           ) > GetBufferOut           ;
        helper::DynFuncHolder< decltype(impact_GetEstimatedDelay      ) > GetEstimatedDelay      ;
        helper::DynFuncHolder< decltype(impact_GetFormatName          ) > GetFormatName          ;
        helper::DynFuncHolder< decltype(impact_GetFormatFromName      ) > GetFormatFromName      ;
        helper::DynFuncHolder< decltype(impact_GetBytesFromFormat     ) > GetBytesFromFormat     ;
        helper::DynFuncHolder< decltype(impact_GetMaxPacketFrame      ) > GetMaxPacketFrame      ;
        helper::DynFuncHolder< decltype(impact_AddAudio               ) > AddAudio               ;
        helper::DynFuncHolder< decltype(impact_AddPadAudio            ) > AddPadAudio            ;
        helper::DynFuncHolder< decltype(impact_GetOutputCount         ) > GetOutputCount         ;
        helper::DynFuncHolder< decltype(impact_GetAudio               ) > GetAudio               ;
        helper::DynFuncHolder< decltype(impact_AudioConvertFrom       ) > AudioConvertFrom       ;
        helper::DynFuncHolder< decltype(impact_AudioConvertTo         ) > AudioConvertTo         ;
        helper::DynFuncHolder< decltype(impact_StereoToMono           ) > StereoToMono           ;
        helper::DynFuncHolder< decltype(impact_MonoToStereo           ) > MonoToStereo           ;
        helper::DynFuncHolder< decltype(impact_AudioMonoFromLiveStereo) > AudioMonoFromLiveStereo;
        helper::DynFuncHolder< decltype(impact_AudioMonoToLiveStereo  ) > AudioMonoToLiveStereo  ;
        helper::DynFuncHolder< decltype(impact_NewClient              ) > NewClient              ;
        helper::DynFuncHolder< decltype(impact_DeleteClient           ) > DeleteClient           ;
        helper::DynFuncHolder< decltype(impact_ProcessJson            ) > ProcessJson            ;
        helper::DynFuncHolder< decltype(impact_FreeJsonAnswer         ) > FreeJsonAnswer         ;
        helper::DynFuncHolder< decltype(impact_SaveState              ) > SaveState              ;
#if IMPACT_HAS_WEBSERVER
        helper::DynFuncHolder< decltype(impact_Webserver_tcp          ) > Webserver_tcp          ;
        helper::DynFuncHolder< decltype(impact_Webserver_tcp2         ) > Webserver_tcp2         ;
        helper::DynFuncHolder< decltype(impact_Webserver              ) > Webserver              ;
        helper::DynFuncHolder< decltype(impact_Webserver_Stop         ) > Webserver_Stop         ;
        helper::DynFuncHolder< decltype(impact_Webserver_Status       ) > Webserver_Status       ;
#endif // IMPACT_HAS_WEBSERVER

        // added 2023-03-22
        helper::DynFuncHolder< decltype(impact_StereoToMono_Planar    ) > StereoToMono_Planar    ;
        helper::DynFuncHolder< decltype(impact_MonoToStereo_Planar    ) > MonoToStereo_Planar    ;

        // added 2023-05-22
        helper::DynFuncHolder< decltype(impact_SetMetadata            ) > SetMetadata            ;
        helper::DynFuncHolder< decltype(impact_GetMetadataInfos       ) > GetMetadataInfos       ;
        helper::DynFuncHolder< decltype(impact_FreeMetadataInfos      ) > FreeMetadataInfos      ;

        // added 2023-06-20
        helper::DynFuncHolder< decltype(impact_SetPresetManager                ) > SetPresetManager
    ;
        helper::DynFuncHolder< decltype(impact_PresetManager_InformChange      ) >
    PresetManager_InformChange      ;

        // added 2023-09-21
        helper::DynFuncHolder< decltype(impact_GetPossibleChunkSizeInFrames    ) >
    GetPossibleChunkSizeInFrames   ;
        helper::DynFuncHolder< decltype(impact_GetProcessChunkFrames           ) >
    GetProcessChunkFrames   ;
        helper::DynFuncHolder< decltype(impact_InitProcess3                    ) > InitProcess3
    ;

#if IMPACT_HAS_CLOUDBUS
        // added 2023-06-26
        helper::DynFuncHolder< decltype(impact_NewBus                 ) > NewBus      ;
        helper::DynFuncHolder< decltype(impact_FreeBus                ) > FreeBus      ;
        helper::DynFuncHolder< decltype(impact_SetInstanceBus         ) > SetInstanceBus      ;
#endif // IMPACT_HAS_CLOUDBUS

        bool Load(const std::filesystem::path& path = std::filesystem::current_path()) {
            std::string filename;
            #ifdef __unix__
                filename="libsound4.impact.cl.so";
            #else
                filename="sound4.impact.cl.dll";
            #endif
            auto p = path;
            p.append(filename);
            m_lib.Load(p.native());
            if (!m_lib.IsOk()) {
                // Search in the same directory this code is in
                p =helper::CDynLoader::GetThisLibraryPath();
```

```
                p.append(filename);
                m_lib.Load(p.native());
                if (!m_lib.IsOk()) {
                    return false;
                }
            }
            // Load all C functions
            try {
                GetVersion              = m_lib.GetSymbol< decltype(impact_GetVersion              )
    >("impact_GetVersion"            );
                GetChunkSizeInFrames    = m_lib.GetSymbol< decltype(impact_GetChunkSizeInFrames    )
    >("impact_GetChunkSizeInFrames"  );
                GetChannelCount         = m_lib.GetSymbol< decltype(impact_GetChannelCount         )
    >("impact_GetChannelCount"       );
                GetAudioInputCount      = m_lib.GetSymbol< decltype(impact_GetAudioInputCount      )
    >("impact_GetAudioInputCount"    );
                GetAudioOutputCount     = m_lib.GetSymbol< decltype(impact_GetAudioOutputCount     )
    >("impact_GetAudioOutputCount"   );
                GetSampleRate           = m_lib.GetSymbol< decltype(impact_GetSampleRate           )
    >("impact_GetSampleRate"         );
                SetLoggerCallback       = m_lib.GetSymbol< decltype(impact_SetLoggerCallback       )
    >("impact_SetLoggerCallback"     );
                SetLogSeverity          = m_lib.GetSymbol< decltype(impact_SetLogSeverity          )
    >("impact_SetLogSeverity"        );
                NewParameters           = m_lib.GetSymbol< decltype(impact_NewParameters           )
    >("impact_NewParameters"         );
                FreeParameters          = m_lib.GetSymbol< decltype(impact_FreeParameters          )
    >("impact_FreeParameters"        );
                SetParameter            = m_lib.GetSymbol< decltype(impact_SetParameter            )
    >("impact_SetParameter"          );
                GetParameter            = m_lib.GetSymbol< decltype(impact_GetParameter            )
    >("impact_GetParameter"          );
                FreeParameterValue      = m_lib.GetSymbol< decltype(impact_FreeParameterValue      )
    >("impact_FreeParameterValue"    );
                InitProcess             = m_lib.GetSymbol< decltype(impact_InitProcess             )
    >("impact_InitProcess"           );
                InitProcess2            = m_lib.GetSymbol< decltype(impact_InitProcess2            )
    >("impact_InitProcess2"          );
                TerminateProcess        = m_lib.GetSymbol< decltype(impact_TerminateProcess        )
    >("impact_TerminateProcess"      );
                ExitProcess             = m_lib.GetSymbol< decltype(impact_ExitProcess             )
    >("impact_ExitProcess"           );
                StartUpdateThread       = m_lib.GetSymbol< decltype(impact_StartUpdateThread       )
    >("impact_StartUpdateThread"     );
                StopUpdateThread        = m_lib.GetSymbol< decltype(impact_StopUpdateThread        )
    >("impact_StopUpdateThread"      );
                WaitUpdateThreadReady   = m_lib.GetSymbol< decltype(impact_WaitUpdateThreadReady   )
    >("impact_WaitUpdateThreadReady" );
                ProcessAudio            = m_lib.GetSymbol< decltype(impact_ProcessAudio            )
    >("impact_ProcessAudio"          );
                ProcessAudio_Planar     = m_lib.GetSymbol< decltype(impact_ProcessAudio_Planar     )
    >("impact_ProcessAudio_Planar"   );
                GetBufferIn             = m_lib.GetSymbol< decltype(impact_GetBufferIn             )
    >("impact_GetBufferIn"           );
                GetBufferOut            = m_lib.GetSymbol< decltype(impact_GetBufferOut            )
    >("impact_GetBufferOut"          );
                GetEstimatedDelay       = m_lib.GetSymbol< decltype(impact_GetEstimatedDelay       )
    >("impact_GetEstimatedDelay"     );
                GetFormatName           = m_lib.GetSymbol< decltype(impact_GetFormatName           )
    >("impact_GetFormatName"         );
                GetFormatFromName       = m_lib.GetSymbol< decltype(impact_GetFormatFromName       )
    >("impact_GetFormatFromName"     );
                GetBytesFromFormat      = m_lib.GetSymbol< decltype(impact_GetBytesFromFormat      )
    >("impact_GetBytesFromFormat"    );
                GetMaxPacketFrame       = m_lib.GetSymbol< decltype(impact_GetMaxPacketFrame       )
    >("impact_GetMaxPacketFrame"     );
                AddAudio                = m_lib.GetSymbol< decltype(impact_AddAudio                )
    >("impact_AddAudio"              );
                AddPadAudio             = m_lib.GetSymbol< decltype(impact_AddPadAudio             )
    >("impact_AddPadAudio"           );
                GetOutputCount          = m_lib.GetSymbol< decltype(impact_GetOutputCount          )
    >("impact_GetOutputCount"        );
                GetAudio                = m_lib.GetSymbol< decltype(impact_GetAudio                )
    >("impact_GetAudio"              );
                AudioConvertFrom        = m_lib.GetSymbol< decltype(impact_AudioConvertFrom        )
    >("impact_AudioConvertFrom"      );
                AudioConvertTo          = m_lib.GetSymbol< decltype(impact_AudioConvertTo          )
    >("impact_AudioConvertTo"        );
                StereoToMono            = m_lib.GetSymbol< decltype(impact_StereoToMono            )
    >("impact_StereoToMono"          );
                MonoToStereo            = m_lib.GetSymbol< decltype(impact_MonoToStereo            )
    >("impact_MonoToStereo"          );
                AudioMonoFromLiveStereo = m_lib.GetSymbol< decltype(impact_AudioMonoFromLiveStereo)
    >("impact_AudioMonoFromLiveStereo");
                AudioMonoToLiveStereo   = m_lib.GetSymbol< decltype(impact_AudioMonoToLiveStereo   )
    >("impact_AudioMonoToLiveStereo"  );
                NewClient               = m_lib.GetSymbol< decltype(impact_NewClient               )
```

```
>("impact_NewClient"              );
                DeleteClient            = m_lib.GetSymbol< decltype(impact_DeleteClient           )
>("impact_DeleteClient"           );
                ProcessJson             = m_lib.GetSymbol< decltype(impact_ProcessJson            )
>("impact_ProcessJson"            );
                FreeJsonAnswer          = m_lib.GetSymbol< decltype(impact_FreeJsonAnswer         )
>("impact_FreeJsonAnswer"         );
                SaveState               = m_lib.GetSymbol< decltype(impact_SaveState              )
>("impact_SaveState"              );
#if IMPACT_HAS_WEBSERVER
                Webserver_tcp           = m_lib.GetSymbol< decltype(impact_Webserver_tcp          )
>("impact_Webserver_tcp"          );
                Webserver_tcp2          = m_lib.GetSymbol< decltype(impact_Webserver_tcp2         )
>("impact_Webserver_tcp2"         );
                Webserver               = m_lib.GetSymbol< decltype(impact_Webserver              )
>("impact_Webserver"              );
                Webserver_Stop          = m_lib.GetSymbol< decltype(impact_Webserver_Stop         )
>("impact_Webserver_Stop"         );
                Webserver_Status        = m_lib.GetSymbol< decltype(impact_Webserver_Status       )
>("impact_Webserver_Status"       );
#endif // IMPACT_HAS_WEBSERVER

                // added 2023-03-22
                StereoToMono_Planar     = m_lib.GetSymbol< decltype(impact_StereoToMono_Planar    )
>("impact_StereoToMono_Planar"    );
                MonoToStereo_Planar     = m_lib.GetSymbol< decltype(impact_MonoToStereo_Planar    )
>("impact_MonoToStereo_Planar"    );

            } catch (std::runtime_error& ) {
                return false;
            }
            // C functions allowed to be missing
            try {
                // added 2023-05-22
                SetMetadata             = m_lib.GetSymbol< decltype(impact_SetMetadata            )
>("impact_SetMetadata"            );
                GetMetadataInfos        = m_lib.GetSymbol< decltype(impact_GetMetadataInfos       )
>("impact_GetMetadataInfos"       );
                FreeMetadataInfos       = m_lib.GetSymbol< decltype(impact_FreeMetadataInfos      )
>("impact_FreeMetadataInfos"      );
                // added 2023-06-20
                SetPresetManager            = m_lib.GetSymbol< decltype(impact_SetPresetManager
        ) >("impact_SetPresetManager"             );
                PresetManager_InformChange  = m_lib.GetSymbol<
decltype(impact_PresetManager_InformChange     ) >("impact_PresetManager_InformChange"    );

                // added 2023-09-21
                GetPossibleChunkSizeInFrames = m_lib.GetSymbol<
decltype(impact_GetPossibleChunkSizeInFrames  ) >("impact_GetPossibleChunkSizeInFrames"  );
                GetProcessChunkFrames        = m_lib.GetSymbol< decltype(impact_GetProcessChunkFrames
        ) >("impact_GetProcessChunkFrames"        );
                InitProcess3                 = m_lib.GetSymbol< decltype(impact_InitProcess3
        ) >("impact_InitProcess3"                 );

#if IMPACT_HAS_CLOUDBUS
                // added 2023-06-26
                NewBus                  = m_lib.GetSymbol< decltype(impact_NewBus                 )
>("impact_NewBus"                 );
                FreeBus                 = m_lib.GetSymbol< decltype(impact_FreeBus                )
>("impact_FreeBus"                );
                SetInstanceBus          = m_lib.GetSymbol< decltype(impact_SetInstanceBus         )
>("impact_SetInstanceBus"         );
#endif // IMPACT_HAS_CLOUDBUS
            }  catch (std::runtime_error& ) {
                // Ignored, handler will take care
            }

            m_bOK=true;
            return true;
        }
        bool IsOk() const {
            return m_bOK;
        }

    };



    static constexpr const char *process_name = "SOUND4 IMPACT.CL";
    static constexpr const char *process_shortname = "impact";
    using SampleFormat = impact_SampleFormat;
    const size_t ChunkFrames=IMPACT_AUDIOFRAME_COUNT;
    const size_t InputChannels=IMPACT_CHANNEL_COUNT*IMPACT_AUDIO_INPUT_COUNT;
    const size_t OutputChannels=IMPACT_CHANNEL_COUNT*IMPACT_AUDIO_OUTPUT_COUNT;
    const size_t InputSampleSize=ChunkFrames*InputChannels;
    const size_t OutputSampleSize=ChunkFrames*OutputChannels;
    const size_t SampleRate=IMPACT_SAMPLE_RATE;
```

```
    namespace helper {
        #ifdef _WIN32
        static inline std::string WStringToUTF8(const std::wstring& wstr) {
            if (wstr.empty()) return {};
            int size = WideCharToMultiByte(CP_UTF8, 0, &wstr[0], (int)wstr.size(), NULL, 0, NULL, NULL);
            std::string str(size, 0);
            WideCharToMultiByte(CP_UTF8, 0, &wstr[0], (int)wstr.size(), &str[0], size, NULL, NULL);
            return str;
        }
        static inline std::wstring UTF8ToWString(const std::string& str) {
            if (str.empty()) return std::wstring();
            int size = MultiByteToWideChar(CP_UTF8, 0, &str[0], (int)str.size(), NULL, 0);
            std::wstring wstr(size, 0);
            MultiByteToWideChar(CP_UTF8, 0, &str[0], (int)str.size(), &wstr[0], size);
            return wstr;
        }
        #endif // _WIN32

            template<typename T>
        struct SampleFormat {
        };
            template<>
        struct SampleFormat<int16_t> {
            const impact_SampleFormat format = S16_NATIVE;
        };
            template<>
        struct SampleFormat<int32_t> {
            const impact_SampleFormat format = S32_NATIVE;
        };
            template<>
        struct SampleFormat<float> {
            const impact_SampleFormat format = F32_NATIVE;
        };


        static inline void AudioConvertFrom(CDynLib& dynlib, const uint8_t *payload, float *output, size_t nSpl, impact_SampleFormat fmt) {
            dynlib.AudioConvertFrom(payload, output, nSpl, fmt);
        }
        static inline void AudioConvertTo(CDynLib& dynlib, const float *input, uint8_t *payload, size_t nSpl, impact_SampleFormat fmt) {
            dynlib.AudioConvertTo(input, payload, nSpl, fmt);
        }
        static inline void StereoToMono(CDynLib& dynlib, const float *input, float *output, size_t nSpl) {
            dynlib.StereoToMono(input, output, nSpl);
        }
        static inline void MonoToStereo(CDynLib& dynlib, const float *input, float *output, size_t nSpl) {
            dynlib.MonoToStereo(input, output, nSpl);
        }
        static inline void StereoToMono_Planar(CDynLib& dynlib, const float *inputL, const float *inputR, float *output, size_t nSpl) {
            dynlib.StereoToMono_Planar(inputL, inputR, output, nSpl);
        }
        static inline void MonoToStereo_Planar(CDynLib& dynlib, const float *input, float *outputL, float *outputR, size_t nSpl) {
            dynlib.MonoToStereo_Planar(input, outputL, outputR, nSpl);
        }
        static inline void AudioMonoFromLiveStereo(CDynLib& dynlib, const uint8_t *payload, float *output) {
            dynlib.AudioMonoFromLiveStereo(payload, output);
        }
        static inline void AudioMonoToLiveStereo(CDynLib& dynlib, const float *input, uint8_t *payload) {
            dynlib.AudioMonoToLiveStereo(input, payload);
        }

    }; // namespace helper

    static inline std::string GetVersion(CDynLib& dynlib)
        {
            return std::string(dynlib.GetVersion());
        }

    static inline unsigned int GetChunkSizeInFrames(CDynLib& dynlib)
        {
            return dynlib.GetChunkSizeInFrames();
        }

    static inline std::vector<unsigned int> GetPossibleChunkSizeInFrames(CDynLib& dynlib)
        {
            std::vector<unsigned int> list;
            if (dynlib.GetPossibleChunkSizeInFrames) {
                for (unsigned int* src=dynlib.GetPossibleChunkSizeInFrames();*src;src++) {
```

```
                    list.push_back(*src);
                }
            } else {
                list.push_back(64);
            }
            return list;
        }

    static inline unsigned int GetChannelCount(CDynLib& dynlib)
        {
            return dynlib.GetChannelCount();
        }

    static inline unsigned int GetAudioInputCount(CDynLib& dynlib)
        {
            return dynlib.GetAudioInputCount();
        }

    static inline unsigned int GetAudioOutputCount(CDynLib& dynlib)
        {
            return dynlib.GetAudioOutputCount();
        }

    static inline unsigned int GetSampleRate(CDynLib& dynlib)
        {
            return dynlib.GetSampleRate();
        }

    static inline void SanityCheck(CDynLib& dynlib, bool a_bCheckFrames = true)
    {
        // Do some sanity checks
        if (dynlib.GetSampleRate()!=IMPACT_SAMPLE_RATE) {
            throw std::runtime_error("Bad library sampling rate");
        }
        if (a_bCheckFrames && dynlib.GetChunkSizeInFrames()!=ChunkFrames) {
            throw std::runtime_error("Bad library frame size");
        }
        if (dynlib.GetChannelCount()!=IMPACT_CHANNEL_COUNT) {
            throw std::runtime_error("Bad library channel count");
        }
        if (dynlib.GetAudioInputCount()!=IMPACT_AUDIO_INPUT_COUNT) {
            throw std::runtime_error("Bad library input count");
        }
        if (dynlib.GetAudioOutputCount()!=IMPACT_AUDIO_OUTPUT_COUNT) {
            throw std::runtime_error("Bad library output count");
        }
    }

    static inline std::string GetFormatName(CDynLib& dynlib, const SampleFormat fmt)
        {
            return std::string(dynlib.GetFormatName(fmt));
        }

    static inline SampleFormat GetFormatFromName(CDynLib& dynlib, const std::string& name)
        {
            return dynlib.GetFormatFromName(name.c_str());
        }

    static inline unsigned int GetBytesFromFormat(CDynLib& dynlib, const SampleFormat fmt)
        {
            return dynlib.GetBytesFromFormat(fmt);
        }

    using LogSeverity = impact_LogSeverity;
    using log_cb_t = std::function<void(LogSeverity,const std::string&)>;

    static log_cb_t _log_cb;
    extern "C" {
        static inline void _log_cb_c(impact_LogSeverity severity, const char *c_msg) {
            _log_cb(severity, std::string(c_msg));
        }
    }
    static inline void SetLogSeverity(CDynLib& dynlib, LogSeverity severity)
        {
            dynlib.SetLogSeverity(severity);
        }
    static inline void SetLoggerCallback(CDynLib& dynlib, log_cb_t cb)
        {
            _log_cb=cb;
            dynlib.SetLoggerCallback(&_log_cb_c);
        }

#if IMPACT_HAS_CLOUDBUS
    class CBus {
        public:
            CBus(CDynLib& dynlib)
                : m_dynlib(dynlib)
```

```
                {
                    if (m_dynlib.NewBus) {
                        m_bus=m_dynlib.NewBus();
                    }
                }
            ~CBus()
                {
                    if (m_bus && m_dynlib.FreeBus) {
                        m_dynlib.FreeBus(m_bus);
                    }
                    m_bus = nullptr;
                }
            impact_CBus *Get() const { return m_bus; }
        private:
            CDynLib& m_dynlib;
            impact_CBus *m_bus;
    };
#endif // IMPACT_HAS_CLOUDBUS

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

        virtual bool IsReadOnly() = 0;
        virtual bool Exists(const std::filesystem::path &name) = 0;
        virtual bool Remove(const std::filesystem::path &name) = 0;
        virtual bool Rename(const std::filesystem::path &from, const std::filesystem::path &to) = 0;
        virtual std::vector<std::filesystem::path> GetAll() = 0;
        virtual std::string Read(const std::filesystem::path &filename) = 0;
        virtual bool Write(const std::filesystem::path &filename, const std::string &content) =0;
    };
    using PresetChange_Kind = impact_PresetChange_Kind;

    class CInstance {
    public:
        CInstance(CDynLib& dynlib)
        : m_dynlib(dynlib)
            {
                params = m_dynlib.NewParameters();
                if (!params) throw std::bad_alloc();
            }
        // non-copyable
        CInstance( const CInstance& ) = delete;
        CInstance& operator=( const CInstance& ) = delete;
        ~CInstance()
            {
                Stop();
                if (params) m_dynlib.FreeParameters(params);
            }
       bool IsOk() const { return instance!=nullptr; }
        void SetParam(const std::string& name, const std::string& value)
            {
                assert(params);
                m_dynlib.SetParameter(params, name.c_str(), value.c_str());
            }
        std::string GetParam(const std::string& name)
            {
                assert(params);
                auto c_value=m_dynlib.GetParameter(params, name.c_str());
                std::string ret(c_value);
                m_dynlib.FreeParameterValue(c_value);
                return ret;
            }
#if IMPACT_HAS_CLOUDBUS
            void SetBus(const CBus& bus) {
                assert(params);
                if (m_dynlib.SetInstanceBus) {
                    m_dynlib.SetInstanceBus(params,bus.Get());
                }
            }
#endif // IMPACT_HAS_CLOUDBUS

        void SetPresetManager(CPresetLoader *preset_manager)
            {
                if (!preset_manager) return;
                if (!m_dynlib.SetPresetManager) return;
                assert(params);
                m_dynlib.SetPresetManager(params,
                    impact_custom_reader,
                    impact_custom_reader_free,
                    impact_custom_writer,
                    impact_custom_exists,
                    impact_custom_getall,
                    impact_custom_getall_free,
                    impact_custom_remove,
                    impact_custom_rename,
```

```
                    preset_manager->IsReadOnly(),
                    preset_manager
                );
            }

        bool Create(const std::string& LoginKey, const std::string& RadioName, const std::string& AWS_Access_ID, const std::string& AWS_Access_Secret, const std::filesystem::path& save_path, int json_port = 0, unsigned int frames_per_chunk=64)
            {
                assert(!instance);
                std::string l_save_path;
                if (!save_path.empty()) {
                    l_save_path = save_path.u8string();
                }
                if (m_dynlib.InitProcess3) {
                    instance = m_dynlib.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);
                } else if (frames_per_chunk!=64) {
                    return false;
                } else {
                    instance = m_dynlib.InitProcess2(LoginKey.c_str(), RadioName.c_str(), AWS_Access_ID.c_str(), AWS_Access_Secret.c_str(), l_save_path.c_str(), params);
                }
                if (!instance) return false;
                if (!init_metadata.empty()) {
                    for (auto&& [key,value]: init_metadata) {
                        m_dynlib.SetMetadata(instance, key.c_str(), value.c_str());
                    }
                    init_metadata.clear();
                }
                update_thread = std::thread([this, json_port](){
                    OnUpdateThreadStart();
                    m_dynlib.StartUpdateThread(instance, json_port);
                    OnUpdateThreadStop();
                });
                int timeout=5000;
                #ifdef DEBUG
                    timeout=60*1000; // For debugger pauses
                #endif
                if (m_dynlib.WaitUpdateThreadReady(instance, timeout)<0) {
                    return false;
                }
                return true;
            }

        unsigned int GetChunkFrames() {
            if (instance && m_dynlib.GetProcessChunkFrames) {
                return m_dynlib.GetProcessChunkFrames(instance);
            }
            return 64;
        }

        void PresetManager_InformChange(const std::filesystem::path& relative_path, PresetChange_Kind change_kind)
            {
                if (m_dynlib.PresetManager_InformChange) {
                    m_dynlib.PresetManager_InformChange(instance, relative_path.c_str(), (impact_PresetChange_Kind) change_kind);
                }
            }

        void SetMetadata(const std::string& key, const char* value)
            {
                if (m_dynlib.SetMetadata) {
                    if (instance) {
                        m_dynlib.SetMetadata(instance, key.c_str(), value);
                    } else {
                        init_metadata.push_back( {key, value} );
                    }
                }
            }

        std::vector< std::tuple<std::string,std::string> > GetMetadataInfos()
            {
                std::vector< std::tuple<std::string,std::string> > values;
                if (m_dynlib.GetMetadataInfos) {
                    const char** c_values = m_dynlib.GetMetadataInfos(instance);
                    if (c_values) {
                        for (const char** c_value=c_values; *c_value; ) {
                            std::string key(*c_value);
                            c_value++;
                            if (*c_value) {
                                std::string descr(*c_value);
                                c_value++;
                                values.push_back( {key,descr} );
                            }
                        }
```

```
                        m_dynlib.FreeMetadataInfos(instance, c_values);
                    }
                }
                return values;
            }

        // Can be overloaded, called at start of UpdateThread in the thread
        virtual void OnUpdateThreadStart() {}
        // Can be overloaded to call at end of UpdateThread in the thread
        virtual void OnUpdateThreadStop() {}

#if IMPACT_HAS_WEBSERVER
        bool StartWebServer(int http_port, int https_port=0)
            {
                assert(instance);
                assert(webserver==SOUND4_INVALID_WEBSERVER_ID);
                webserver = m_dynlib.Webserver(http_port,https_port,instance);
                if (webserver==SOUND4_INVALID_WEBSERVER_ID) {
                    return false;
                }
                int web_status=m_dynlib.Webserver_Status(webserver);
                if (http_port && (web_status & SOUND4_WEBSERVER_HTTP_OK) == 0) {
                    return false;
                }
                if (https_port && (web_status & SOUND4_WEBSERVER_HTTPS_OK) == 0) {
                    return false;
                }
                return true;
            }
        void StopWebServer()
            {
                if (webserver != SOUND4_INVALID_WEBSERVER_ID) {
                    m_dynlib.Webserver_Stop(webserver, 1000);
                    webserver = SOUND4_INVALID_WEBSERVER_ID;
                }
            }
#endif // IMPACT_HAS_WEBSERVER

        int Terminate()
            {
                if (!instance) return 0;
                return m_dynlib.TerminateProcess(instance);
            }

        void Stop()
            {
                if (!instance) return;
#if IMPACT_HAS_WEBSERVER
                StopWebServer();
#endif // IMPACT_HAS_WEBSERVER
                if (update_thread.joinable()) {
                    m_dynlib.StopUpdateThread(instance); // this will exit the update thread
                    update_thread.join();
                }
                m_dynlib.ExitProcess(instance);
                instance=nullptr;
            }

        unsigned int GetEstimatedDelay()
            {
                assert(instance);
                return m_dynlib.GetEstimatedDelay(instance);
            }

        std::array<float,InputSampleSize>& GetBufferIn()
            {
                assert(instance);
                float *buf=m_dynlib.GetBufferIn(instance);
                return reinterpret_cast< std::array<float,InputSampleSize>& >(*buf);

            }
        std::array<float,OutputSampleSize>& GetBufferOut()
            {
                assert(instance);
                float *buf= m_dynlib.GetBufferOut(instance);
                return reinterpret_cast< std::array<float,OutputSampleSize>& >(*buf);
            }
        void ProcessAudio()
            {
                assert(instance);
                m_dynlib.ProcessAudio(instance, m_dynlib.GetBufferIn(instance),
 m_dynlib.GetBufferOut(instance));
            }
        // void ProcessAudio_Planar(const std::array< const std::array<const float,ChunkFrames>,
 InputChannels>& input, std::array< std::array<float,ChunkFrames>, OutputChannels>& output)
        void ProcessAudio_Planar(float const * const *input, float * const *output)
            {
```

```
                assert(instance);
                m_dynlib.ProcessAudio_Planar(instance, input, output);
            }

            template<typename T>
        std::vector<T> ProcessAnyAudio(const std::vector<T> input)
            {
                assert(instance);
                std::vector<T> output;
                unsigned int out_offset=0;
                unsigned int in_offset=0;
                unsigned int todo = input.size();
                while (todo>0) {
                    unsigned int left = AddAudio(&input[in_offset], todo);
                    unsigned int out_avail = m_dynlib.GetOutputCount(instance);
                    output.resize(out_offset + out_avail);
                    GetAudio(&output[out_offset], out_avail);
                    out_offset+=out_avail;
                    in_offset += todo-left;
                    todo=left;
                }
                return output;
            }

        class CClient {
            public:
                CClient(impact_CInstance* instance, CDynLib& dynlib)
                : m_dynlib(dynlib)
                    {
                        assert(instance);
                        client=m_dynlib.NewClient(instance);
                        if (!client) throw std::bad_alloc();
                    }
                // non-copyable
                CClient( const CInstance& ) = delete;
                CClient& operator=( const CInstance& ) = delete;
                ~CClient()
                    {
                        if (client) {
                            m_dynlib.DeleteClient(client);
                            client=nullptr;
                        }
                    }
                std::string ProcessJson(const std::string &request, bool *NeedSave = nullptr)
                    {
                        assert(client);
                        int need_save=0;
                        const char *canswer = m_dynlib.ProcessJson(client, request.c_str(), &need_save);
                        if (!canswer) return {};
                        std::string answer(canswer);
                        m_dynlib.FreeJsonAnswer (canswer);
                        if (NeedSave) {
                            *NeedSave=(need_save!=0);
                        }
                        return answer;
                    }
            private:
                CDynLib& m_dynlib;
                impact_CClientInstance *client = nullptr;
        };
        std::shared_ptr<CClient> NewClient()
            {
                assert(instance);
                return std::make_shared<CClient>(instance, m_dynlib);
            }

    protected:
            template<typename T>
        unsigned int AddAudio(const T* payload, unsigned int nFrame)
            {
                assert(instance);
                return m_dynlib.AddAudio(instance, reinterpret_cast<const T*>(payload), nFrame, helper::SampleFormat<T>::format);
            }
            template<typename T>
        unsigned int GetAudio(T* payload, unsigned int max_nFrame)
            {
                assert(instance);
                return m_dynlib.GetAudio(instance, reinterpret_cast<T*>(payload), max_nFrame, helper::SampleFormat<T>::format);
            }

    private:
        CDynLib& m_dynlib;
        impact_CParameters* params = nullptr;
        impact_CInstance* instance = nullptr;
```

```
        std::thread update_thread;
        std::vector< std::pair<std::string,std::string> > init_metadata;
#if IMPACT_HAS_WEBSERVER
        uint64_t webserver = SOUND4_INVALID_WEBSERVER_ID;
#endif // IMPACT_HAS_WEBSERVER
    };

}; // namespace dyn
}; // namespace impact
}; // namespace sound4


// bridge C callbacks to CPresetLoader
extern "C" {
    static char *impact_custom_reader(const fs_char *filename, void* userdata) {
        sound4::impact::dyn::CPresetLoader *preset_loader=reinterpret_cast<sound4::impact::dyn::CPresetLoader*>(userdata);
        std::string content=preset_loader->Read(filename);
        return strdup(content.c_str());
    }
    static void impact_custom_reader_free(char *content, void* userdata) {
        if (!content) return;
        free(content);
    }
    static int impact_custom_writer(const fs_char *filename, const char *content, void* userdata) {
        sound4::impact::dyn::CPresetLoader *preset_loader=reinterpret_cast<sound4::impact::dyn::CPresetLoader*>(userdata);
        auto res=preset_loader->Write(filename,content);
        return res?0:-1;
    }
    static int impact_custom_exists(const fs_char *filename, void* userdata) {
        sound4::impact::dyn::CPresetLoader *preset_loader=reinterpret_cast<sound4::impact::dyn::CPresetLoader*>(userdata);
        auto res=preset_loader->Exists(filename);
        return res?0:-1;
    }
    static fs_char** impact_custom_getall(void* userdata) {
        sound4::impact::dyn::CPresetLoader *preset_loader=reinterpret_cast<sound4::impact::dyn::CPresetLoader*>(userdata);
        auto res=preset_loader->GetAll();
        fs_char**all=(fs_char**)malloc((res.size()+1)*sizeof(const fs_char*));
        for (size_t n=0;n<res.size();n++) {
            all[n]=fs_strdup(res[n].c_str());
        }
        all[res.size()]=nullptr;
        return all;
    }
    static void impact_custom_getall_free(fs_char** all, void* userdata) {
        if (!all) return;
        for (fs_char** one=all;*one;one++) {
            free(*one);
        }
        free(all);
    }
    static int impact_custom_remove(const fs_char *filename, void* userdata) {
        sound4::impact::dyn::CPresetLoader *preset_loader=reinterpret_cast<sound4::impact::dyn::CPresetLoader*>(userdata);
        auto res=preset_loader->Remove(filename);
        return res?0:-1;
    }
    static int impact_custom_rename(const fs_char *from, const fs_char *to, void* userdata) {
        sound4::impact::dyn::CPresetLoader *preset_loader=reinterpret_cast<sound4::impact::dyn::CPresetLoader*>(userdata);
        auto res=preset_loader->Rename(from,to);
        return res?0:-1;
    }

};
```

Index

~CClient
- sound4::impact::CInstance::CClient, 76
- sound4::impact::dyn::CInstance::CClient, 78

~CDynLoader
- sound4::impact::dyn::helper::CDynLoader, 92

~CInstance
- sound4::impact::CInstance, 96
- sound4::impact::dyn::CInstance, 106

~CPresetLoader
- sound4::impact::CPresetLoader, 115
- sound4::impact::dyn::CPresetLoader, 117

~MyPresetManager
- MyPresetManager, 122

AddAudio
- sound4::impact::CInstance, 96
- sound4::impact::dyn::CDynLib, 81
- sound4::impact::dyn::CInstance, 106

AddPadAudio
- sound4::impact::dyn::CDynLib, 81

Audio convert helpers, 46
- impact_AudioConvertFrom, 46
- impact_AudioConvertTo, 47
- impact_AudioMonoFromLiveStereo, 47
- impact_AudioMonoToLiveStereo, 48
- impact_MonoToStereo, 48
- impact_MonoToStereo_Planar, 48
- impact_StereoToMono, 49
- impact_StereoToMono_Planar, 49

Audio managing (any audio), 41
- F32_BE, 42
- F32_LE, 42
- F32_NATIVE, 42
- impact_AddAudio, 42
- impact_AddPadAudio, 42
- impact_GetAudio, 43
- impact_GetBytesFromFormat, 43
- impact_GetFormatFromName, 44
- impact_GetFormatName, 44
- impact_GetMaxPacketFrame, 44
- impact_GetOutputCount, 46
- impact_SampleFormat, 41
- INVALID_FORMAT, 42
- S16_BE, 42
- S16_LE, 42
- S16_NATIVE, 42
- S24_BE, 42
- S24_LE, 42
- S24_NATIVE, 42
- S32_BE, 42
- S32_LE, 42
- S32_NATIVE, 42

AudioConvertFrom
- sound4::impact::dyn::CDynLib, 81

AudioConvertTo
- sound4::impact::dyn::CDynLib, 82

AudioMonoFromLiveStereo
- sound4::impact::dyn::CDynLib, 82

AudioMonoToLiveStereo
- sound4::impact::dyn::CDynLib, 82

CClient
- sound4::impact::CInstance::CClient, 75
- sound4::impact::dyn::CInstance::CClient, 77

CDynLib
- sound4::impact::dyn::CDynLib, 80

CDynLoader
- sound4::impact::dyn::helper::CDynLoader, 92

change_kind_created
- Custom preset managagement, 34

change_kind_deleted
- Custom preset managagement, 34

change_kind_modified
- Custom preset managagement, 34

CHANNEL_COUNT
- sound4.impact.cl-pipe.cpp, 138

CHUNK_SIZE
- sound4.impact.cl-pipe.cpp, 139

ChunkFrames
- sound4::impact, 69
- sound4::impact::dyn, 72

CInstance
- sound4::impact::CInstance, 95, 96
- sound4::impact::dyn::CInstance, 105, 106

Client's JSon config interfacing, 50
- impact_DeleteClient, 50
- impact_FreeJsonAnswer, 51
- impact_NewClient, 51
- impact_ProcessJson, 51
- impact_SaveState, 52

Close
- sound4::impact::dyn::helper::CDynLoader, 92

CPresetLoader
- sound4::impact::CPresetLoader, 115
- sound4::impact::dyn::CPresetLoader, 117

Create
- sound4::impact::CInstance, 96
- sound4::impact::dyn::CInstance, 106

Custom preset managagement, 29
- change_kind_created, 34

change_kind_deleted, 34
change_kind_modified, 34
fs_char, 30
impact_PresetChange_Kind, 33
impact_PresetManager_InformChange, 34
impact_SetPresetManager, 34
impact_storage_exists, 30
impact_storage_getall, 31
impact_storage_getall_free, 31
impact_storage_reader, 31
impact_storage_reader_free, 32
impact_storage_remove, 32
impact_storage_rename, 32
impact_storage_writer, 33

DeleteClient
sound4::impact::dyn::CDynLib, 82
Direct processing, 38
impact_GetBufferIn, 38
impact_GetBufferOut, 39
impact_GetEstimatedDelay, 39
impact_ProcessAudio, 39
impact_ProcessAudio_Planar, 40
dyn, 67
DynFunc_t
sound4::impact::dyn::helper, 73
DynFuncHolder
sound4::impact::dyn::helper::DynFuncHolder< T >, 119
DynLib_t
sound4::impact::dyn::helper, 73

Environment variables and parameters, 17
error
Global setup, 21
Exists
MyPresetManager, 122
sound4::impact::CPresetLoader, 115
sound4::impact::dyn::CPresetLoader, 117
ExitProcess
sound4::impact::dyn::CDynLib, 82

F32_BE
Audio managing (any audio), 42
F32_LE
Audio managing (any audio), 42
F32_NATIVE
Audio managing (any audio), 42
fatal
Global setup, 21
format
sound4::impact::dyn::helper::SampleFormat< float >, 125
sound4::impact::dyn::helper::SampleFormat< int16_t >, 126
sound4::impact::dyn::helper::SampleFormat< int32_t >, 128
sound4::impact::helper::SampleFormat< float >, 126
sound4::impact::helper::SampleFormat< int16_t >, 127
sound4::impact::helper::SampleFormat< int32_t >, 128
FreeJsonAnswer
sound4::impact::dyn::CDynLib, 82
FreeMetadataInfos
sound4::impact::dyn::CDynLib, 83
FreeParameters
sound4::impact::dyn::CDynLib, 83
FreeParameterValue
sound4::impact::dyn::CDynLib, 83
fs_char
Custom preset managagement, 30
fs_strdup
sound4.impact.cl.hpp, 151
sound4.impact.cl_dyn.hpp, 160

GetAll
MyPresetManager, 122
sound4::impact::CPresetLoader, 115
sound4::impact::dyn::CPresetLoader, 117
GetAudio
sound4::impact::CInstance, 97
sound4::impact::dyn::CDynLib, 83
sound4::impact::dyn::CInstance, 107
GetAudioInputCount
sound4::impact::dyn::CDynLib, 83
GetAudioOutputCount
sound4::impact::dyn::CDynLib, 83
GetBufferIn
sound4::impact::CInstance, 97
sound4::impact::dyn::CDynLib, 84
sound4::impact::dyn::CInstance, 107
GetBufferOut
sound4::impact::CInstance, 97
sound4::impact::dyn::CDynLib, 84
sound4::impact::dyn::CInstance, 107
GetBytesFromFormat
sound4::impact::dyn::CDynLib, 84
GetChannelCount
sound4::impact::dyn::CDynLib, 84
GetChunkFrames
sound4::impact::CInstance, 98
sound4::impact::dyn::CInstance, 108
GetChunkSizeInFrames
sound4::impact::dyn::CDynLib, 84
GetEnv
sound4.impact.cl-pipe-cxx.cpp, 129
sound4.impact.cl-pipe-dyn.cpp, 133
GetEstimatedDelay
sound4::impact::CInstance, 98
sound4::impact::dyn::CDynLib, 84
sound4::impact::dyn::CInstance, 108
GetFormatFromName
sound4::impact::dyn::CDynLib, 85
GetFormatName
sound4::impact::dyn::CDynLib, 85
GetMaxPacketFrame

sound4::impact::dyn::CDynLib, 85
GetMetadataInfos
sound4::impact::CInstance, 98
sound4::impact::dyn::CDynLib, 85
sound4::impact::dyn::CInstance, 108
GetOutputCount
sound4::impact::dyn::CDynLib, 85
GetParam
sound4::impact::CInstance, 99
sound4::impact::dyn::CInstance, 109
GetParameter
sound4::impact::dyn::CDynLib, 85
GetPossibleChunkSizeInFrames
sound4::impact::dyn::CDynLib, 86
GetProcessChunkFrames
sound4::impact::dyn::CDynLib, 86
GetSampleRate
sound4::impact::dyn::CDynLib, 86
GetSymbol
sound4::impact::dyn::helper::CDynLoader, 93
GetSymbol_safe
sound4::impact::dyn::helper::CDynLoader, 93
GetThisLibraryPath
sound4::impact::dyn::helper::CDynLoader, 93
GetVersion
sound4::impact::dyn::CDynLib, 86
Global setup, 18
error, 21
fatal, 21
IMPACT_AUDIO_INPUT_COUNT, 19
IMPACT_AUDIO_OUTPUT_COUNT, 19
IMPACT_AUDIOFRAME_COUNT, 19
IMPACT_CHANNEL_COUNT, 19
impact_GetAudioInputCount, 21
impact_GetAudioOutputCount, 21
impact_GetChannelCount, 21
impact_GetChunkSizeInFrames, 22
impact_GetPossibleChunkSizeInFrames, 22
impact_GetSampleRate, 22
impact_GetVersion, 23
impact_loggerfn, 20
impact_LogSeverity, 20
IMPACT_SAMPLE_RATE, 20
impact_SetLoggerCallback, 23
impact_SetLogSeverity, 23
info, 21
none, 21
verbose, 21
verbose2, 21
verbose3, 21
verbose4, 21
verbose5, 21
warning, 21

impact_AddAudio
Audio managing (any audio), 42
impact_AddPadAudio
Audio managing (any audio), 42
IMPACT_AUDIO_INPUT_COUNT
Global setup, 19
IMPACT_AUDIO_OUTPUT_COUNT
Global setup, 19
impact_AudioConvertFrom
Audio convert helpers, 46
impact_AudioConvertTo
Audio convert helpers, 47
IMPACT_AUDIOFRAME_COUNT
Global setup, 19
impact_AudioMonoFromLiveStereo
Audio convert helpers, 47
impact_AudioMonoToLiveStereo
Audio convert helpers, 48
impact_CClientInstance, 120
IMPACT_CHANNEL_COUNT
Global setup, 19
impact_CInstance, 121
impact_CParameters, 121
IMPACT_DECL
sound4.impact.cl.h, 146
impact_DeleteClient
Client's JSon config interfacing, 50
impact_ExitProcess
Process creation/destruction, 24
impact_FreeJsonAnswer
Client's JSon config interfacing, 51
impact_FreeMetadataInfos
Metadata Management, 35
impact_FreeParameters
Process creation/destruction, 25
impact_FreeParameterValue
Process creation/destruction, 25
impact_GetAudio
Audio managing (any audio), 43
impact_GetAudioInputCount
Global setup, 21
impact_GetAudioOutputCount
Global setup, 21
impact_GetBufferIn
Direct processing, 38
impact_GetBufferOut
Direct processing, 39
impact_GetBytesFromFormat
Audio managing (any audio), 43
impact_GetChannelCount
Global setup, 21
impact_GetChunkSizeInFrames
Global setup, 22
impact_GetEstimatedDelay
Direct processing, 39
impact_GetFormatFromName
Audio managing (any audio), 44
impact_GetFormatName
Audio managing (any audio), 44
impact_GetMaxPacketFrame
Audio managing (any audio), 44
impact_GetMetadataInfos
Metadata Management, 35

impact_GetOutputCount
Audio managing (any audio), 46
impact_GetParameter
Process creation/destruction, 25
impact_GetPossibleChunkSizeInFrames
Global setup, 22
impact_GetProcessChunkFrames
Process creation/destruction, 25
impact_GetSampleRate
Global setup, 22
impact_GetVersion
Global setup, 23
IMPACT_HAS_CLOUDBUS
sound4.impact.cl.h, 146
IMPACT_HAS_WEBSERVER
sound4.impact.cl.h, 146
impact_InitProcess
Process creation/destruction, 26
impact_InitProcess2
Process creation/destruction, 26
impact_InitProcess3
Process creation/destruction, 26
impact_loggerfn
Global setup, 20
impact_LogSeverity
Global setup, 20
impact_MonoToStereo
Audio convert helpers, 48
impact_MonoToStereo_Planar
Audio convert helpers, 48
impact_NewClient
Client's JSon config interfacing, 51
impact_NewParameters
Process creation/destruction, 28
impact_PresetChange_Kind
Custom preset managagement, 33
impact_PresetManager_InformChange
Custom preset managagement, 34
impact_ProcessAudio
Direct processing, 39
impact_ProcessAudio_Planar
Direct processing, 40
impact_ProcessJson
Client's JSon config interfacing, 51
IMPACT_SAMPLE_RATE
Global setup, 20
impact_SampleFormat
Audio managing (any audio), 41
impact_SaveState
Client's JSon config interfacing, 52
impact_SetLoggerCallback
Global setup, 23
impact_SetLogSeverity
Global setup, 23
impact_SetMetadata
Metadata Management, 36
impact_SetParameter
Process creation/destruction, 28
impact_SetPresetManager
Custom preset managagement, 34
impact_StartUpdateThread
Update Management, 37
impact_StereoToMono
Audio convert helpers, 49
impact_StereoToMono_Planar
Audio convert helpers, 49
impact_StopUpdateThread
Update Management, 37
impact_storage_exists
Custom preset managagement, 30
impact_storage_getall
Custom preset managagement, 31
impact_storage_getall_free
Custom preset managagement, 31
impact_storage_reader
Custom preset managagement, 31
impact_storage_reader_free
Custom preset managagement, 32
impact_storage_remove
Custom preset managagement, 32
impact_storage_rename
Custom preset managagement, 32
impact_storage_writer
Custom preset managagement, 33
impact_TerminateProcess
Process creation/destruction, 28
impact_WaitUpdateThreadReady
Update Management, 37
impact_Webserver
Web server, 60
impact_Webserver_Status
Web server, 61
impact_Webserver_Stop
Web server, 61
impact_Webserver_tcp
Web server, 62
impact_Webserver_tcp2
Web server, 62
info
Global setup, 21
InitProcess
sound4::impact::dyn::CDynLib, 86
InitProcess2
sound4::impact::dyn::CDynLib, 86
InitProcess3
sound4::impact::dyn::CDynLib, 87
InputChannels
sound4::impact, 69
sound4::impact::dyn, 72
InputSampleSize
sound4::impact, 69
sound4::impact::dyn, 72
INVALID_FORMAT
Audio managing (any audio), 42
IsOk
sound4::impact::CInstance, 99

sound4::impact::dyn::CDynLib, 80
sound4::impact::dyn::CInstance, 109
sound4::impact::dyn::helper::CDynLoader, 93
sound4::impact::dyn::helper::DynFuncHolder< T >, 120
IsReadOnly
MyPresetManager, 122
sound4::impact::CPresetLoader, 115
sound4::impact::dyn::CPresetLoader, 117

JSON syntax, 52
JSON syntax for internal setup, 55

Load
sound4::impact::dyn::CDynLib, 81
sound4::impact::dyn::helper::CDynLoader, 93
log_cb_t
sound4::impact, 68
sound4::impact::dyn, 71
LogSeverity
sound4::impact, 68
sound4::impact::dyn, 71

main
sound4.impact.cl-pipe-cxx.cpp, 129
sound4.impact.cl-pipe-dyn.cpp, 133
sound4.impact.cl-pipe.cpp, 139
Metadata Management, 35
impact_FreeMetadataInfos, 35
impact_GetMetadataInfos, 35
impact_SetMetadata, 36
MonoToStereo
sound4::impact::dyn::CDynLib, 87
MonoToStereo_Planar
sound4::impact::dyn::CDynLib, 87
MyLogger
sound4.impact.cl-pipe-cxx.cpp, 130
sound4.impact.cl-pipe-dyn.cpp, 134
sound4.impact.cl-pipe.cpp, 139
MyPresetManager, 121
~MyPresetManager, 122
Exists, 122
GetAll, 122
IsReadOnly, 122
MyPresetManager, 122
Read, 123
Remove, 123
Rename, 123
SetupPath, 123
Write, 123

NewClient
sound4::impact::CInstance, 99
sound4::impact::dyn::CDynLib, 87
sound4::impact::dyn::CInstance, 109
NewParameters
sound4::impact::dyn::CDynLib, 87
none
Global setup, 21

OnUpdateThreadStart
sound4::impact::CInstance, 99
sound4::impact::dyn::CInstance, 109
OnUpdateThreadStop
sound4::impact::CInstance, 100
sound4::impact::dyn::CInstance, 110
operator bool
sound4::impact::dyn::helper::CDynLoader, 94
sound4::impact::dyn::helper::DynFuncHolder< T >, 120
operator T*
sound4::impact::dyn::helper::DynFuncHolder< T >, 120
operator=
sound4::impact::CInstance, 100
sound4::impact::CInstance::CClient, 76
sound4::impact::dyn::CInstance, 110
sound4::impact::dyn::CInstance::CClient, 78
sound4::impact::dyn::helper::CDynLoader, 94
OutputChannels
sound4::impact, 69
sound4::impact::dyn, 72
OutputSampleSize
sound4::impact, 70
sound4::impact::dyn, 72

PresetChange_Kind
sound4::impact, 68
sound4::impact::dyn, 71
PresetManager_InformChange
sound4::impact::CInstance, 100
sound4::impact::dyn::CDynLib, 87
sound4::impact::dyn::CInstance, 110
Process creation/destruction, 24
impact_ExitProcess, 24
impact_FreeParameters, 25
impact_FreeParameterValue, 25
impact_GetParameter, 25
impact_GetProcessChunkFrames, 25
impact_InitProcess, 26
impact_InitProcess2, 26
impact_InitProcess3, 26
impact_NewParameters, 28
impact_SetParameter, 28
impact_TerminateProcess, 28
ProcessAnyAudio
sound4::impact::CInstance, 100
sound4::impact::dyn::CInstance, 110
ProcessAudio
sound4::impact::CInstance, 101
sound4::impact::dyn::CDynLib, 88
sound4::impact::dyn::CInstance, 111
ProcessAudio_Planar
sound4::impact::CInstance, 101
sound4::impact::dyn::CDynLib, 88
sound4::impact::dyn::CInstance, 111
ProcessJson
sound4::impact::CInstance::CClient, 76
sound4::impact::dyn::CDynLib, 88

sound4::impact::dyn::CInstance::CClient, 78
ProcessLoop
sound4.impact.cl-pipe-cxx.cpp, 130
sound4.impact.cl-pipe-dyn.cpp, 134
sound4.impact.cl-pipe.cpp, 140

Read
MyPresetManager, 123
sound4::impact::CPresetLoader, 115
sound4::impact::dyn::CPresetLoader, 118
Remove
MyPresetManager, 123
sound4::impact::CPresetLoader, 116
sound4::impact::dyn::CPresetLoader, 118
Rename
MyPresetManager, 123
sound4::impact::CPresetLoader, 116
sound4::impact::dyn::CPresetLoader, 118

S16_BE
Audio managing (any audio), 42
S16_LE
Audio managing (any audio), 42
S16_NATIVE
Audio managing (any audio), 42
S24_BE
Audio managing (any audio), 42
S24_LE
Audio managing (any audio), 42
S24_NATIVE
Audio managing (any audio), 42
S32_BE
Audio managing (any audio), 42
S32_LE
Audio managing (any audio), 42
S32_NATIVE
Audio managing (any audio), 42
Sample using C interface, 15
Sample using C++ interface wrapper, 63
Sample using C++ interface wrapper with dynamic library loading, 64
SAMPLE_RATE
sound4.impact.cl-pipe.cpp, 139
SampleFormat
sound4::impact, 69
sound4::impact::dyn, 71
SampleRate
sound4::impact, 70
sound4::impact::dyn, 72
SaveState
sound4::impact::dyn::CDynLib, 88
SetLoggerCallback
sound4::impact::dyn::CDynLib, 88
SetLogSeverity
sound4::impact::dyn::CDynLib, 88
SetMetadata
sound4::impact::CInstance, 102
sound4::impact::dyn::CDynLib, 89
sound4::impact::dyn::CInstance, 112
SetParam
sound4::impact::CInstance, 102
sound4::impact::dyn::CInstance, 112
SetParameter
sound4::impact::dyn::CDynLib, 89
SetPresetManager
sound4::impact::CInstance, 102
sound4::impact::dyn::CDynLib, 89
sound4::impact::dyn::CInstance, 112
SetupPath
MyPresetManager, 123
sound4, 67
sound4.impact.cl-pipe-cxx.cpp, 129, 131
GetEnv, 129
main, 129
MyLogger, 130
ProcessLoop, 130
Usage, 130
sound4.impact.cl-pipe-dyn.cpp, 132, 135
GetEnv, 133
main, 133
MyLogger, 134
ProcessLoop, 134
Usage, 134
sound4.impact.cl-pipe.cpp, 138, 140
CHANNEL_COUNT, 138
CHUNK_SIZE, 139
main, 139
MyLogger, 139
ProcessLoop, 140
SAMPLE_RATE, 139
Usage, 140
sound4.impact.cl.h, 143, 146
IMPACT_DECL, 146
IMPACT_HAS_CLOUDBUS, 146
IMPACT_HAS_WEBSERVER, 146
sound4.impact.cl.hpp, 150, 152
fs_strdup, 151
sound4.impact.cl_dyn.hpp, 159, 160
fs_strdup, 160
sound4::impact, 67
ChunkFrames, 69
InputChannels, 69
InputSampleSize, 69
log_cb_t, 68
LogSeverity, 68
OutputChannels, 69
OutputSampleSize, 70
PresetChange_Kind, 68
SampleFormat, 69
SampleRate, 70
sound4::impact::CInstance, 94
~CInstance, 96
AddAudio, 96
CInstance, 95, 96
Create, 96
GetAudio, 97
GetBufferIn, 97

GetBufferOut, 97
GetChunkFrames, 98
GetEstimatedDelay, 98
GetMetadataInfos, 98
GetParam, 99
IsOk, 99
NewClient, 99
OnUpdateThreadStart, 99
OnUpdateThreadStop, 100
operator=, 100
PresetManager_InformChange, 100
ProcessAnyAudio, 100
ProcessAudio, 101
ProcessAudio_Planar, 101
SetMetadata, 102
SetParam, 102
SetPresetManager, 102
StartWebServer, 103
Stop, 103
StopWebServer, 103
Terminate, 104
sound4::impact::CInstance::CClient, 75
~CClient, 76
CClient, 75
operator=, 76
ProcessJson, 76
sound4::impact::CPresetLoader, 114
~CPresetLoader, 115
CPresetLoader, 115
Exists, 115
GetAll, 115
IsReadOnly, 115
Read, 115
Remove, 116
Rename, 116
Write, 116
sound4::impact::dyn, 70
ChunkFrames, 72
InputChannels, 72
InputSampleSize, 72
log_cb_t, 71
LogSeverity, 71
OutputChannels, 72
OutputSampleSize, 72
PresetChange_Kind, 71
SampleFormat, 71
SampleRate, 72
sound4::impact::dyn::CDynLib, 79
AddAudio, 81
AddPadAudio, 81
AudioConvertFrom, 81
AudioConvertTo, 82
AudioMonoFromLiveStereo, 82
AudioMonoToLiveStereo, 82
CDynLib, 80
DeleteClient, 82
ExitProcess, 82
FreeJsonAnswer, 82
FreeMetadataInfos, 83
FreeParameters, 83
FreeParameterValue, 83
GetAudio, 83
GetAudioInputCount, 83
GetAudioOutputCount, 83
GetBufferIn, 84
GetBufferOut, 84
GetBytesFromFormat, 84
GetChannelCount, 84
GetChunkSizeInFrames, 84
GetEstimatedDelay, 84
GetFormatFromName, 85
GetFormatName, 85
GetMaxPacketFrame, 85
GetMetadataInfos, 85
GetOutputCount, 85
GetParameter, 85
GetPossibleChunkSizeInFrames, 86
GetProcessChunkFrames, 86
GetSampleRate, 86
GetVersion, 86
InitProcess, 86
InitProcess2, 86
InitProcess3, 87
IsOk, 80
Load, 81
MonoToStereo, 87
MonoToStereo_Planar, 87
NewClient, 87
NewParameters, 87
PresetManager_InformChange, 87
ProcessAudio, 88
ProcessAudio_Planar, 88
ProcessJson, 88
SaveState, 88
SetLoggerCallback, 88
SetLogSeverity, 88
SetMetadata, 89
SetParameter, 89
SetPresetManager, 89
StartUpdateThread, 89
StereoToMono, 89
StereoToMono_Planar, 89
StopUpdateThread, 90
TerminateProcess, 90
WaitUpdateThreadReady, 90
Webserver, 90
Webserver_Status, 90
Webserver_Stop, 90
Webserver_tcp, 91
Webserver_tcp2, 91
sound4::impact::dyn::CInstance, 104
~CInstance, 106
AddAudio, 106
CInstance, 105, 106
Create, 106
GetAudio, 107

GetBufferIn, 107
GetBufferOut, 107
GetChunkFrames, 108
GetEstimatedDelay, 108
GetMetadataInfos, 108
GetParam, 109
IsOk, 109
NewClient, 109
OnUpdateThreadStart, 109
OnUpdateThreadStop, 110
operator=, 110
PresetManager_InformChange, 110
ProcessAnyAudio, 110
ProcessAudio, 111
ProcessAudio_Planar, 111
SetMetadata, 112
SetParam, 112
SetPresetManager, 112
StartWebServer, 113
Stop, 113
StopWebServer, 113
Terminate, 114
sound4::impact::dyn::CInstance::CClient, 77
~CClient, 78
CClient, 77
operator=, 78
ProcessJson, 78
sound4::impact::dyn::CPresetLoader, 116
~CPresetLoader, 117
CPresetLoader, 117
Exists, 117
GetAll, 117
IsReadOnly, 117
Read, 118
Remove, 118
Rename, 118
Write, 118
sound4::impact::dyn::helper, 73
DynFunc_t, 73
DynLib_t, 73
sound4::impact::dyn::helper::CDynLoader, 91
~CDynLoader, 92
CDynLoader, 92
Close, 92
GetSymbol, 93
GetSymbol_safe, 93
GetThisLibraryPath, 93
IsOk, 93
Load, 93
operator bool, 94
operator=, 94
sound4::impact::dyn::helper::DynFuncHolder< T >, 119
DynFuncHolder, 119
IsOk, 120
operator bool, 120
operator T*, 120
sound4::impact::dyn::helper::SampleFormat< float >, 125
format, 125
sound4::impact::dyn::helper::SampleFormat< int16_t >, 126
format, 126
sound4::impact::dyn::helper::SampleFormat< int32_t >, 127
format, 128
sound4::impact::dyn::helper::SampleFormat< T >, 124
sound4::impact::helper, 74
sound4::impact::helper::SampleFormat< float >, 125
format, 126
sound4::impact::helper::SampleFormat< int16_t >, 127
format, 127
sound4::impact::helper::SampleFormat< int32_t >, 128
format, 128
sound4::impact::helper::SampleFormat< T >, 124
SOUND4_INVALID_WEBSERVER_ID
Web server, 59
SOUND4_WEBSERVER_HTTP_OK
Web server, 59
SOUND4_WEBSERVER_HTTPS_OK
Web server, 60
StartUpdateThread
sound4::impact::dyn::CDynLib, 89
StartWebServer
sound4::impact::CInstance, 103
sound4::impact::dyn::CInstance, 113
std, 74
StereoToMono
sound4::impact::dyn::CDynLib, 89
StereoToMono_Planar
sound4::impact::dyn::CDynLib, 89
Stop
sound4::impact::CInstance, 103
sound4::impact::dyn::CInstance, 113
StopUpdateThread
sound4::impact::dyn::CDynLib, 90
StopWebServer
sound4::impact::CInstance, 103
sound4::impact::dyn::CInstance, 113

Terminate
sound4::impact::CInstance, 104
sound4::impact::dyn::CInstance, 114
TerminateProcess
sound4::impact::dyn::CDynLib, 90

Update Management, 36
impact_StartUpdateThread, 37
impact_StopUpdateThread, 37
impact_WaitUpdateThreadReady, 37
Usage
sound4.impact.cl-pipe-cxx.cpp, 130
sound4.impact.cl-pipe-dyn.cpp, 134
sound4.impact.cl-pipe.cpp, 140

verbose

Global setup, 21
verbose2
Global setup, 21
verbose3
Global setup, 21
verbose4
Global setup, 21
verbose5
Global setup, 21

WaitUpdateThreadReady
sound4::impact::dyn::CDynLib, 90
warning
Global setup, 21
Web server, 59
impact_Webserver, 60
impact_Webserver_Status, 61
impact_Webserver_Stop, 61
impact_Webserver_tcp, 62
impact_Webserver_tcp2, 62
SOUND4_INVALID_WEBSERVER_ID, 59
SOUND4_WEBSERVER_HTTP_OK, 59
SOUND4_WEBSERVER_HTTPS_OK, 60
Webserver
sound4::impact::dyn::CDynLib, 90
Webserver_Status
sound4::impact::dyn::CDynLib, 90
Webserver_Stop
sound4::impact::dyn::CDynLib, 90
Webserver_tcp
sound4::impact::dyn::CDynLib, 91
Webserver_tcp2
sound4::impact::dyn::CDynLib, 91
Write
MyPresetManager, 123
sound4::impact::CPresetLoader, 116
sound4::impact::dyn::CPresetLoader, 118