SOUND4 BIGVOICE.CL Library [1.1.5]

Generated by Doxygen 1.9.8

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

2 Deprecated List 3

3 Topic Index 5
3.1 Topics . . . 5

4 Namespace Index 7
4.1 Namespace List . . . 7

5 Hierarchical Index 9
5.1 Class Hierarchy . . . 9

6 Class Index 13
6.1 Class List . . . 13

7 File Index 15
7.1 File List . . . 15

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

8.3.5 Function Documentation . . . 23
8.3.5.1 bigvoice_GetAudioInputCount() . . . 23
8.3.5.2 bigvoice_GetAudioOutputCount() . . . 23
8.3.5.3 bigvoice_GetChannelCount() . . . 24
8.3.5.4 bigvoice_GetChunkSizeInFrames() . . . 24
8.3.5.5 bigvoice_GetPossibleChunkSizeInFrames() . . . 24
8.3.5.6 bigvoice_GetSampleRate() . . . 25
8.3.5.7 bigvoice_GetVersion() . . . 25
8.3.5.8 bigvoice_SetLoggerCallback() . . . 25
8.3.5.9 bigvoice_SetLogSeverity() . . . 25
8.3.5.10 SOUND4_GetProcessInfo() . . . 27
8.4 Process creation/destruction . . . 27
8.4.1 Detailed Description . . . 27
8.4.2 Function Documentation . . . 27
8.4.2.1 bigvoice_ExitProcess() . . . 27
8.4.2.2 bigvoice_FreeParameters() . . . 28
8.4.2.3 bigvoice_FreeParameterValue() . . . 28
8.4.2.4 bigvoice_GetParameter() . . . 28
8.4.2.5 bigvoice_GetProcessChunkFrames() . . . 28
8.4.2.6 bigvoice_InitProcess() . . . 29
8.4.2.7 bigvoice_InitProcess2() . . . 29
8.4.2.8 bigvoice_InitProcess3() . . . 30
8.4.2.9 bigvoice_NewParameters() . . . 31
8.4.2.10 bigvoice_SetParameter() . . . 31
8.4.2.11 bigvoice_TerminateProcess() . . . 31
8.5 Instance's bus . . . 32
8.5.1 Detailed Description . . . 32
8.5.2 Function Documentation . . . 32
8.5.2.1 bigvoice_FreeBus() . . . 32
8.5.2.2 bigvoice_NewBus() . . . 32
8.5.2.3 bigvoice_SetInstanceBus() . . . 33
8.6 Custom preset managagement . . . 33
8.6.1 Detailed Description . . . 34
8.6.2 Typedef Documentation . . . 34
8.6.2.1 bigvoice_storage_exists . . . 34
8.6.2.2 bigvoice_storage_getall . . . 34
8.6.2.3 bigvoice_storage_getall_free . . . 35
8.6.2.4 bigvoice_storage_reader . . . 35
8.6.2.5 bigvoice_storage_reader_free . . . 35
8.6.2.6 bigvoice_storage_remove . . . 36
8.6.2.7 bigvoice_storage_rename . . . 36
8.6.2.8 bigvoice_storage_writer . . . 36

8.6.2.9 fs_char . . . 37
8.6.3 Enumeration Type Documentation . . . 37
8.6.3.1 bigvoice_PresetChange_Kind . . . 37
8.6.4 Function Documentation . . . 37
8.6.4.1 bigvoice_PresetManager_InformChange() . . . 37
8.6.4.2 bigvoice_SetPresetManager() . . . 38
8.7 Metadata Management . . . 38
8.7.1 Detailed Description . . . 39
8.7.2 Function Documentation . . . 39
8.7.2.1 bigvoice_FreeMetadataInfos() . . . 39
8.7.2.2 bigvoice_GetMetadataInfos() . . . 39
8.7.2.3 bigvoice_SetMetadata() . . . 39
8.7.2.4 bigvoice_SetMetadataMulti() . . . 40
8.8 Update Management . . . 40
8.8.1 Detailed Description . . . 40
8.8.2 Function Documentation . . . 41
8.8.2.1 bigvoice_StartUpdateThread() . . . 41
8.8.2.2 bigvoice_StopUpdateThread() . . . 41
8.8.2.3 bigvoice_WaitUpdateThreadReady() . . . 41
8.9 Direct processing . . . 42
8.9.1 Detailed Description . . . 42
8.9.2 Function Documentation . . . 42
8.9.2.1 bigvoice_GetBufferIn() . . . 42
8.9.2.2 bigvoice_GetBufferOut() . . . 42
8.9.2.3 bigvoice_GetEstimatedDelay() . . . 43
8.9.2.4 bigvoice_ProcessAudio() . . . 43
8.9.2.5 bigvoice_ProcessAudio_Planar() . . . 44
8.10 Audio managing (any audio) . . . 44
8.10.1 Detailed Description . . . 45
8.10.2 Enumeration Type Documentation . . . 45
8.10.2.1 bigvoice_SampleFormat . . . 45
8.10.3 Function Documentation . . . 45
8.10.3.1 bigvoice_AddAudio() . . . 45
8.10.3.2 bigvoice_AddPadAudio() . . . 46
8.10.3.3 bigvoice_GetAudio() . . . 46
8.10.3.4 bigvoice_GetBytesFromFormat() . . . 47
8.10.3.5 bigvoice_GetFormatFromName() . . . 47
8.10.3.6 bigvoice_GetFormatName() . . . 47
8.10.3.7 bigvoice_GetMaxPacketFrame() . . . 48
8.10.3.8 bigvoice_GetOutputCount() . . . 48
8.11 Audio convert helpers . . . 48
8.11.1 Detailed Description . . . 49

8.11.2 Function Documentation . . . 49
8.11.2.1 bigvoice_AudioConvertFrom() . . . 49
8.11.2.2 bigvoice_AudioConvertTo() . . . 49
8.11.2.3 bigvoice_AudioMonoFromLiveStereo() . . . 50
8.11.2.4 bigvoice_AudioMonoToLiveStereo() . . . 50
8.11.2.5 bigvoice_MonoToStereo() . . . 50
8.11.2.6 bigvoice_MonoToStereo_Planar() . . . 51
8.11.2.7 bigvoice_StereoToMono() . . . 51
8.11.2.8 bigvoice_StereoToMono_Planar() . . . 51
8.12 Client's JSon config interfacing . . . 52
8.12.1 Detailed Description . . . 52
8.12.2 Function Documentation . . . 52
8.12.2.1 bigvoice_DeleteClient() . . . 52
8.12.2.2 bigvoice_FreeJsonAnswer() . . . 53
8.12.2.3 bigvoice_NewClient() . . . 53
8.12.2.4 bigvoice_ProcessJson() . . . 53
8.12.2.5 bigvoice_SaveState() . . . 54
8.13 JSON syntax . . . 54
8.13.1 Commands . . . 54
8.13.2 Answers . . . 55
8.13.3 Examples . . . 55
8.13.3.1 Get status . . . 55
8.13.3.2 Get all changes since last request . . . 56
8.13.3.3 Check license . . . 56
8.13.3.4 Set Processor name . . . 56
8.13.3.5 Get presetlist . . . 56
8.13.3.6 Load a preset . . . 56
8.13.3.7 Get information on preset on air . . . 56
8.13.3.8 Set input gain . . . 56
8.13.3.9 Set bypass . . . 57
8.13.3.10 Export a preset . . . 57
8.13.3.11 Import a preset . . . 57
8.14 JSON syntax for internal setup . . . 57
8.14.1 Get process and modules information . . . 57
8.14.2 Authentication management . . . 58
8.14.2.1 Difference between setup and admin account . . . 58
8.14.2.2 Access encoding . . . 58
8.14.2.3 Change setup account . . . 58
8.14.2.4 Get user info . . . 59
8.14.2.5 Add or modify a user . . . 59
8.14.2.6 Removes a user . . . 59
8.14.2.7 List users . . . 59

8.14.2.8 Get/set unlogged access . . . 59
8.14.2.9 Get realm . . . 59
8.14.2.10 Get API version . . . 59
8.14.2.11 Get setup is admin . . . 60
8.14.2.12 Get/set setup account . . . 60
8.14.2.13 Get/set custom storage . . . 60
8.14.3 Get process' interface description . . . 60
8.14.3.1 Get XML decription . . . 60
8.14.3.2 Get Process Image . . . 60
8.14.3.3 Get Process Ico . . . 60
8.15 Web server . . . 61
8.15.1 Detailed Description . . . 61
8.15.2 Macro Definition Documentation . . . 61
8.15.2.1 SOUND4_INVALID_WEBSERVER_ID . . . 61
8.15.2.2 SOUND4_WEBSERVER_HTTP_OK . . . 62
8.15.2.3 SOUND4_WEBSERVER_HTTPS_OK . . . 62
8.15.3 Function Documentation . . . 62
8.15.3.1 bigvoice_Webserver() . . . 62
8.15.3.2 bigvoice_Webserver_FreeString() . . . 63
8.15.3.3 bigvoice_Webserver_GetAppHealth() . . . 63
8.15.3.4 bigvoice_Webserver_SetAppHealth() . . . 63
8.15.3.5 bigvoice_Webserver_Status() . . . 64
8.15.3.6 bigvoice_Webserver_Stop() . . . 64
8.15.3.7 bigvoice_Webserver_tcp() . . . 64
8.15.3.8 bigvoice_Webserver_tcp2() . . . 65
8.16 Sample using C++ interface wrapper . . . 66
8.16.1 Setup and sanity checks . . . 66
8.16.2 Sample to create the processing instance . . . 66
8.16.3 Communicating with the instance . . . 66
8.16.4 Starting a web-server . . . 66
8.16.5 Sample processing loop . . . 67
8.17 Sample using C++ interface wrapper with dynamic library loading . . . 67
8.17.1 Dynamic loading . . . 67
8.17.2 Setup and sanity checks . . . 67
8.17.3 Sample to create the processing instance . . . 67
8.17.4 Communicating with the instance . . . 67
8.17.5 Starting a web-server . . . 67
8.17.6 Sample processing loop . . . 68
8.17.7 Custom preset manager . . . 68
8.17.8 Custom preset manager loading . . . 69
8.18 Sample using Generic C++ interface (multi-process compatible) . . . 69
8.18.1 Loading the library . . . 69

8.18.2 Setup and configure . . . 69
8.18.3 Create the processing instance . . . 69
8.18.4 Processing loop . . . 70
8.18.5 Starting a web-server . . . 70
8.18.6 Communicating with the instance . . . 70
8.18.7 Custom preset manager example . . . 70
8.18.8 Custom preset manager loading . . . 71

9 Namespace Documentation **73**
9.1 dyn Namespace Reference . . . 73
9.1.1 Detailed Description . . . 73
9.2 sound4 Namespace Reference . . . 73
9.2.1 Detailed Description . . . 74
9.2.2 Typedef Documentation . . . 74
9.2.2.1 log_cb_t . . . 74
9.2.2.2 LogSeverity . . . 74
9.2.2.3 PresetChange_Kind . . . 74
9.2.3 Variable Documentation . . . 74
9.2.3.1 error . . . 74
9.2.3.2 fatal . . . 75
9.2.3.3 info . . . 75
9.2.3.4 none . . . 75
9.2.3.5 verbose . . . 75
9.2.3.6 verbose2 . . . 75
9.2.3.7 verbose3 . . . 75
9.2.3.8 verbose4 . . . 75
9.2.3.9 verbose5 . . . 75
9.2.3.10 warning . . . 76
9.3 sound4::bigvoice Namespace Reference . . . 76
9.3.1 Detailed Description . . . 76
9.3.2 Typedef Documentation . . . 76
9.3.2.1 log_cb_t . . . 76
9.3.2.2 LogSeverity . . . 77
9.3.2.3 PresetChange_Kind . . . 77
9.3.2.4 SampleFormat . . . 77
9.3.3 Variable Documentation . . . 77
9.3.3.1 ChunkFrames . . . 77
9.3.3.2 InputChannels . . . 77
9.3.3.3 InputSampleSize . . . 77
9.3.3.4 OutputChannels . . . 78
9.3.3.5 OutputSampleSize . . . 78
9.3.3.6 SampleRate . . . 78

9.4 sound4::bigvoice::dyn Namespace Reference . . . 78
9.4.1 Typedef Documentation . . . 79
9.4.1.1 log_cb_t . . . 79
9.4.1.2 LogSeverity . . . 79
9.4.1.3 PresetChange_Kind . . . 79
9.4.1.4 SampleFormat . . . 79
9.4.2 Variable Documentation . . . 79
9.4.2.1 ChunkFrames . . . 79
9.4.2.2 InputChannels . . . 80
9.4.2.3 InputSampleSize . . . 80
9.4.2.4 OutputChannels . . . 80
9.4.2.5 OutputSampleSize . . . 80
9.4.2.6 SampleRate . . . 80
9.5 sound4::bigvoice::dyn::helper Namespace Reference . . . 80
9.5.1 Detailed Description . . . 81
9.5.2 Typedef Documentation . . . 81
9.5.2.1 DynFunc_t . . . 81
9.5.2.2 DynLib_t . . . 81
9.6 sound4::bigvoice::helper Namespace Reference . . . 81
9.6.1 Detailed Description . . . 81
9.7 sound4::helper Namespace Reference . . . 81
9.7.1 Detailed Description . . . 82
9.7.2 Typedef Documentation . . . 82
9.7.2.1 DynFunc_t . . . 82
9.7.2.2 DynLib_t . . . 82
9.8 std Namespace Reference . . . 82

10 Class Documentation **83**

10.1 bigvoice_CBus Struct Reference . . . 83
10.1.1 Detailed Description . . . 83
10.2 bigvoice_CClientInstance Struct Reference . . . 83
10.2.1 Detailed Description . . . 83
10.3 bigvoice_CInstance Struct Reference . . . 83
10.3.1 Detailed Description . . . 84
10.4 bigvoice_CParameters Struct Reference . . . 84
10.4.1 Detailed Description . . . 84
10.5 sound4::bigvoice::CBus Class Reference . . . 84
10.5.1 Detailed Description . . . 84
10.5.2 Constructor & Destructor Documentation . . . 84
10.5.2.1 CBus() . . . 84
10.5.2.2 ~CBus() . . . 85
10.5.3 Member Function Documentation . . . 85

10.5.3.1 Get() . . . 85
10.6 sound4::bigvoice::dyn::CBus Class Reference . . . 85
10.6.1 Detailed Description . . . 85
10.6.2 Constructor & Destructor Documentation . . . 85
10.6.2.1 CBus() . . . 85
10.6.2.2 ~CBus() . . . 85
10.6.3 Member Function Documentation . . . 86
10.6.3.1 Get() . . . 86
10.7 sound4::CBus Class Reference . . . 86
10.7.1 Detailed Description . . . 86
10.7.2 Constructor & Destructor Documentation . . . 86
10.7.2.1 CBus() . . . 86
10.7.2.2 ~CBus() . . . 86
10.7.3 Member Function Documentation . . . 87
10.7.3.1 Get() . . . 87
10.8 sound4::bigvoice::CInstance::CClient Class Reference . . . 87
10.8.1 Detailed Description . . . 87
10.8.2 Constructor & Destructor Documentation . . . 87
10.8.2.1 CClient() [1/2] . . . 87
10.8.2.2 CClient() [2/2] . . . 88
10.8.2.3 ~CClient() . . . 88
10.8.3 Member Function Documentation . . . 88
10.8.3.1 operator=() . . . 88
10.8.3.2 ProcessJson() . . . 88
10.9 sound4::bigvoice::dyn::CInstance::CClient Class Reference . . . 89
10.9.1 Detailed Description . . . 89
10.9.2 Constructor & Destructor Documentation . . . 89
10.9.2.1 CClient() [1/2] . . . 89
10.9.2.2 CClient() [2/2] . . . 89
10.9.2.3 ~CClient() . . . 89
10.9.3 Member Function Documentation . . . 89
10.9.3.1 operator=() . . . 89
10.9.3.2 ProcessJson() . . . 89
10.10 sound4::CInstance::CClient Class Reference . . . 90
10.10.1 Detailed Description . . . 90
10.10.2 Constructor & Destructor Documentation . . . 91
10.10.2.1 CClient() [1/2] . . . 91
10.10.2.2 CClient() [2/2] . . . 91
10.10.2.3 ~CClient() . . . 91
10.10.3 Member Function Documentation . . . 91
10.10.3.1 operator=() . . . 91
10.10.3.2 ProcessJson() . . . 91

10.11 sound4::bigvoice::dyn::CDynLib Class Reference 92
10.11.1 Detailed Description 93
10.11.2 Constructor & Destructor Documentation 93
10.11.2.1 CDynLib() 93
10.11.3 Member Function Documentation 94
10.11.3.1 IsOk() 94
10.11.3.2 Load() 94
10.11.4 Member Data Documentation 94
10.11.4.1 AddAudio 94
10.11.4.2 AddPadAudio 94
10.11.4.3 AudioConvertFrom 95
10.11.4.4 AudioConvertTo 95
10.11.4.5 AudioMonoFromLiveStereo 95
10.11.4.6 AudioMonoToLiveStereo 95
10.11.4.7 DeleteClient 95
10.11.4.8 ExitProcess 95
10.11.4.9 FreeBus 95
10.11.4.10 FreeJsonAnswer 96
10.11.4.11 FreeMetadataInfos 96
10.11.4.12 FreeParameters 96
10.11.4.13 FreeParameterValue 96
10.11.4.14 GetAudio 96
10.11.4.15 GetAudioInputCount 96
10.11.4.16 GetAudioOutputCount 96
10.11.4.17 GetBufferIn 97
10.11.4.18 GetBufferOut 97
10.11.4.19 GetBytesFromFormat 97
10.11.4.20 GetChannelCount 97
10.11.4.21 GetChunkSizeInFrames 97
10.11.4.22 GetEstimatedDelay 97
10.11.4.23 GetFormatFromName 97
10.11.4.24 GetFormatName 98
10.11.4.25 GetMaxPacketFrame 98
10.11.4.26 GetMetadataInfos 98
10.11.4.27 GetOutputCount 98
10.11.4.28 GetParameter 98
10.11.4.29 GetPossibleChunkSizeInFrames 98
10.11.4.30 GetProcessChunkFrames 98
10.11.4.31 GetSampleRate 99
10.11.4.32 GetVersion 99
10.11.4.33 InitProcess 99
10.11.4.34 InitProcess2 99

10.11.4.35 InitProcess3 . . . 99
10.11.4.36 MonoToStereo . . . 99
10.11.4.37 MonoToStereo_Planar . . . 99
10.11.4.38 NewBus . . . 100
10.11.4.39 NewClient . . . 100
10.11.4.40 NewParameters . . . 100
10.11.4.41 PresetManager_InformChange . . . 100
10.11.4.42 ProcessAudio . . . 100
10.11.4.43 ProcessAudio_Planar . . . 100
10.11.4.44 ProcessJson . . . 100
10.11.4.45 SaveState . . . 101
10.11.4.46 SetInstanceBus . . . 101
10.11.4.47 SetLoggerCallback . . . 101
10.11.4.48 SetLogSeverity . . . 101
10.11.4.49 SetMetadata . . . 101
10.11.4.50 SetMetadataMulti . . . 101
10.11.4.51 SetParameter . . . 101
10.11.4.52 SetPresetManager . . . 102
10.11.4.53 StartUpdateThread . . . 102
10.11.4.54 StereoToMono . . . 102
10.11.4.55 StereoToMono_Planar . . . 102
10.11.4.56 StopUpdateThread . . . 102
10.11.4.57 TerminateProcess . . . 102
10.11.4.58 WaitUpdateThreadReady . . . 102
10.11.4.59 Webserver . . . 103
10.11.4.60 Webserver_FreeString . . . 103
10.11.4.61 Webserver_GetAppHealth . . . 103
10.11.4.62 Webserver_SetAppHealth . . . 103
10.11.4.63 Webserver_Status . . . 103
10.11.4.64 Webserver_Stop . . . 103
10.11.4.65 Webserver_tcp . . . 103
10.11.4.66 Webserver_tcp2 . . . 104
10.12 sound4::bigvoice::dyn::helper::CDynLoader Class Reference . . . 104
10.12.1 Detailed Description . . . 104
10.12.2 Constructor & Destructor Documentation . . . 104
10.12.2.1 CDynLoader() [1/3] . . . 104
10.12.2.2 CDynLoader() [2/3] . . . 105
10.12.2.3 CDynLoader() [3/3] . . . 105
10.12.2.4 ~CDynLoader() . . . 105
10.12.3 Member Function Documentation . . . 105
10.12.3.1 Close() . . . 105
10.12.3.2 GetSymbol() . . . 105

10.12.3.3 GetSymbol_safe() . . . 105
10.12.3.4 GetThisLibraryPath() . . . 105
10.12.3.5 IsOk() . . . 106
10.12.3.6 Load() . . . 106
10.12.3.7 operator bool() . . . 106
10.12.3.8 operator=() . . . 106
10.13 sound4::helper::CDynLoader Class Reference . . . 106
10.13.1 Detailed Description . . . 107
10.13.2 Constructor & Destructor Documentation . . . 107
10.13.2.1 CDynLoader() [1/3] . . . 107
10.13.2.2 CDynLoader() [2/3] . . . 107
10.13.2.3 CDynLoader() [3/3] . . . 107
10.13.2.4 ~CDynLoader() . . . 107
10.13.3 Member Function Documentation . . . 107
10.13.3.1 Close() . . . 107
10.13.3.2 GetSymbol() . . . 107
10.13.3.3 GetSymbol_safe() . . . 108
10.13.3.4 GetThisLibraryPath() . . . 108
10.13.3.5 IsOk() . . . 108
10.13.3.6 Load() . . . 108
10.13.3.7 operator bool() . . . 108
10.13.3.8 operator=() . . . 108
10.14 sound4::bigvoice::CInstance Class Reference . . . 109
10.14.1 Detailed Description . . . 110
10.14.2 Constructor & Destructor Documentation . . . 110
10.14.2.1 CInstance() [1/2] . . . 110
10.14.2.2 CInstance() [2/2] . . . 110
10.14.2.3 ~CInstance() . . . 110
10.14.3 Member Function Documentation . . . 110
10.14.3.1 AddAudio() . . . 110
10.14.3.2 Create() . . . 110
10.14.3.3 GetAudio() . . . 111
10.14.3.4 GetBufferIn() . . . 111
10.14.3.5 GetBufferOut() . . . 111
10.14.3.6 GetChunkFrames() . . . 112
10.14.3.7 GetEstimatedDelay() . . . 112
10.14.3.8 GetMetadataInfos() . . . 112
10.14.3.9 GetParam() . . . 112
10.14.3.10 GetWebServerAppHealth() . . . 113
10.14.3.11 IsOk() . . . 113
10.14.3.12 NewClient() . . . 113
10.14.3.13 OnUpdateThreadStart() . . . 113

10.14.3.14 OnUpdateThreadStop() . . . 113
10.14.3.15 operator=() . . . 113
10.14.3.16 PresetManager_InformChange() . . . 113
10.14.3.17 ProcessAnyAudio() . . . 114
10.14.3.18 ProcessAudio() . . . 114
10.14.3.19 ProcessAudio_Planar() . . . 115
10.14.3.20 SetBus() . . . 115
10.14.3.21 SetMetadata() . . . 115
10.14.3.22 SetMetadataMulti() . . . 116
10.14.3.23 SetParam() . . . 116
10.14.3.24 SetPresetManager() . . . 116
10.14.3.25 SetWebServerAppHealth() . . . 117
10.14.3.26 StartWebServer() . . . 117
10.14.3.27 Stop() . . . 117
10.14.3.28 StopWebServer() . . . 118
10.14.3.29 Terminate() . . . 118
10.15 sound4::bigvoice::dyn::CInstance Class Reference . . . 118
10.15.1 Detailed Description . . . 119
10.15.2 Constructor & Destructor Documentation . . . 120
10.15.2.1 CInstance() [1/2] . . . 120
10.15.2.2 CInstance() [2/2] . . . 120
10.15.2.3 ∼CInstance() . . . 120
10.15.3 Member Function Documentation . . . 120
10.15.3.1 AddAudio() . . . 120
10.15.3.2 Create() . . . 120
10.15.3.3 GetAudio() . . . 121
10.15.3.4 GetBufferIn() . . . 121
10.15.3.5 GetBufferOut() . . . 121
10.15.3.6 GetChunkFrames() . . . 122
10.15.3.7 GetEstimatedDelay() . . . 122
10.15.3.8 GetMetadataInfos() . . . 122
10.15.3.9 GetParam() . . . 122
10.15.3.10 GetWebServerAppHealth() . . . 123
10.15.3.11 IsOk() . . . 123
10.15.3.12 NewClient() . . . 123
10.15.3.13 OnUpdateThreadStart() . . . 123
10.15.3.14 OnUpdateThreadStop() . . . 123
10.15.3.15 operator=() . . . 123
10.15.3.16 PresetManager_InformChange() . . . 123
10.15.3.17 ProcessAnyAudio() . . . 124
10.15.3.18 ProcessAudio() . . . 124
10.15.3.19 ProcessAudio_Planar() . . . 125

10.15.3.20 SetBus() . . . 125
10.15.3.21 SetMetadata() . . . 125
10.15.3.22 SetMetadataMulti() . . . 126
10.15.3.23 SetParam() . . . 126
10.15.3.24 SetPresetManager() . . . 126
10.15.3.25 SetWebServerAppHealth() . . . 127
10.15.3.26 StartWebServer() . . . 127
10.15.3.27 Stop() . . . 127
10.15.3.28 StopWebServer() . . . 128
10.15.3.29 Terminate() . . . 128
10.16 sound4::CInstance Class Reference . . . 128
10.16.1 Detailed Description . . . 129
10.16.2 Constructor & Destructor Documentation . . . 130
10.16.2.1 CInstance() [1/2] . . . 130
10.16.2.2 CInstance() [2/2] . . . 130
10.16.2.3 ~CInstance() . . . 130
10.16.3 Member Function Documentation . . . 130
10.16.3.1 AddAudio() . . . 130
10.16.3.2 Create() . . . 130
10.16.3.3 GetAudio() . . . 131
10.16.3.4 GetBufferIn() . . . 131
10.16.3.5 GetBufferOut() . . . 131
10.16.3.6 GetChunkFrames() . . . 132
10.16.3.7 GetEstimatedDelay() . . . 132
10.16.3.8 GetMetadataInfos() . . . 132
10.16.3.9 GetParam() . . . 132
10.16.3.10 GetWebServerAppHealth() . . . 133
10.16.3.11 IsOk() . . . 133
10.16.3.12 NewClient() . . . 133
10.16.3.13 OnUpdateThreadStart() . . . 133
10.16.3.14 OnUpdateThreadStop() . . . 133
10.16.3.15 operator=() . . . 133
10.16.3.16 PresetManager_InformChange() . . . 133
10.16.3.17 ProcessAnyAudio() . . . 134
10.16.3.18 ProcessAudio() . . . 134
10.16.3.19 ProcessAudio_Planar() . . . 135
10.16.3.20 SetBus() . . . 135
10.16.3.21 SetMetadata() . . . 135
10.16.3.22 SetMetadataMulti() . . . 136
10.16.3.23 SetParam() . . . 136
10.16.3.24 SetPresetManager() . . . 136
10.16.3.25 SetWebServerAppHealth() . . . 137

10.16.3.26 StartWebServer() . . . 137
10.16.3.27 Stop() . . . 137
10.16.3.28 StopWebServer() . . . 138
10.16.3.29 Terminate() . . . 138
10.17 sound4::bigvoice::CPresetLoader Class Reference . . . 138
10.17.1 Detailed Description . . . 138
10.17.2 Constructor & Destructor Documentation . . . 139
10.17.2.1 CPresetLoader() . . . 139
10.17.2.2 ~CPresetLoader() . . . 139
10.17.3 Member Function Documentation . . . 139
10.17.3.1 Exists() . . . 139
10.17.3.2 GetAll() . . . 139
10.17.3.3 IsReadOnly() . . . 139
10.17.3.4 Read() . . . 139
10.17.3.5 Remove() . . . 139
10.17.3.6 Rename() . . . 139
10.17.3.7 Write() . . . 140
10.18 sound4::bigvoice::dyn::CPresetLoader Class Reference . . . 140
10.18.1 Detailed Description . . . 140
10.18.2 Constructor & Destructor Documentation . . . 140
10.18.2.1 CPresetLoader() . . . 140
10.18.2.2 ~CPresetLoader() . . . 141
10.18.3 Member Function Documentation . . . 141
10.18.3.1 Exists() . . . 141
10.18.3.2 GetAll() . . . 141
10.18.3.3 IsReadOnly() . . . 141
10.18.3.4 Read() . . . 141
10.18.3.5 Remove() . . . 141
10.18.3.6 Rename() . . . 141
10.18.3.7 Write() . . . 142
10.19 sound4::CPresetLoader Class Reference . . . 142
10.19.1 Detailed Description . . . 142
10.19.2 Constructor & Destructor Documentation . . . 142
10.19.2.1 CPresetLoader() . . . 142
10.19.2.2 ~CPresetLoader() . . . 143
10.19.3 Member Function Documentation . . . 143
10.19.3.1 Exists() . . . 143
10.19.3.2 GetAll() . . . 143
10.19.3.3 IsReadOnly() . . . 143
10.19.3.4 Read() . . . 143
10.19.3.5 Remove() . . . 143
10.19.3.6 Rename() . . . 143

10.19.3.7 Write() . . . 144
10.20 sound4::CProcessor Class Reference . . . 144
10.20.1 Detailed Description . . . 146
10.20.2 Constructor & Destructor Documentation . . . 146
10.20.2.1 CProcessor() . . . 146
10.20.3 Member Function Documentation . . . 146
10.20.3.1 AudioConvertFrom() . . . 146
10.20.3.2 AudioConvertTo() . . . 147
10.20.3.3 AudioMonoFromLiveStereo() . . . 147
10.20.3.4 AudioMonoToLiveStereo() . . . 147
10.20.3.5 GetAudioInputCount() . . . 147
10.20.3.6 GetAudioOutputCount() . . . 147
10.20.3.7 GetBytesFromFormat() . . . 148
10.20.3.8 GetChannelCount() . . . 148
10.20.3.9 GetChunkSizeInFrames() . . . 148
10.20.3.10 GetFormatFromName() . . . 148
10.20.3.11 GetFormatName() . . . 149
10.20.3.12 GetPossibleChunkSizeInFrames() . . . 149
10.20.3.13 GetPrefixSymbol() . . . 150
10.20.3.14 GetSampleRate() . . . 150
10.20.3.15 GetVersion() . . . 150
10.20.3.16 IsOk() . . . 150
10.20.3.17 Load() . . . 150
10.20.3.18 MonoToStereo() . . . 151
10.20.3.19 MonoToStereo_Planar() . . . 151
10.20.3.20 SetLoggerCallback() . . . 151
10.20.3.21 SetLogSeverity() . . . 151
10.20.3.22 StereoToMono() . . . 152
10.20.3.23 StereoToMono_Planar() . . . 152
10.20.4 Friends And Related Symbol Documentation . . . 152
10.20.4.1 CBus . . . 152
10.20.4.2 CInstance . . . 152
10.20.5 Member Data Documentation . . . 152
10.20.5.1 fnAddAudio . . . 152
10.20.5.2 fnAddPadAudio . . . 152
10.20.5.3 fnAudioConvertFrom . . . 153
10.20.5.4 fnAudioConvertTo . . . 153
10.20.5.5 fnAudioMonoFromLiveStereo . . . 153
10.20.5.6 fnAudioMonoToLiveStereo . . . 153
10.20.5.7 fnDeleteClient . . . 153
10.20.5.8 fnExitProcess . . . 153
10.20.5.9 fnFreeBus . . . 153

10.20.5.10 fnFreeJsonAnswer . . . 154
10.20.5.11 fnFreeMetadataInfos . . . 154
10.20.5.12 fnFreeParameters . . . 154
10.20.5.13 fnFreeParameterValue . . . 154
10.20.5.14 fnGetAudio . . . 154
10.20.5.15 fnGetAudioInputCount . . . 154
10.20.5.16 fnGetAudioOutputCount . . . 154
10.20.5.17 fnGetBufferIn . . . 155
10.20.5.18 fnGetBufferOut . . . 155
10.20.5.19 fnGetBytesFromFormat . . . 155
10.20.5.20 fnGetChannelCount . . . 155
10.20.5.21 fnGetChunkSizeInFrames . . . 155
10.20.5.22 fnGetEstimatedDelay . . . 155
10.20.5.23 fnGetFormatFromName . . . 155
10.20.5.24 fnGetFormatName . . . 156
10.20.5.25 fnGetMaxPacketFrame . . . 156
10.20.5.26 fnGetMetadataInfos . . . 156
10.20.5.27 fnGetOutputCount . . . 156
10.20.5.28 fnGetParameter . . . 156
10.20.5.29 fnGetPossibleChunkSizeInFrames . . . 156
10.20.5.30 fnGetProcessChunkFrames . . . 156
10.20.5.31 fnGetSampleRate . . . 157
10.20.5.32 fnGetVersion . . . 157
10.20.5.33 fnInitProcess . . . 157
10.20.5.34 fnInitProcess2 . . . 157
10.20.5.35 fnInitProcess3 . . . 157
10.20.5.36 fnMonoToStereo . . . 157
10.20.5.37 fnMonoToStereo_Planar . . . 157
10.20.5.38 fnNewBus . . . 158
10.20.5.39 fnNewClient . . . 158
10.20.5.40 fnNewParameters . . . 158
10.20.5.41 fnPresetManager_InformChange . . . 158
10.20.5.42 fnProcessAudio . . . 158
10.20.5.43 fnProcessAudio_Planar . . . 158
10.20.5.44 fnProcessJson . . . 158
10.20.5.45 fnSaveState . . . 159
10.20.5.46 fnSetInstanceBus . . . 159
10.20.5.47 fnSetLoggerCallback . . . 159
10.20.5.48 fnSetLogSeverity . . . 159
10.20.5.49 fnSetMetadata . . . 159
10.20.5.50 fnSetMetadataMulti . . . 159
10.20.5.51 fnSetParameter . . . 159

10.20.5.52 fnSetPresetManager . . . 160
10.20.5.53 fnStartUpdateThread . . . 160
10.20.5.54 fnStereoToMono . . . 160
10.20.5.55 fnStereoToMono_Planar . . . 160
10.20.5.56 fnStopUpdateThread . . . 160
10.20.5.57 fnTerminateProcess . . . 160
10.20.5.58 fnWaitUpdateThreadReady . . . 160
10.20.5.59 fnWebserver . . . 161
10.20.5.60 fnWebserver_FreeString . . . 161
10.20.5.61 fnWebserver_GetAppHealth . . . 161
10.20.5.62 fnWebserver_SetAppHealth . . . 161
10.20.5.63 fnWebserver_Status . . . 161
10.20.5.64 fnWebserver_Stop . . . 161
10.20.5.65 fnWebserver_tcp . . . 161
10.20.5.66 fnWebserver_tcp2 . . . 162
10.21 sound4::bigvoice::dyn::helper::DynFuncHolder< T > Class Template Reference . . . 162
10.21.1 Detailed Description . . . 162
10.21.2 Constructor & Destructor Documentation . . . 162
10.21.2.1 DynFuncHolder() [1/2] . . . 162
10.21.2.2 DynFuncHolder() [2/2] . . . 163
10.21.3 Member Function Documentation . . . 163
10.21.3.1 IsOk() . . . 163
10.21.3.2 operator bool() . . . 163
10.21.3.3 operator T*() . . . 163
10.22 sound4::helper::DynFuncHolder< T > Class Template Reference . . . 163
10.22.1 Detailed Description . . . 163
10.22.2 Constructor & Destructor Documentation . . . 164
10.22.2.1 DynFuncHolder() [1/2] . . . 164
10.22.2.2 DynFuncHolder() [2/2] . . . 164
10.22.3 Member Function Documentation . . . 164
10.22.3.1 IsOk() . . . 164
10.22.3.2 operator bool() . . . 164
10.22.3.3 operator T*() . . . 164
10.23 MyPresetManager Class Reference . . . 165
10.23.1 Detailed Description . . . 165
10.23.2 Constructor & Destructor Documentation . . . 166
10.23.2.1 MyPresetManager() [1/2] . . . 166
10.23.2.2 ~MyPresetManager() [1/2] . . . 166
10.23.2.3 MyPresetManager() [2/2] . . . 166
10.23.2.4 ~MyPresetManager() [2/2] . . . 166
10.23.3 Member Function Documentation . . . 166
10.23.3.1 Exists() [1/2] . . . 166

10.23.3.2 Exists() [2/2] . . . 166
10.23.3.3 GetAll() [1/2] . . . 166
10.23.3.4 GetAll() [2/2] . . . 167
10.23.3.5 IsReadOnly() [1/2] . . . 167
10.23.3.6 IsReadOnly() [2/2] . . . 167
10.23.3.7 Read() [1/2] . . . 167
10.23.3.8 Read() [2/2] . . . 167
10.23.3.9 Remove() [1/2] . . . 167
10.23.3.10 Remove() [2/2] . . . 168
10.23.3.11 Rename() [1/2] . . . 168
10.23.3.12 Rename() [2/2] . . . 168
10.23.3.13 SetupPath() [1/2] . . . 168
10.23.3.14 SetupPath() [2/2] . . . 168
10.23.3.15 Write() [1/2] . . . 168
10.23.3.16 Write() [2/2] . . . 169
10.24 sound4::bigvoice::dyn::helper::SampleFormat< T > Struct Template Reference . . . 169
10.24.1 Detailed Description . . . 169
10.25 sound4::bigvoice::helper::SampleFormat< T > Struct Template Reference . . . 169
10.25.1 Detailed Description . . . 170
10.26 sound4::helper::SampleFormat< T > Struct Template Reference . . . 171
10.26.1 Detailed Description . . . 171
10.27 sound4::bigvoice::dyn::helper::SampleFormat< float > Struct Reference . . . 171
10.27.1 Detailed Description . . . 171
10.27.2 Member Data Documentation . . . 172
10.27.2.1 format . . . 172
10.28 sound4::bigvoice::helper::SampleFormat< float > Struct Reference . . . 172
10.28.1 Detailed Description . . . 172
10.28.2 Member Data Documentation . . . 172
10.28.2.1 format . . . 172
10.29 sound4::helper::SampleFormat< float > Struct Reference . . . 172
10.29.1 Detailed Description . . . 173
10.29.2 Member Data Documentation . . . 173
10.29.2.1 format . . . 173
10.30 sound4::bigvoice::dyn::helper::SampleFormat< int16_t > Struct Reference . . . 173
10.30.1 Detailed Description . . . 173
10.30.2 Member Data Documentation . . . 173
10.30.2.1 format . . . 173
10.31 sound4::bigvoice::helper::SampleFormat< int16_t > Struct Reference . . . 174
10.31.1 Detailed Description . . . 174
10.31.2 Member Data Documentation . . . 174
10.31.2.1 format . . . 174
10.32 sound4::helper::SampleFormat< int16_t > Struct Reference . . . 174

10.32.1 Detailed Description . . . 174
10.32.2 Member Data Documentation . . . 174
10.32.2.1 format . . . 174
10.33 sound4::bigvoice::dyn::helper::SampleFormat< int32_t > Struct Reference . . . 175
10.33.1 Detailed Description . . . 175
10.33.2 Member Data Documentation . . . 175
10.33.2.1 format . . . 175
10.34 sound4::bigvoice::helper::SampleFormat< int32_t > Struct Reference . . . 175
10.34.1 Detailed Description . . . 175
10.34.2 Member Data Documentation . . . 176
10.34.2.1 format . . . 176
10.35 sound4::helper::SampleFormat< int32_t > Struct Reference . . . 176
10.35.1 Detailed Description . . . 176
10.35.2 Member Data Documentation . . . 176
10.35.2.1 format . . . 176
10.36 SOUND4_ProcessInfo Struct Reference . . . 176
10.36.1 Detailed Description . . . 177
10.36.2 Member Data Documentation . . . 177
10.36.2.1 name . . . 177
10.36.2.2 prefix . . . 177
10.36.2.3 shortname . . . 177
10.36.2.4 struct_size . . . 177
10.36.2.5 version . . . 178
10.37 sound4cl_SOUND4_ProcessInfo Struct Reference . . . 178
10.37.1 Detailed Description . . . 178
10.37.2 Member Data Documentation . . . 178
10.37.2.1 name . . . 178
10.37.2.2 prefix . . . 178
10.37.2.3 shortname . . . 179
10.37.2.4 struct_size . . . 179
10.37.2.5 version . . . 179

11 File Documentation **181**

11.1 documentation.h File Reference . . . 181
11.2 documentation.h . . . 181
11.3 sound4.bigvoice.cl-pipe-cxx.cpp File Reference . . . 181
11.3.1 Function Documentation . . . 182
11.3.1.1 GetEnv() . . . 182
11.3.1.2 main() . . . 182
11.3.1.3 MyLogger() . . . 182
11.3.1.4 ProcessLoop() . . . 182
11.3.1.5 Usage() . . . 183

11.4 sound4.bigvoice.cl-pipe-cxx.cpp . . . 183
11.5 sound4.bigvoice.cl-pipe-dyn.cpp File Reference . . . 185
11.5.1 Function Documentation . . . 185
11.5.1.1 GetEnv() . . . 185
11.5.1.2 main() . . . 186
11.5.1.3 MyLogger() . . . 186
11.5.1.4 ProcessLoop() . . . 186
11.5.1.5 Usage() . . . 186
11.6 sound4.bigvoice.cl-pipe-dyn.cpp . . . 187
11.7 sound4.bigvoice.cl-pipe.cpp File Reference . . . 190
11.7.1 Macro Definition Documentation . . . 190
11.7.1.1 CHANNEL_COUNT . . . 190
11.7.1.2 CHUNK_SIZE . . . 191
11.7.1.3 SAMPLE_RATE . . . 191
11.7.2 Function Documentation . . . 191
11.7.2.1 main() . . . 191
11.7.2.2 MyLogger() . . . 191
11.7.2.3 ProcessLoop() . . . 191
11.7.2.4 Usage() . . . 192
11.8 sound4.bigvoice.cl-pipe.cpp . . . 192
11.9 sound4cl-pipe.cpp File Reference . . . 195
11.9.1 Function Documentation . . . 195
11.9.1.1 GetEnv() . . . 195
11.9.1.2 main() . . . 196
11.9.1.3 ProcessLoop() . . . 196
11.10 sound4cl-pipe.cpp . . . 197
11.11 sound4.bigvoice.cl.h File Reference . . . 200
11.11.1 Macro Definition Documentation . . . 202
11.11.1.1 BIGVOICE_DECL . . . 202
11.11.1.2 BIGVOICE_HAS_CLOUDBUS . . . 203
11.11.1.3 BIGVOICE_HAS_WEBSERVER . . . 203
11.12 sound4.bigvoice.cl.h . . . 203
11.13 sound4.bigvoice.cl.hpp File Reference . . . 207
11.13.1 Macro Definition Documentation . . . 208
11.13.1.1 fs_strdup . . . 208
11.14 sound4.bigvoice.cl.hpp . . . 209
11.15 sound4.bigvoice.cl_dyn.hpp File Reference . . . 216
11.15.1 Detailed Description . . . 217
11.15.2 Macro Definition Documentation . . . 218
11.15.2.1 fs_strdup . . . 218
11.15.2.2 UNIXLIKE . . . 218
11.16 sound4.bigvoice.cl_dyn.hpp . . . 218

11.17 sound4cl.hpp File Reference . . . 231
11.17.1 Detailed Description . . . 233
11.17.2 Macro Definition Documentation . . . 233
11.17.2.1 UNIXLIKE . . . 233
11.18 sound4cl.hpp . . . 233
11.19 sound4cl_cdef.h File Reference . . . 246
11.19.1 Macro Definition Documentation . . . 248
11.19.1.1 fs_strdup . . . 248
11.19.1.2 SOUND4_INVALID_WEBSERVER_ID . . . 249
11.19.1.3 SOUND4_WEBSERVER_HTTP_OK . . . 249
11.19.1.4 SOUND4_WEBSERVER_HTTPS_OK . . . 249
11.19.2 Typedef Documentation . . . 249
11.19.2.1 fs_char . . . 249
11.19.2.2 sound4cl_loggerfn . . . 249
11.19.2.3 sound4cl_storage_exists . . . 249
11.19.2.4 sound4cl_storage_getall . . . 249
11.19.2.5 sound4cl_storage_getall_free . . . 250
11.19.2.6 sound4cl_storage_reader . . . 250
11.19.2.7 sound4cl_storage_reader_free . . . 250
11.19.2.8 sound4cl_storage_remove . . . 250
11.19.2.9 sound4cl_storage_rename . . . 250
11.19.2.10 sound4cl_storage_writer . . . 250
11.19.3 Enumeration Type Documentation . . . 250
11.19.3.1 sound4cl_LogSeverity . . . 250
11.19.3.2 sound4cl_PresetChange_Kind . . . 251
11.19.3.3 sound4cl_SampleFormat . . . 251
11.19.4 Function Documentation . . . 252
11.19.4.1 sound4cl_AddAudio() . . . 252
11.19.4.2 sound4cl_AddPadAudio() . . . 252
11.19.4.3 sound4cl_AudioConvertFrom() . . . 252
11.19.4.4 sound4cl_AudioConvertTo() . . . 252
11.19.4.5 sound4cl_AudioMonoFromLiveStereo() . . . 252
11.19.4.6 sound4cl_AudioMonoToLiveStereo() . . . 252
11.19.4.7 sound4cl_DeleteClient() . . . 253
11.19.4.8 sound4cl_ExitProcess() . . . 253
11.19.4.9 sound4cl_FreeBus() . . . 253
11.19.4.10 sound4cl_FreeJsonAnswer() . . . 253
11.19.4.11 sound4cl_FreeMetadataInfos() . . . 253
11.19.4.12 sound4cl_FreeParameters() . . . 253
11.19.4.13 sound4cl_FreeParameterValue() . . . 253
11.19.4.14 sound4cl_GetAudio() . . . 253
11.19.4.15 sound4cl_GetAudioInputCount() . . . 254

11.19.4.16 sound4cl_GetAudioOutputCount() . . . 254
11.19.4.17 sound4cl_GetBufferIn() . . . 254
11.19.4.18 sound4cl_GetBufferOut() . . . 254
11.19.4.19 sound4cl_GetBytesFromFormat() . . . 254
11.19.4.20 sound4cl_GetChannelCount() . . . 254
11.19.4.21 sound4cl_GetChunkSizeInFrames() . . . 254
11.19.4.22 sound4cl_GetEstimatedDelay() . . . 254
11.19.4.23 sound4cl_GetFormatFromName() . . . 254
11.19.4.24 sound4cl_GetFormatName() . . . 255
11.19.4.25 sound4cl_GetMaxPacketFrame() . . . 255
11.19.4.26 sound4cl_GetMetadataInfos() . . . 255
11.19.4.27 sound4cl_GetOutputCount() . . . 255
11.19.4.28 sound4cl_GetParameter() . . . 255
11.19.4.29 sound4cl_GetPossibleChunkSizeInFrames() . . . 255
11.19.4.30 sound4cl_GetProcessChunkFrames() . . . 255
11.19.4.31 sound4cl_GetSampleRate() . . . 255
11.19.4.32 sound4cl_GetVersion() . . . 255
11.19.4.33 sound4cl_InitProcess() . . . 256
11.19.4.34 sound4cl_InitProcess2() . . . 256
11.19.4.35 sound4cl_InitProcess3() . . . 256
11.19.4.36 sound4cl_MonoToStereo() . . . 256
11.19.4.37 sound4cl_MonoToStereo_Planar() . . . 256
11.19.4.38 sound4cl_NewBus() . . . 256
11.19.4.39 sound4cl_NewClient() . . . 257
11.19.4.40 sound4cl_NewParameters() . . . 257
11.19.4.41 sound4cl_PresetManager_InformChange() . . . 257
11.19.4.42 sound4cl_ProcessAudio() . . . 257
11.19.4.43 sound4cl_ProcessAudio_Planar() . . . 257
11.19.4.44 sound4cl_ProcessJson() . . . 257
11.19.4.45 sound4cl_SaveState() . . . 257
11.19.4.46 sound4cl_SetInstanceBus() . . . 257
11.19.4.47 sound4cl_SetLoggerCallback() . . . 258
11.19.4.48 sound4cl_SetLogSeverity() . . . 258
11.19.4.49 sound4cl_SetMetadata() . . . 258
11.19.4.50 sound4cl_SetMetadataMulti() . . . 258
11.19.4.51 sound4cl_SetParameter() . . . 258
11.19.4.52 sound4cl_SetPresetManager() . . . 258
11.19.4.53 sound4cl_SOUND4_GetProcessInfo() . . . 258
11.19.4.54 sound4cl_StartUpdateThread() . . . 259
11.19.4.55 sound4cl_StereoToMono() . . . 259
11.19.4.56 sound4cl_StereoToMono_Planar() . . . 259
11.19.4.57 sound4cl_StopUpdateThread() . . . 259

11.19.4.58 sound4cl_TerminateProcess() . . . 259
11.19.4.59 sound4cl_WaitUpdateThreadReady() . . . 259
11.19.4.60 sound4cl_Webserver() . . . 259
11.19.4.61 sound4cl_Webserver_FreeString() . . . 260
11.19.4.62 sound4cl_Webserver_GetAppHealth() . . . 260
11.19.4.63 sound4cl_Webserver_SetAppHealth() . . . 260
11.19.4.64 sound4cl_Webserver_Status() . . . 260
11.19.4.65 sound4cl_Webserver_Stop() . . . 260
11.19.4.66 sound4cl_Webserver_tcp() . . . 260
11.19.4.67 sound4cl_Webserver_tcp2() . . . 260
11.20 sound4cl_cdef.h . . . 261

Index **265**

Chapter 1

SOUND4 BIGVOICE.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 bigvoice_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.bigvoice.cl-pipe.cpp

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

Note

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

1.2 Examples

See Sample using C interface

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

See Sample using C++ interface wrapper

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

Chapter 2

Deprecated List

Member bigvoice_GetChunkSizeInFrames ()

Use bigvoice_GetPossibleChunkSizeInFrames

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

Use bigvoice_InitProcess3

Member bigvoice_InitProcess2 (const char ∗LoginKey, const char ∗RadioName, const char ∗Access_Key_ID, const char ∗Access_Key_Secret, const char ∗save_path, const struct bigvoice_CParameters ∗parameters)

Use bigvoice_InitProcess3

Chapter 3

Topic Index

3.1 Topics

Here is a list of all topics with brief descriptions:

Sample using C interface . . . 17
Environment variables and parameters . . . 19
Global setup . . . 20
Process creation/destruction . . . 27
Instance's bus . . . 32
Custom preset managagement . . . 33
Metadata Management . . . 38
Update Management . . . 40
Direct processing . . . 42
Audio managing (any audio) . . . 44
Audio convert helpers . . . 48
Client's JSon config interfacing . . . 52
JSON syntax . . . 54
JSON syntax for internal setup . . . 57
Web server . . . 61
Sample using C++ interface wrapper . . . 66
Sample using C++ interface wrapper with dynamic library loading . . . 67
Sample using Generic C++ interface (multi-process compatible) . . . 69

Chapter 4

Namespace Index

4.1 Namespace List

Here is a list of all namespaces with brief descriptions:

dyn 73
sound4 73
sound4::bigvoice 76
sound4::bigvoice::dyn 78
sound4::bigvoice::dyn::helper 80
sound4::bigvoice::helper 81
sound4::helper 81
std 82

Chapter 5

Hierarchical Index

5.1 Class Hierarchy

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

bigvoice_CBus . . . 83
bigvoice_CClientInstance . . . 83
bigvoice_CInstance . . . 83
bigvoice_CParameters . . . 84
sound4::bigvoice::CBus . . . 84
sound4::bigvoice::dyn::CBus . . . 85
sound4::CBus . . . 86
sound4::bigvoice::CInstance::CClient . . . 87
sound4::bigvoice::dyn::CInstance::CClient . . . 89
sound4::CInstance::CClient . . . 90
sound4::bigvoice::dyn::CDynLib . . . 92
sound4::bigvoice::dyn::helper::CDynLoader . . . 104
sound4::helper::CDynLoader . . . 106
sound4::bigvoice::CInstance . . . 109
sound4::bigvoice::dyn::CInstance . . . 118
sound4::CInstance . . . 128
sound4::bigvoice::CPresetLoader . . . 138
sound4::bigvoice::dyn::CPresetLoader . . . 140
MyPresetManager . . . 165
sound4::CPresetLoader . . . 142
MyPresetManager . . . 165
sound4::CProcessor . . . 144
sound4::bigvoice::dyn::helper::DynFuncHolder< T > . . . 162
sound4::helper::DynFuncHolder< T > . . . 163
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_AddAudio) > . . . 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_AddPadAudio) > . . . 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_AudioConvertFrom) > . . . 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_AudioConvertTo) > . . . 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_AudioMonoFromLiveStereo) > . . . 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_AudioMonoToLiveStereo) > . . . 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_DeleteClient) > . . . 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_ExitProcess) > . . . 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_FreeBus) > . . . 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_FreeJsonAnswer) > . . . 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_FreeMetadataInfos) > . . . 162

sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_FreeParameters) > 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_FreeParameterValue) > 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_GetAudio) > 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_GetAudioInputCount) > 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_GetAudioOutputCount) > 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_GetBufferIn) > 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_GetBufferOut) > 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_GetBytesFromFormat) > 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_GetChannelCount) > 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_GetChunkSizeInFrames) > 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_GetEstimatedDelay) > 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_GetFormatFromName) > 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_GetFormatName) > 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_GetMaxPacketFrame) > 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_GetMetadataInfos) > 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_GetOutputCount) > 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_GetParameter) > 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_GetPossibleChunkSizeInFrames) > . 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_GetProcessChunkFrames) > 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_GetSampleRate) > 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_GetVersion) > 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_InitProcess) > 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_InitProcess2) > 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_InitProcess3) > 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_MonoToStereo) > 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_MonoToStereo_Planar) > 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_NewBus) > 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_NewClient) > 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_NewParameters) > 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_PresetManager_InformChange) > . 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_ProcessAudio) > 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_ProcessAudio_Planar) > 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_ProcessJson) > 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_SaveState) > 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_SetInstanceBus) > 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_SetLoggerCallback) > 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_SetLogSeverity) > 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_SetMetadata) > 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_SetMetadataMulti) > 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_SetParameter) > 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_SetPresetManager) > 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_StartUpdateThread) > 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_StereoToMono) > 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_StereoToMono_Planar) > 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_StopUpdateThread) > 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_TerminateProcess) > 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_WaitUpdateThreadReady) > 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_Webserver) > 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_Webserver_FreeString) > 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_Webserver_GetAppHealth) > 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_Webserver_SetAppHealth) > 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_Webserver_Status) > 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_Webserver_Stop) > 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_Webserver_tcp) > 162
sound4::bigvoice::dyn::helper::DynFuncHolder< decltype(bigvoice_Webserver_tcp2) > 162
sound4::helper::DynFuncHolder< decltype(sound4cl_AddAudio) > 163
sound4::helper::DynFuncHolder< decltype(sound4cl_AddPadAudio) > 163
sound4::helper::DynFuncHolder< decltype(sound4cl_AudioConvertFrom) > 163

sound4::helper::DynFuncHolder< decltype(sound4cl_AudioConvertTo) > . . . 163
sound4::helper::DynFuncHolder< decltype(sound4cl_AudioMonoFromLiveStereo) > . . . 163
sound4::helper::DynFuncHolder< decltype(sound4cl_AudioMonoToLiveStereo) > . . . 163
sound4::helper::DynFuncHolder< decltype(sound4cl_DeleteClient) > . . . 163
sound4::helper::DynFuncHolder< decltype(sound4cl_ExitProcess) > . . . 163
sound4::helper::DynFuncHolder< decltype(sound4cl_FreeBus) > . . . 163
sound4::helper::DynFuncHolder< decltype(sound4cl_FreeJsonAnswer) > . . . 163
sound4::helper::DynFuncHolder< decltype(sound4cl_FreeMetadataInfos) > . . . 163
sound4::helper::DynFuncHolder< decltype(sound4cl_FreeParameters) > . . . 163
sound4::helper::DynFuncHolder< decltype(sound4cl_FreeParameterValue) > . . . 163
sound4::helper::DynFuncHolder< decltype(sound4cl_GetAudio) > . . . 163
sound4::helper::DynFuncHolder< decltype(sound4cl_GetAudioInputCount) > . . . 163
sound4::helper::DynFuncHolder< decltype(sound4cl_GetAudioOutputCount) > . . . 163
sound4::helper::DynFuncHolder< decltype(sound4cl_GetBufferIn) > . . . 163
sound4::helper::DynFuncHolder< decltype(sound4cl_GetBufferOut) > . . . 163
sound4::helper::DynFuncHolder< decltype(sound4cl_GetBytesFromFormat) > . . . 163
sound4::helper::DynFuncHolder< decltype(sound4cl_GetChannelCount) > . . . 163
sound4::helper::DynFuncHolder< decltype(sound4cl_GetChunkSizeInFrames) > . . . 163
sound4::helper::DynFuncHolder< decltype(sound4cl_GetEstimatedDelay) > . . . 163
sound4::helper::DynFuncHolder< decltype(sound4cl_GetFormatFromName) > . . . 163
sound4::helper::DynFuncHolder< decltype(sound4cl_GetFormatName) > . . . 163
sound4::helper::DynFuncHolder< decltype(sound4cl_GetMaxPacketFrame) > . . . 163
sound4::helper::DynFuncHolder< decltype(sound4cl_GetMetadataInfos) > . . . 163
sound4::helper::DynFuncHolder< decltype(sound4cl_GetOutputCount) > . . . 163
sound4::helper::DynFuncHolder< decltype(sound4cl_GetParameter) > . . . 163
sound4::helper::DynFuncHolder< decltype(sound4cl_GetPossibleChunkSizeInFrames) > . . . 163
sound4::helper::DynFuncHolder< decltype(sound4cl_GetProcessChunkFrames) > . . . 163
sound4::helper::DynFuncHolder< decltype(sound4cl_GetSampleRate) > . . . 163
sound4::helper::DynFuncHolder< decltype(sound4cl_GetVersion) > . . . 163
sound4::helper::DynFuncHolder< decltype(sound4cl_InitProcess) > . . . 163
sound4::helper::DynFuncHolder< decltype(sound4cl_InitProcess2) > . . . 163
sound4::helper::DynFuncHolder< decltype(sound4cl_InitProcess3) > . . . 163
sound4::helper::DynFuncHolder< decltype(sound4cl_MonoToStereo) > . . . 163
sound4::helper::DynFuncHolder< decltype(sound4cl_MonoToStereo_Planar) > . . . 163
sound4::helper::DynFuncHolder< decltype(sound4cl_NewBus) > . . . 163
sound4::helper::DynFuncHolder< decltype(sound4cl_NewClient) > . . . 163
sound4::helper::DynFuncHolder< decltype(sound4cl_NewParameters) > . . . 163
sound4::helper::DynFuncHolder< decltype(sound4cl_PresetManager_InformChange) > . . . 163
sound4::helper::DynFuncHolder< decltype(sound4cl_ProcessAudio) > . . . 163
sound4::helper::DynFuncHolder< decltype(sound4cl_ProcessAudio_Planar) > . . . 163
sound4::helper::DynFuncHolder< decltype(sound4cl_ProcessJson) > . . . 163
sound4::helper::DynFuncHolder< decltype(sound4cl_SaveState) > . . . 163
sound4::helper::DynFuncHolder< decltype(sound4cl_SetInstanceBus) > . . . 163
sound4::helper::DynFuncHolder< decltype(sound4cl_SetLoggerCallback) > . . . 163
sound4::helper::DynFuncHolder< decltype(sound4cl_SetLogSeverity) > . . . 163
sound4::helper::DynFuncHolder< decltype(sound4cl_SetMetadata) > . . . 163
sound4::helper::DynFuncHolder< decltype(sound4cl_SetMetadataMulti) > . . . 163
sound4::helper::DynFuncHolder< decltype(sound4cl_SetParameter) > . . . 163
sound4::helper::DynFuncHolder< decltype(sound4cl_SetPresetManager) > . . . 163
sound4::helper::DynFuncHolder< decltype(sound4cl_StartUpdateThread) > . . . 163
sound4::helper::DynFuncHolder< decltype(sound4cl_StereoToMono) > . . . 163
sound4::helper::DynFuncHolder< decltype(sound4cl_StereoToMono_Planar) > . . . 163
sound4::helper::DynFuncHolder< decltype(sound4cl_StopUpdateThread) > . . . 163
sound4::helper::DynFuncHolder< decltype(sound4cl_TerminateProcess) > . . . 163
sound4::helper::DynFuncHolder< decltype(sound4cl_WaitUpdateThreadReady) > . . . 163
sound4::helper::DynFuncHolder< decltype(sound4cl_Webserver) > . . . 163
sound4::helper::DynFuncHolder< decltype(sound4cl_Webserver_FreeString) > . . . 163
sound4::helper::DynFuncHolder< decltype(sound4cl_Webserver_GetAppHealth) > . . . 163

sound4::helper::DynFuncHolder< decltype(sound4cl_Webserver_SetAppHealth) > 163
sound4::helper::DynFuncHolder< decltype(sound4cl_Webserver_Status) > 163
sound4::helper::DynFuncHolder< decltype(sound4cl_Webserver_Stop) > 163
sound4::helper::DynFuncHolder< decltype(sound4cl_Webserver_tcp) > 163
sound4::helper::DynFuncHolder< decltype(sound4cl_Webserver_tcp2) > 163
sound4::bigvoice::dyn::helper::SampleFormat< T > 169
sound4::bigvoice::helper::SampleFormat< T > 169
sound4::helper::SampleFormat< T > 171
sound4::bigvoice::dyn::helper::SampleFormat< float > 171
sound4::bigvoice::helper::SampleFormat< float > 172
sound4::helper::SampleFormat< float > 172
sound4::bigvoice::dyn::helper::SampleFormat< int16_t > 173
sound4::bigvoice::helper::SampleFormat< int16_t > 174
sound4::helper::SampleFormat< int16_t > 174
sound4::bigvoice::dyn::helper::SampleFormat< int32_t > 175
sound4::bigvoice::helper::SampleFormat< int32_t > 175
sound4::helper::SampleFormat< int32_t > 176
SOUND4_ProcessInfo 176
sound4cl_SOUND4_ProcessInfo 178

Chapter 6

Class Index

6.1 Class List

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

bigvoice_CBus . . . 83
bigvoice_CClientInstance . . . 83
bigvoice_CInstance . . . 83
bigvoice_CParameters . . . 84
sound4::bigvoice::CBus . . . 84
sound4::bigvoice::dyn::CBus . . . 85
sound4::CBus
    Shared Bus . . . 86
sound4::bigvoice::CInstance::CClient . . . 87
sound4::bigvoice::dyn::CInstance::CClient . . . 89
sound4::CInstance::CClient . . . 90
sound4::bigvoice::dyn::CDynLib
    Dynamic library interface . . . 92
sound4::bigvoice::dyn::helper::CDynLoader
    Helper to load dynamic library . . . 104
sound4::helper::CDynLoader
    Helper to load dynamic library . . . 106
sound4::bigvoice::CInstance . . . 109
sound4::bigvoice::dyn::CInstance . . . 118
sound4::CInstance
    Instance handling class . . . 128
sound4::bigvoice::CPresetLoader . . . 138
sound4::bigvoice::dyn::CPresetLoader . . . 140
sound4::CPresetLoader
    Custom preset handler helper . . . 142
sound4::CProcessor
    Dynamic library interface . . . 144
sound4::bigvoice::dyn::helper::DynFuncHolder< T >
    Helper to wrap a function pointer . . . 162
sound4::helper::DynFuncHolder< T >
    Helper to wrap a function pointer . . . 163
MyPresetManager
    [Custom preset manager] . . . 165
sound4::bigvoice::dyn::helper::SampleFormat< T >
    Helper for Sample format types . . . 169

sound4::bigvoice::helper::SampleFormat< T >
Helper for Sample format types . . . 169
sound4::helper::SampleFormat< T >
Helper for Sample format types . . . 171
sound4::bigvoice::dyn::helper::SampleFormat< float > . . . 171
sound4::bigvoice::helper::SampleFormat< float > . . . 172
sound4::helper::SampleFormat< float > . . . 172
sound4::bigvoice::dyn::helper::SampleFormat< int16_t > . . . 173
sound4::bigvoice::helper::SampleFormat< int16_t > . . . 174
sound4::helper::SampleFormat< int16_t > . . . 174
sound4::bigvoice::dyn::helper::SampleFormat< int32_t > . . . 175
sound4::bigvoice::helper::SampleFormat< int32_t > . . . 175
sound4::helper::SampleFormat< int32_t > . . . 176
SOUND4_ProcessInfo . . . 176
sound4cl_SOUND4_ProcessInfo . . . 178

Chapter 7

File Index

7.1 File List

Here is a list of all files with brief descriptions:

documentation.h 181
sound4.bigvoice.cl-pipe-cxx.cpp
C++ sample using C++ interface wrapper 181
sound4.bigvoice.cl-pipe-dyn.cpp
C++ sample using C++ interface wrapper with dynamic library loading 185
sound4.bigvoice.cl-pipe.cpp
C++ sample using only C interface 190
sound4cl-pipe.cpp
C++ sample using C++ interface wrapper with dynamic library loading Works with any SOUND4 processor provided 195
sound4.bigvoice.cl.h
C interface for library 200
sound4.bigvoice.cl.hpp
C++ wrapper for library 207
sound4.bigvoice.cl_dyn.hpp
C++ wrapper for dynamic loading library 216
sound4cl.hpp
C++ wrapper for dynamic loading library 231
sound4cl_cdef.h
C function signature definition for all sound4.cl libraries 246

Chapter 8

Topic Documentation

8.1 Sample using C interface

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

8.1.1 Sample to create the processing instance

see Process creation/destruction and Update Management

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

    // Create the processing instance
    pInstance=bigvoice_InitProcess2(pLoginKey,pRadioName,pKeyId,pAccessKey,store_path.c_str(), params);
    bigvoice_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 « "bigvoice_StartUpdateThread (no json port)" « std::endl;
       bigvoice_StartUpdateThread(pInstance, 0);
       std::cerr « "bigvoice_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 (bigvoice_WaitUpdateThreadReady(pInstance, 5000)<0) {
        std::cerr « "Timeout to create the process instance"« std::endl;
    }
    std::cerr « "Waiting update thread DONE" « std::endl;

#if (BIGVOICE_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 = bigvoice_Webserver(listenport,listenport_secure,pInstance);
        if (webserver==SOUND4_INVALID_WEBSERVER_ID) {
            std::cerr « "Starting web server FAILED" « std::endl;
        } else {
            int web_status=bigvoice_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 // (BIGVOICE_HAS_WEBSERVER)

     // Force load a factory preset now
     // Could be used instead of setting a store_path
     /*{
         bigvoice_CClientInstance *pClient=bigvoice_NewClient(pInstance);
         const char *answer=bigvoice_ProcessJson(pClient, "{ \"factoryload\": \"All Programs\" }", NULL);
         std::cerr « "Loaded factory preset answer: " « answer« std::endl;
         bigvoice_FreeJsonAnswer(answer);
         bigvoice_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
        bigvoice_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 (BIGVOICE_HAS_WEBSERVER)
    // Stopping the webserver
    std::cerr « "Stopping web server" « std::endl;
    bigvoice_Webserver_Stop(webserver, 1000);
    std::cerr « "Stopping web server DONE" « std::endl;
#endif // (BIGVOICE_HAS_WEBSERVER)

    // Stopping the processing instance
    std::cerr « "Stopping Update thread" « std::endl;
    bigvoice_StopUpdateThread(pInstance); // this will exit the update thread
    std::cerr « "Stopping Update thread DONE" « std::endl;

    updater.join();

    bigvoice_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 bigvoice_InitProcess2

8.2.1.1 Webserver authentification setup :

- `SETUP_USER` see bigvoice_InitProcess
- `SETUP_SECRET` see bigvoice_InitProcess
- `ADMIN_USER` see bigvoice_InitProcess
- `ADMIN_SECRET` see bigvoice_InitProcess

8.2.1.2 Webserver secure :

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

8.2.1.3 Network access :

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

8.2.2 Parameters

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

Run mode:

- `FILE_MODE=seconds` : force file mode, settings the maximum estimated audio duration (non realtime, wait license ok to process)
- `STATE_FILENAME=filename` : force this instance to store its state in file named `filename.json` (instead of `state.json`)
- `STATE_ONAIR=filename` : force this instance to store its on air preset name in file named `filename.onair` (instead of none). Used for sharing between instances.

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

Web server options:

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

NOTE: boolvalue are (case insensitive):

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

8.2.3 Preset sharing

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

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

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

8.3 Global setup

Classes

- struct SOUND4_ProcessInfo

Macros

- #define BIGVOICE_AUDIOFRAME_COUNT 12
- #define BIGVOICE_CHANNEL_COUNT 1
- #define BIGVOICE_AUDIO_INPUT_COUNT 1
- #define BIGVOICE_AUDIO_OUTPUT_COUNT 1
- #define BIGVOICE_SAMPLE_RATE 48000

Typedefs

- typedef void(∗ bigvoice_loggerfn) (enum bigvoice_LogSeverity, const char ∗)

Enumerations

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

Functions

- const struct SOUND4_ProcessInfo ∗ SOUND4_GetProcessInfo ()
- const char ∗ bigvoice_GetVersion ()
- unsigned int bigvoice_GetChunkSizeInFrames ()
- unsigned int ∗ bigvoice_GetPossibleChunkSizeInFrames ()
- unsigned int bigvoice_GetChannelCount ()
- unsigned int bigvoice_GetAudioInputCount ()
- unsigned int bigvoice_GetAudioOutputCount ()
- unsigned int bigvoice_GetSampleRate ()
- void bigvoice_SetLoggerCallback (bigvoice_loggerfn logger)
- void bigvoice_SetLogSeverity (enum bigvoice_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 BIGVOICE_AUDIO_INPUT_COUNT

```
#define BIGVOICE_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 bigvoice_GetAudioInputCount()

Definition at line 184 of file sound4.bigvoice.cl.h.

8.3.2.2 BIGVOICE_AUDIO_OUTPUT_COUNT

```
#define BIGVOICE_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 bigvoice_GetAudioOutputCount()

Definition at line 190 of file sound4.bigvoice.cl.h.

8.3.2.3 BIGVOICE_AUDIOFRAME_COUNT

```
#define BIGVOICE_AUDIOFRAME_COUNT 12
```

This version number of frames in a process chunk.

Remarks

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

Definition at line 172 of file sound4.bigvoice.cl.h.

8.3.2.4 BIGVOICE_CHANNEL_COUNT

```
#define BIGVOICE_CHANNEL_COUNT 1
```

This version number of channels.

Remarks

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

Definition at line 178 of file sound4.bigvoice.cl.h.

8.3.2.5 BIGVOICE_SAMPLE_RATE

```
#define BIGVOICE_SAMPLE_RATE 48000
```

This input/output sample-rate.

Remarks

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

Definition at line 196 of file sound4.bigvoice.cl.h.

8.3.3 Typedef Documentation

8.3.3.1 bigvoice_loggerfn

```
typedef void(* bigvoice_loggerfn) (enum bigvoice_LogSeverity, const char *)
```

The logging callback format

Warning

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

Note

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

Definition at line 311 of file sound4.bigvoice.cl.h.

8.3.4 Enumeration Type Documentation

8.3.4.1 bigvoice_LogSeverity

```
enum bigvoice_LogSeverity
```

The logging levels

Enumerator

none	
fatal	
error	
warning	
info	
verbose	
verbose2	
verbose3	
verbose4	
verbose5	

Definition at line 292 of file sound4.bigvoice.cl.h.

8.3.5 Function Documentation

8.3.5.1 bigvoice_GetAudioInputCount()

```
unsigned int bigvoice_GetAudioInputCount ( )
```

Get the number of inputs for this process

Remarks

All inputs have bigvoice_GetChannelCount() channels

Fixed to 1, but should be checked for backward compatibility

Returns

The process input count

8.3.5.2 bigvoice_GetAudioOutputCount()

```
unsigned int bigvoice_GetAudioOutputCount ( )
```

Get the number of outputs for this process

Remarks

All outputs have bigvoice_GetChannelCount() channels

Fixed to 1, but should be checked for backward compatibility

Returns

The process outputs count

8.3.5.3 bigvoice_GetChannelCount()

```
unsigned int bigvoice_GetChannelCount ( )
```

Get the number of channels needed by this process

Remarks

Fixed to 1, but should be checked for backward compatibility

Returns

The process channels count

8.3.5.4 bigvoice_GetChunkSizeInFrames()

```
unsigned int bigvoice_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 bigvoice_GetChannelCount samples

Remarks

Fixed to 12, but should be checked for backward compatibility

Returns

The process chunk size in frame

Deprecated Use bigvoice_GetPossibleChunkSizeInFrames

8.3.5.5 bigvoice_GetPossibleChunkSizeInFrames()

```
unsigned int * bigvoice_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 bigvoice_GetChannelCount samples

Returns

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

8.3.5.6 bigvoice_GetSampleRate()

```
unsigned int bigvoice_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 bigvoice_GetVersion()

```
const char * bigvoice_GetVersion ( )
```

Get the process version string

Returns

version string

8.3.5.8 bigvoice_SetLoggerCallback()

```
void bigvoice_SetLoggerCallback (
            bigvoice_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 bigvoice_SetLogSeverity()

```
void bigvoice_SetLogSeverity (
            enum bigvoice_LogSeverity severity )
```

Change the global log severity

Logs above this level will not be printed.

Parameters

severity	Log severity

8.3.5.10 SOUND4_GetProcessInfo()

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

Get this processor description

Common to all SOUND4 processors.

Returns

the process description

8.4 Process creation/destruction

Classes

- struct bigvoice_CInstance
- struct bigvoice_CParameters

Functions

- struct bigvoice_CParameters * bigvoice_NewParameters ()
- void bigvoice_FreeParameters (struct bigvoice_CParameters *params)
- void bigvoice_SetParameter (struct bigvoice_CParameters *params, const char *name, const char *value)
- const char * bigvoice_GetParameter (struct bigvoice_CParameters *params, const char *name)
- void bigvoice_FreeParameterValue (const char *value)
- struct bigvoice_CInstance * bigvoice_InitProcess (const char *LoginKey, const char *RadioName, const char *Access_Key_ID, const char *Access_Key_Secret, const char *save_path)
- struct bigvoice_CInstance * bigvoice_InitProcess2 (const char *LoginKey, const char *RadioName, const char *Access_Key_ID, const char *Access_Key_Secret, const char *save_path, const struct bigvoice_CParameters *parameters)
- struct bigvoice_CInstance * bigvoice_InitProcess3 (const char *LoginKey, const char *RadioName, const char *Access_Key_ID, const char *Access_Key_Secret, const char *save_path, const struct bigvoice_CParameters *parameters, unsigned int frames_per_chunk)
- unsigned int bigvoice_GetProcessChunkFrames (struct bigvoice_CInstance *instance)
- int bigvoice_TerminateProcess (struct bigvoice_CInstance *instance)
- void bigvoice_ExitProcess (struct bigvoice_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 bigvoice_ExitProcess()

```
void bigvoice_ExitProcess (
            struct bigvoice_CInstance * instance )
```

Quit and destroy the instance

Parameters

instance	the instance to destroy

8.4.2.2 bigvoice_FreeParameters()

```
void bigvoice_FreeParameters (
            struct bigvoice_CParameters * params )
```

Free the parameters created with bigvoice_NewParameters.

8.4.2.3 bigvoice_FreeParameterValue()

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

Free a parameter value returned by bigvoice_GetParameter

Parameters

value	the returned value

8.4.2.4 bigvoice_GetParameter()

```
const char * bigvoice_GetParameter (
            struct bigvoice_CParameters * params,
            const char * name )
```

Get current parameter value.

Returned value must be freed with bigvoice_FreeParameterValue

8.4.2.5 bigvoice_GetProcessChunkFrames()

```
unsigned int bigvoice_GetProcessChunkFrames (
            struct bigvoice_CInstance * instance )
```

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

The value reflects the frames_per_chunk parameter passed to bigvoice_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 bigvoice_GetChannelCount samples

8.4.2.6 bigvoice_InitProcess()

```
struct bigvoice_CInstance * bigvoice_InitProcess (
            const char * LoginKey,
            const char * RadioName,
            const char * Access_Key_ID,
            const char * Access_Key_Secret,
            const char * save_path )
```

Create a process instance with default parameters (from env)

Same as bigvoice_InitProcess2, but with default parameters (from env)

Deprecated Use bigvoice_InitProcess3

8.4.2.7 bigvoice_InitProcess2()

```
struct bigvoice_CInstance * bigvoice_InitProcess2 (
            const char * LoginKey,
            const char * RadioName,
            const char * Access_Key_ID,
            const char * Access_Key_Secret,
            const char * save_path,
            const struct bigvoice_CParameters * parameters )
```

Create a process instance with default frames_per_chunk (12)

Same as bigvoice_InitProcess3, but with default frames_per_chunk

Deprecated Use bigvoice_InitProcess3

8.4.2.8 bigvoice_InitProcess3()

```
struct bigvoice_CInstance * bigvoice_InitProcess3 (
            const char * LoginKey,
            const char * RadioName,
            const char * Access_Key_ID,
            const char * Access_Key_Secret,
            const char * save_path,
            const struct bigvoice_CParameters * parameters,
            unsigned int frames_per_chunk )
```

Create a process instance

The output will have to be freed with bigvoice_ExitProcess

Attention

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

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

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

Web login initialization

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

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

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

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

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

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

License access

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

Parameters

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

Returns

A new process instance

8.4.2.9 bigvoice_NewParameters()

```
struct bigvoice_CParameters * bigvoice_NewParameters ( )
```

Create parameters to use with bigvoice_InitProcess2.

The output will have to be freed with bigvoice_FreeParameters

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

8.4.2.10 bigvoice_SetParameter()

```
void bigvoice_SetParameter (
            struct bigvoice_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

See also Instance's bus

8.4.2.11 bigvoice_TerminateProcess()

```
int bigvoice_TerminateProcess (
            struct bigvoice_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 Instance's bus

This allows instances to exchange real-time data.

Classes

- struct bigvoice_CBus

Functions

- struct bigvoice_CBus * bigvoice_NewBus ()
- void bigvoice_FreeBus (struct bigvoice_CBus *bus)
- void bigvoice_SetInstanceBus (struct bigvoice_CParameters *params, struct bigvoice_CBus *bus)

8.5.1 Detailed Description

The bus is used to manage Dominate-it between instances.

8.5.2 Function Documentation

8.5.2.1 bigvoice_FreeBus()

```
void bigvoice_FreeBus (
            struct bigvoice_CBus * bus )
```

Frees the bus allocated with bigvoice_NewBus

Parameters

bus	The bus to free

8.5.2.2 bigvoice_NewBus()

```
struct bigvoice_CBus * bigvoice_NewBus ( )
```

Creates a new bus

Returns

the new bus, to be freed with bigvoice_FreeBus

8.5.2.3 bigvoice_SetInstanceBus()

```
void bigvoice_SetInstanceBus (
            struct bigvoice_CParameters * params,
            struct bigvoice_CBus * bus )
```

Set the instance's bus. It can be used by instances to exchange real-time information if needed.

Attention

The bus should be created with the same library

Parameters

params	the parameters to configure
bus	the bus to use

8.6 Custom preset managagement

This allows to provide a customized preset management.

Typedefs

- typedef wchar_t fs_char
- typedef char ∗(∗ bigvoice_storage_reader) (const fs_char ∗filename, void ∗userdata)
- typedef void(∗ bigvoice_storage_reader_free) (char ∗content, void ∗userdata)
- typedef int(∗ bigvoice_storage_writer) (const fs_char ∗filename, const char ∗content, void ∗userdata)
- typedef int(∗ bigvoice_storage_exists) (const fs_char ∗filename, void ∗userdata)
- typedef fs_char ∗∗(∗ bigvoice_storage_getall) (void ∗userdata)
- typedef void(∗ bigvoice_storage_getall_free) (fs_char ∗∗all, void ∗userdata)
- typedef int(∗ bigvoice_storage_remove) (const fs_char ∗filename, void ∗userdata)
- typedef int(∗ bigvoice_storage_rename) (const fs_char ∗from, const fs_char ∗to, void ∗userdata)

Enumerations

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

Functions

- void bigvoice_SetPresetManager (struct bigvoice_CParameters ∗params, bigvoice_storage_reader reader, bigvoice_storage_reader_free, bigvoice_storage_writer writer, bigvoice_storage_exists exists, bigvoice_storage_getall getall, bigvoice_storage_getall_free getall_free, bigvoice_storage_remove remove, bigvoice_storage_rename rename, int IsReadOnly, void ∗userdata)
- void bigvoice_PresetManager_InformChange (struct bigvoice_CInstance ∗instance, const fs_char ∗relative↩ _path, enum bigvoice_PresetChange_Kind change_kind)

8.6.1 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 bigvoice_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)

8.6.2 Typedef Documentation

8.6.2.1 bigvoice_storage_exists

```
typedef int(* bigvoice_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 bigvoice_SetPresetManager

Returns

-1 if not exists, 0 if it exists

Definition at line 618 of file sound4.bigvoice.cl.h.

8.6.2.2 bigvoice_storage_getall

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

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

Parameters

userdata	the userdata passed to bigvoice_SetPresetManager

Returns

A NULL terminated array of file names

Definition at line 627 of file sound4.bigvoice.cl.h.

8.6.2.3 bigvoice_storage_getall_free

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

Function to call to free the result of getall.

Parameters

all	the returned value from bigvoice_storage_getall
userdata	the userdata passed to bigvoice_SetPresetManager

Definition at line 635 of file sound4.bigvoice.cl.h.

8.6.2.4 bigvoice_storage_reader

```
typedef char *(* bigvoice_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 bigvoice_SetPresetManager

Returns

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

Definition at line 591 of file sound4.bigvoice.cl.h.

8.6.2.5 bigvoice_storage_reader_free

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

Function to call to free the result of bigvoice_storage_reader.

Parameters

content	the returned value from bigvoice_storage_reader
userdata	the userdata passed to bigvoice_SetPresetManager

Definition at line 599 of file sound4.bigvoice.cl.h.

8.6.2.6 bigvoice_storage_remove

```
typedef int(* bigvoice_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 bigvoice_SetPresetManager

Returns

-1 if failed, 0 if OK

Definition at line 644 of file sound4.bigvoice.cl.h.

8.6.2.7 bigvoice_storage_rename

```
typedef int(* bigvoice_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 bigvoice_SetPresetManager

Returns

-1 if failed, 0 if OK

Definition at line 654 of file sound4.bigvoice.cl.h.

8.6.2.8 bigvoice_storage_writer

```
typedef int(* bigvoice_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 bigvoice_SetPresetManager

Returns

-1 if failed, 0 if OK

Definition at line 609 of file sound4.bigvoice.cl.h.

8.6.2.9 fs_char

```
typedef wchar_t fs_char
```

Definition at line 579 of file sound4.bigvoice.cl.h.

8.6.3 Enumeration Type Documentation

8.6.3.1 bigvoice_PresetChange_Kind

```
enum bigvoice_PresetChange_Kind
```

Kind of change on a storage path

Enumerator

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

Definition at line 687 of file sound4.bigvoice.cl.h.

8.6.4 Function Documentation

8.6.4.1 bigvoice_PresetManager_InformChange()

```
void bigvoice_PresetManager_InformChange (
            struct bigvoice_CInstance * instance,
            const fs_char * relative_path,
            enum bigvoice_PresetChange_Kind change_kind )
```

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

Parameters

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

8.6.4.2 bigvoice_SetPresetManager()

```
void bigvoice_SetPresetManager (
            struct bigvoice_CParameters * params,
            bigvoice_storage_reader reader,
            bigvoice_storage_reader_free ,
            bigvoice_storage_writer writer,
            bigvoice_storage_exists exists,
            bigvoice_storage_getall getall,
            bigvoice_storage_getall_free getall_free,
            bigvoice_storage_remove remove,
            bigvoice_storage_rename rename,
            int IsReadOnly,
            void * userdata )
```

Setup a custom preset manager. This overrides the save_path in bigvoice_InitProcess2

To dynamically update when storage change, see bigvoice_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

8.7 Metadata Management

Functions

- void bigvoice_SetMetadata (struct bigvoice_CInstance *instance, const char *key, const char *value)
- void bigvoice_SetMetadataMulti (struct bigvoice_CInstance *instance, const char **keyvalue)
- const char ** bigvoice_GetMetadataInfos (struct bigvoice_CInstance *instance)
- void bigvoice_FreeMetadataInfos (struct bigvoice_CInstance *instance, const char **infos)

8.7.1 Detailed Description

Metadata handling.

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

8.7.2 Function Documentation

8.7.2.1 bigvoice_FreeMetadataInfos()

```
void bigvoice_FreeMetadataInfos (
            struct bigvoice_CInstance * instance,
            const char ** infos )
```

Frees the information returned by bigvoice_GetMetadataInfos.

Parameters

instance	the instance
infos	the infos

8.7.2.2 bigvoice_GetMetadataInfos()

```
const char ** bigvoice_GetMetadataInfos (
            struct bigvoice_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 bigvoice_FreeMetadataInfos

8.7.2.3 bigvoice_SetMetadata()

```
void bigvoice_SetMetadata (
            struct bigvoice_CInstance * instance,
            const char * key,
            const char * value )
```

Adds the specific metadata to the process.

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

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

See also

bigvoice_SetMetadataMulti

Parameters

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

8.7.2.4 bigvoice_SetMetadataMulti()

```
void bigvoice_SetMetadataMulti (
            struct bigvoice_CInstance * instance,
            const char ** keyvalue )
```

Adds multiple metadata to the process.

This works like bigvoice_SetMetadata but with multiple metadata at once.

See also

bigvoice_SetMetadata

Parameters

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

8.8 Update Management

Functions

- void bigvoice_StartUpdateThread (struct bigvoice_CInstance ∗instance, unsigned int port)
- void bigvoice_StopUpdateThread (struct bigvoice_CInstance ∗instance)
- int bigvoice_WaitUpdateThreadReady (struct bigvoice_CInstance ∗instance, int milliseconds)

8.8.1 Detailed Description

Manage the process update.

8.8.2 Function Documentation

8.8.2.1 bigvoice_StartUpdateThread()

```
void bigvoice_StartUpdateThread (
            struct bigvoice_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 bigvoice_StopUpdateThread

Parameters

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

8.8.2.2 bigvoice_StopUpdateThread()

```
void bigvoice_StopUpdateThread (
            struct bigvoice_CInstance * instance )
```

Stop the update thread

Parameters

instance	the processing instance

8.8.2.3 bigvoice_WaitUpdateThreadReady()

```
int bigvoice_WaitUpdateThreadReady (
            struct bigvoice_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

8.9 Direct processing

Functions

- void bigvoice_ProcessAudio (struct bigvoice_CInstance *instance, const float *input, float *output)
- void bigvoice_ProcessAudio_Planar (struct bigvoice_CInstance *instance, float const *const *input, float *const *output)
- float * bigvoice_GetBufferIn (struct bigvoice_CInstance *instance)
- float * bigvoice_GetBufferOut (struct bigvoice_CInstance *instance)
- unsigned int bigvoice_GetEstimatedDelay (struct bigvoice_CInstance *instance)

8.9.1 Detailed Description

Optimized processing with direct in->out.

8.9.2 Function Documentation

8.9.2.1 bigvoice_GetBufferIn()

```
float * bigvoice_GetBufferIn (
            struct bigvoice_CInstance * instance )
```

Get preallocated aligned buffer for audio input for bigvoice_ProcessAudio()

The buffer is a float[256 * bigvoice_GetAudioInputCount() * bigvoice_GetChannelCount()]

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

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.

8.9.2.2 bigvoice_GetBufferOut()

```
float * bigvoice_GetBufferOut (
            struct bigvoice_CInstance * instance )
```

Get preallocated aligned buffer for audio output for bigvoice_ProcessAudio()

The buffer is a float[256 * bigvoice_GetAudioOutputCount() * bigvoice_GetChannelCount()]

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

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.

8.9.2.3 bigvoice_GetEstimatedDelay()

```
unsigned int bigvoice_GetEstimatedDelay (
            struct bigvoice_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

8.9.2.4 bigvoice_ProcessAudio()

```
void bigvoice_ProcessAudio (
            struct bigvoice_CInstance * instance,
            const float * input,
            float * output )
```

Process an audio chunk

A chunk contains bigvoice_GetProcessChunkFrames frames, set at init by frames_per_chunk in bigvoice_InitProcess3

Input has bigvoice_GetAudioInputCount() ∗ bigvoice_GetChannelCount() channels

Ouput has bigvoice_GetAudioOutputCount() ∗ bigvoice_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 bigvoice_GetBufferIn
output	audio output buffer (64 byte aligned) or NULL to use internal buffer bigvoice_GetBufferOut

8.9.2.5 bigvoice_ProcessAudio_Planar()

```
void bigvoice_ProcessAudio_Planar (
            struct bigvoice_CInstance * instance,
            float const *const * input,
            float *const * output )
```

Process an audio chunk of planar audio

A chunk contains bigvoice_GetProcessChunkFrames frames, set at init by frames_per_chunk in bigvoice_InitProcess3

Input has bigvoice_GetAudioInputCount() * bigvoice_GetChannelCount() channels

Ouput has bigvoice_GetAudioOutputCount() * bigvoice_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)

8.10 Audio managing (any audio)

Enumerations

- enum bigvoice_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 * bigvoice_GetFormatName (const enum bigvoice_SampleFormat fmt)
- enum bigvoice_SampleFormat bigvoice_GetFormatFromName (const char *name)
- unsigned int bigvoice_GetBytesFromFormat (const enum bigvoice_SampleFormat fmt)
- unsigned int bigvoice_GetMaxPacketFrame (struct bigvoice_CInstance *instance)

- unsigned int bigvoice_AddAudio (struct bigvoice_CInstance *instance, const uint8_t *payload, unsigned int nFrame, enum bigvoice_SampleFormat fmt)
- unsigned int bigvoice_AddPadAudio (struct bigvoice_CInstance *instance)
- unsigned int bigvoice_GetOutputCount (struct bigvoice_CInstance *instance)
- unsigned int bigvoice_GetAudio (struct bigvoice_CInstance *instance, uint8_t *payload, unsigned int max_nFrame, enum bigvoice_SampleFormat fmt)

8.10.1 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.

8.10.2 Enumeration Type Documentation

8.10.2.1 bigvoice_SampleFormat

```
enum bigvoice_SampleFormat
```

Supported formats for audio I/O

Enumerator

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

Definition at line 910 of file sound4.bigvoice.cl.h.

8.10.3 Function Documentation

8.10.3.1 bigvoice_AddAudio()

```
unsigned int bigvoice_AddAudio (
            struct bigvoice_CInstance * instance,
```

```
            const uint8_t * payload,
            unsigned int nFrame,
            enum bigvoice_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

8.10.3.2 bigvoice_AddPadAudio()

```
unsigned int bigvoice_AddPadAudio (
            struct bigvoice_CInstance * instance )
```

Add silence to finish the current audio chunk and process it

Parameters

instance	the processing instance

Returns

number of added silence frame

8.10.3.3 bigvoice_GetAudio()

```
unsigned int bigvoice_GetAudio (
            struct bigvoice_CInstance * instance,
            uint8_t * payload,
            unsigned int max_nFrame,
            enum bigvoice_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

8.10.3.4 bigvoice_GetBytesFromFormat()

```
unsigned int bigvoice_GetBytesFromFormat (
            const enum bigvoice_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

8.10.3.5 bigvoice_GetFormatFromName()

```
enum bigvoice_SampleFormat bigvoice_GetFormatFromName (
            const char * name )
```

Convert string to format

Parameters

name	the format string

Returns

the matching format, or INVALID_FORMAT if invalid.

8.10.3.6 bigvoice_GetFormatName()

```
const char * bigvoice_GetFormatName (
            const enum bigvoice_SampleFormat fmt )
```

Convert format to string

Parameters

fmt	the format

Returns

the string name of the format

8.10.3.7 bigvoice_GetMaxPacketFrame()

```
unsigned int bigvoice_GetMaxPacketFrame (
            struct bigvoice_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.

8.10.3.8 bigvoice_GetOutputCount()

```
unsigned int bigvoice_GetOutputCount (
            struct bigvoice_CInstance * instance )
```

Get the available output size

Parameters

instance	the processing instance

Returns

number of available frames at output

8.11 Audio convert helpers

Functions

- void bigvoice_AudioConvertFrom (const uint8_t ∗payload, float ∗output, size_t nSpl, enum bigvoice_SampleFormat fmt)
- void bigvoice_AudioConvertTo (const float ∗input, uint8_t ∗payload, size_t nSpl, enum bigvoice_SampleFormat fmt)
- void bigvoice_StereoToMono (const float ∗input, float ∗output, size_t nFrame)
- void bigvoice_MonoToStereo (const float ∗input, float ∗output, size_t nFrame)
- void bigvoice_StereoToMono_Planar (const float ∗inputL, const float ∗inputR, float ∗output, size_t nFrame)
- void bigvoice_MonoToStereo_Planar (const float ∗input, float ∗outputL, float ∗outputR, size_t nFrame)
- void bigvoice_AudioMonoFromLiveStereo (const uint8_t ∗payload, float ∗output)
- void bigvoice_AudioMonoToLiveStereo (const float ∗input, uint8_t ∗payload)

8.11.1 Detailed Description

8.11.2 Function Documentation

8.11.2.1 bigvoice_AudioConvertFrom()

```
void bigvoice_AudioConvertFrom (
            const uint8_t * payload,
            float * output,
            size_t nSpl,
            enum bigvoice_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

8.11.2.2 bigvoice_AudioConvertTo()

```
void bigvoice_AudioConvertTo (
            const float * input,
            uint8_t * payload,
            size_t nSpl,
            enum bigvoice_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

8.11.2.3 bigvoice_AudioMonoFromLiveStereo()

```
void bigvoice_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)

8.11.2.4 bigvoice_AudioMonoToLiveStereo()

```
void bigvoice_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

8.11.2.5 bigvoice_MonoToStereo()

```
void bigvoice_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

8.11.2.6 bigvoice_MonoToStereo_Planar()

```
void bigvoice_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

8.11.2.7 bigvoice_StereoToMono()

```
void bigvoice_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

8.11.2.8 bigvoice_StereoToMono_Planar()

```
void bigvoice_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

8.12 Client's JSon config interfacing

Classes

- struct bigvoice_CClientInstance

Functions

- struct bigvoice_CClientInstance * bigvoice_NewClient (struct bigvoice_CInstance *instance)
- void bigvoice_DeleteClient (struct bigvoice_CClientInstance *client)
- const char * bigvoice_ProcessJson (struct bigvoice_CClientInstance *client, const char *json_str, int *need_save)
- void bigvoice_FreeJsonAnswer (const char *json_str)
- int bigvoice_SaveState (struct bigvoice_CInstance *instance)

8.12.1 Detailed Description

Allow low-level JSON configuration. see also JSON syntax

8.12.2 Function Documentation

8.12.2.1 bigvoice_DeleteClient()

```
void bigvoice_DeleteClient (
            struct bigvoice_CClientInstance * client )
```

Deletes a client

Parameters

client	the client instance

8.12.2.2 bigvoice_FreeJsonAnswer()

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

Free a JSON answer returned by bigvoice_ProcessJson

Parameters

json_str	the JSON request

8.12.2.3 bigvoice_NewClient()

```
struct bigvoice_CClientInstance * bigvoice_NewClient (
            struct bigvoice_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

8.12.2.4 bigvoice_ProcessJson()

```
const char * bigvoice_ProcessJson (
            struct bigvoice_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 bigvoice_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 bigvoice_FreeJsonAnswer)

8.12.2.5 bigvoice_SaveState()

```
int bigvoice_SaveState (
            struct bigvoice_CInstance * instance )
```

Save the current state

Needed when bigvoice_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

8.13 JSON syntax

JSON protocol to configure the process

8.13.1 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 BigVoice.cl version <= 0.9.7, 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

8.13.2 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.

8.13.3 Examples

8.13.3.1 Get status

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

answer:

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

8.13.3.2 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":{...}},
       ...
       }
```

8.13.3.3 Check license

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

answer:

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

8.13.3.4 Set Processor name

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

answer:

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

8.13.3.5 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"},
            ...
       ]}}
```

8.13.3.6 Load a preset

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

answer:

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

8.13.3.7 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"}}}
```

8.13.3.8 Set input gain

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

answer:

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

8.13.3.9 Set bypass

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

answer:

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

8.13.3.10 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

8.13.3.11 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"}
```

8.14 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.

8.14.1 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}
       }}}
```

8.14.2 Authentication management

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

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

8.14.2.1 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.

8.14.2.2 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

8.14.2.3 Change setup account

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

8.14.2.3.1 Set the setup/admin account

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

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

8.14.2.3.2 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"}}
```

8.14.2.3.3 Remove the setup/admin account

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

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

8.14.2.4 Get user info

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

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

8.14.2.5 Add or modify a user

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

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

8.14.2.6 Removes a user

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

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

8.14.2.7 List users

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

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

8.14.2.8 Get/set unlogged access

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

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

8.14.2.9 Get realm

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

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

8.14.2.10 Get API version

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

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

8.14.2.11 Get setup is admin

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

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

8.14.2.12 Get/set setup account

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

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

8.14.2.13 Get/set custom storage

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

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

8.14.3 Get process' interface description

8.14.3.1 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 !

8.14.3.2 Get Process Image

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

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

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

8.14.3.3 Get Process Ico

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

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

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

8.15 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 bigvoice_Webserver_tcp (unsigned int listenport, unsigned int listenport_secure, const char *socket_ip, unsigned int socket_port)
- uint64_t bigvoice_Webserver_tcp2 (unsigned int listenport, unsigned int listenport_secure, const char *socket_ip, unsigned int socket_port, const struct bigvoice_CParameters *parameters)
- uint64_t bigvoice_Webserver (unsigned int listenport, unsigned int listenport_secure, struct bigvoice_CInstance *instance)
- int bigvoice_Webserver_Stop (uint64_t id, int timeout_ms)
- int bigvoice_Webserver_Status (uint64_t id)
- void bigvoice_Webserver_SetAppHealth (struct bigvoice_CInstance *instance, int httpcode, const char *contenttype, const char *content)
- void bigvoice_Webserver_GetAppHealth (struct bigvoice_CInstance *instance, int *httpcode, char **contenttype, char **content)
- void bigvoice_Webserver_FreeString (char *str)

8.15.1 Detailed Description

Run a web server to configure the processor

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

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

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

8.15.2 Macro Definition Documentation

8.15.2.1 SOUND4_INVALID_WEBSERVER_ID

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

This represents an invalid webserver id bigvoice_Webserver and bigvoice_Webserver_tcp will never return this value

Definition at line 1591 of file sound4.bigvoice.cl.h.

8.15.2.2 SOUND4_WEBSERVER_HTTP_OK

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

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

Definition at line 1596 of file sound4.bigvoice.cl.h.

8.15.2.3 SOUND4_WEBSERVER_HTTPS_OK

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

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

Definition at line 1600 of file sound4.bigvoice.cl.h.

8.15.3 Function Documentation

8.15.3.1 bigvoice_Webserver()

```
uint64_t bigvoice_Webserver (
            unsigned int listenport,
            unsigned int listenport_secure,
            struct bigvoice_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 bigvoice_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 bigvoice_Webserver_Stop

8.15.3.2 bigvoice_Webserver_FreeString()

```
void bigvoice_Webserver_FreeString (
            char * str )
```

Free a a string returned by bigvoice_Webserver_GetAppHealth

Parameters

str	the string to free

8.15.3.3 bigvoice_Webserver_GetAppHealth()

```
void bigvoice_Webserver_GetAppHealth (
            struct bigvoice_CInstance * instance,
            int * httpcode,
            char ** contenttype,
            char ** content )
```

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

Parameters

instance	The process instance
httpcode	The http code pointer
contenttype	The http Content-Type header pointer (to be freed with bigvoice_Webserver_FreeString)
content	The http body content pointer (to be freed with bigvoice_Webserver_FreeString)

8.15.3.4 bigvoice_Webserver_SetAppHealth()

```
void bigvoice_Webserver_SetAppHealth (
            struct bigvoice_CInstance * instance,
            int httpcode,
            const char * contenttype,
            const char * content )
```

Set the web server /apphealth status info

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

Parameters

instance	The process instance
httpcode	The http code
contenttype	The http Content-Type header
content	The http body content

8.15.3.5 bigvoice_Webserver_Status()

```
int bigvoice_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 bigvoice_Webserver_tcp or bigvoice_Webserver

Returns

a combinaison of SOUND4_WEBSERVER_HTTP_OK and SOUND4_WEBSERVER_HTTPS_OK depending on the server status

8.15.3.6 bigvoice_Webserver_Stop()

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

Stop a web server started with bigvoice_Webserver_tcp

Warning

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

Parameters

id	The web server Id, returned by bigvoice_Webserver_tcp or bigvoice_Webserver
timeout_ms	The timeout to wait gracefull shutdown

Returns

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

8.15.3.7 bigvoice_Webserver_tcp()

```
uint64_t bigvoice_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 bigvoice_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 bigvoice_Webserver_Stop

8.15.3.8 bigvoice_Webserver_tcp2()

```
uint64_t bigvoice_Webserver_tcp2 (
            unsigned int listenport,
            unsigned int listenport_secure,
            const char * socket_ip,
            unsigned int socket_port,
            const struct bigvoice_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 bigvoice_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 bigvoice_NewParameters

Returns

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

8.16 Sample using C++ interface wrapper

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

8.16.1 Setup and sanity checks

```
sound4::bigvoice::SetLogSeverity(verbose5);
sound4::bigvoice::SetLoggerCallback(MyLogger);

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

8.16.2 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;
}
```

8.16.3 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;
```

8.16.4 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;
}
```

8.16.5 Sample processing loop

```
while (!stopthread)
{
    // Read a chunk from stdin
    if (std::fread(instance.GetBufferIn().data(), sizeof(float), sound4::bigvoice::InputSampleSize,
  stdin) != sound4::bigvoice::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::bigvoice::OutputSampleSize,
  stdout) != sound4::bigvoice::OutputSampleSize) {
        std::cerr « "Failed to write audio output" « std::endl;
        stopthread=true;
    }
}
```

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

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

8.17.1 Dynamic loading

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

8.17.2 Setup and sanity checks

```
sound4::bigvoice::dyn::SetLogSeverity(dynlib, verbose5);
sound4::bigvoice::dyn::SetLoggerCallback(dynlib, MyLogger);

sound4::bigvoice::dyn::CInstance instance(dynlib);
instance.SetParam("ADMIN_USER","admin");
instance.SetParam("ADMIN_SECRET","admin");
```

8.17.3 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;
}
```

8.17.4 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;
```

8.17.5 Starting a web-server

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

8.17.6 Sample processing loop

```
    while (!stopthread)
    {
        // Read a chunk from stdin
        if (std::fread(instance.GetBufferIn().data(), sizeof(float), sound4::bigvoice::dyn::InputSampleSize,
      stdin) != sound4::bigvoice::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::bigvoice::dyn::OutputSampleSize,
      stdout) != sound4::bigvoice::dyn::OutputSampleSize) {
            std::cerr « "Failed to write audio output" « std::endl;
            stopthread=true;
        }
    }
```

8.17.7 Custom preset manager

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

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

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

8.17.8 Custom preset manager loading

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

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

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

It uses sound4cl.hpp header.

See full example cpp/sound4cl-pipe.cpp

8.18.1 Loading the library

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

8.18.2 Setup and configure

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

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

8.18.3 Create the processing instance

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

8.18.4 Processing loop

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

8.18.5 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;
}
```

8.18.6 Communicating with the instance

See also

JSON syntax

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

8.18.7 Custom preset manager example

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

    bool SetupPath(const std::filesystem::path& a_preset_path)
        {
            m_preset_path=a_preset_path;
            std::error_code ec;
            if ( !std::filesystem::exists(m_preset_path,ec ) ) {
                return false;
            }
            if ( !std::filesystem::is_directory(m_preset_path, ec) ) {
                return false;
            }
            return true;
        }
    virtual bool IsReadOnly() override { return false; };
    virtual bool Exists(const std::filesystem::path &name) override
        {
            std::error_code ec;
            return std::filesystem::exists(GetRealPath(name), ec);
        }
    virtual bool Remove(const std::filesystem::path &name) override
        {
            std::error_code ec;
            return std::filesystem::remove(GetRealPath(name), ec);
        }
    virtual bool Rename(const std::filesystem::path &from, const std::filesystem::path &to) override
        {
            std::error_code ec;
            std::filesystem::rename(GetRealPath(from), GetRealPath(to), ec);
            return (!ec);
        }
```

```
    virtual std::vector<std::filesystem::path> GetAll() override
        {
            std::vector<std::filesystem::path> list;
            std::error_code ec;
            for (auto it = std::filesystem::directory_iterator(m_preset_path.c_str(),
      std::filesystem::directory_options::skip_permission_denied, ec);
                !ec && it != std::filesystem::directory_iterator{};
                 it.increment(ec) )
            {
                if (it->is_regular_file(ec)) {
                    list.push_back(it->path().lexically_relative(m_preset_path));
                }
            }
            return list;
        }
    virtual std::string Read(const std::filesystem::path &filename) override
        {
            std::ifstream f(GetRealPath(filename));
            if ( f.fail() ) {
                return std::string();
            }
            std::ostringstream ss;
            ss « f.rdbuf();
            return ss.str();
        }
    virtual bool Write(const std::filesystem::path &filename, const std::string &content) override
        {
            std::ofstream f(GetRealPath(filename));
            if ( f.fail() ) {
                return false;
            }
            f « content;
            if ( f.fail() ) {
                return false;
            }
            return true;
        }
private:
    std::filesystem::path m_preset_path;
    std::filesystem::path GetRealPath(const std::filesystem::path &name)
        {
            return m_preset_path / name;
        }
};
```

8.18.8 Custom preset manager loading

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

Chapter 9

Namespace Documentation

9.1 dyn Namespace Reference

9.1.1 Detailed Description

Dynamic loading namespace

9.2 sound4 Namespace Reference

Namespaces

- namespace bigvoice
- namespace helper

Classes

- class CBus

  Shared Bus.
- class CInstance

  Instance handling class.
- class CPresetLoader

  Custom preset handler helper.
- class CProcessor

  Dynamic library interface.

Typedefs

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

Variables

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

9.2.1 Detailed Description

SOUND4 namespace

9.2.2 Typedef Documentation

9.2.2.1 log_cb_t

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

Log callback type

Definition at line 258 of file sound4cl.hpp.

9.2.2.2 LogSeverity

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

Log Severity levels

Definition at line 243 of file sound4cl.hpp.

9.2.2.3 PresetChange_Kind

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

Definition at line 753 of file sound4cl.hpp.

9.2.3 Variable Documentation

9.2.3.1 error

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

Definition at line 246 of file sound4cl.hpp.

9.2.3.2 fatal

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

Definition at line 245 of file sound4cl.hpp.

9.2.3.3 info

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

Definition at line 248 of file sound4cl.hpp.

9.2.3.4 none

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

Definition at line 244 of file sound4cl.hpp.

9.2.3.5 verbose

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

Definition at line 249 of file sound4cl.hpp.

9.2.3.6 verbose2

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

Definition at line 250 of file sound4cl.hpp.

9.2.3.7 verbose3

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

Definition at line 251 of file sound4cl.hpp.

9.2.3.8 verbose4

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

Definition at line 252 of file sound4cl.hpp.

9.2.3.9 verbose5

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

Definition at line 253 of file sound4cl.hpp.

9.2.3.10 warning

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

Definition at line 247 of file sound4cl.hpp.

9.3 sound4::bigvoice Namespace Reference

Namespaces

- namespace dyn
- namespace helper

Classes

- class CBus
- class CInstance
- class CPresetLoader

Typedefs

- using SampleFormat = bigvoice_SampleFormat
- using LogSeverity = bigvoice_LogSeverity
- using log_cb_t = std::function< void(LogSeverity, const std::string &)>
- using PresetChange_Kind = bigvoice_PresetChange_Kind

Variables

- const size_t ChunkFrames = 12
- const size_t InputChannels = 1 * 1
- const size_t OutputChannels = 1 * 1
- const size_t InputSampleSize =ChunkFrames*InputChannels
- const size_t OutputSampleSize =ChunkFrames*OutputChannels
- const size_t SampleRate = 48000

9.3.1 Detailed Description

bigvoice namespace

9.3.2 Typedef Documentation

9.3.2.1 log_cb_t

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

Log callback type

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

9.3.2.2 LogSeverity

```
using sound4::bigvoice::LogSeverity = typedef bigvoice_LogSeverity
```

Log Severity levels

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

9.3.2.3 PresetChange_Kind

```
using sound4::bigvoice::PresetChange_Kind = typedef bigvoice_PresetChange_Kind
```

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

9.3.2.4 SampleFormat

```
using sound4::bigvoice::SampleFormat = typedef bigvoice_SampleFormat
```

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

9.3.3 Variable Documentation

9.3.3.1 ChunkFrames

```
const size_t sound4::bigvoice::ChunkFrames = 12
```

Number of frames in a chunk

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

9.3.3.2 InputChannels

```
const size_t sound4::bigvoice::InputChannels = 1 * 1
```

Number of float values in an input frame

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

9.3.3.3 InputSampleSize

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

Number of float values for an input chunk

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

9.3.3.4 OutputChannels

```
const size_t sound4::bigvoice::OutputChannels = 1 * 1
```

Number of float values in an output frame

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

9.3.3.5 OutputSampleSize

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

Number of float values for an output chunk

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

9.3.3.6 SampleRate

```
const size_t sound4::bigvoice::SampleRate = 48000
```

Sample rate of library

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

9.4 sound4::bigvoice::dyn Namespace Reference

Namespaces

- namespace helper

Classes

- class CBus
- class CDynLib

  Dynamic library interface.
- class CInstance
- class CPresetLoader

Typedefs

- using SampleFormat = bigvoice_SampleFormat
- using LogSeverity = bigvoice_LogSeverity
- using log_cb_t = std::function< void(LogSeverity, const std::string &)>
- using PresetChange_Kind = bigvoice_PresetChange_Kind

Variables

- const size_t ChunkFrames = 12
- const size_t InputChannels = 1 ∗ 1
- const size_t OutputChannels = 1 ∗ 1
- const size_t InputSampleSize =ChunkFrames∗InputChannels
- const size_t OutputSampleSize =ChunkFrames∗OutputChannels
- const size_t SampleRate = 48000

9.4.1 Typedef Documentation

9.4.1.1 log_cb_t

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

Log callback type

Definition at line 743 of file sound4.bigvoice.cl_dyn.hpp.

9.4.1.2 LogSeverity

```
using sound4::bigvoice::dyn::LogSeverity = typedef bigvoice_LogSeverity
```

Log Severity levels

Definition at line 739 of file sound4.bigvoice.cl_dyn.hpp.

9.4.1.3 PresetChange_Kind

```
using sound4::bigvoice::dyn::PresetChange_Kind = typedef bigvoice_PresetChange_Kind
```

Definition at line 811 of file sound4.bigvoice.cl_dyn.hpp.

9.4.1.4 SampleFormat

```
using sound4::bigvoice::dyn::SampleFormat = typedef bigvoice_SampleFormat
```

Definition at line 472 of file sound4.bigvoice.cl_dyn.hpp.

9.4.2 Variable Documentation

9.4.2.1 ChunkFrames

```
const size_t sound4::bigvoice::dyn::ChunkFrames = 12
```

Number of frames in a chunk

Definition at line 476 of file sound4.bigvoice.cl_dyn.hpp.

9.4.2.2 InputChannels

```
const size_t sound4::bigvoice::dyn::InputChannels = 1 * 1
```

Number of float values in an input frame

Definition at line 480 of file sound4.bigvoice.cl_dyn.hpp.

9.4.2.3 InputSampleSize

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

Number of float values for an input chunk

Definition at line 488 of file sound4.bigvoice.cl_dyn.hpp.

9.4.2.4 OutputChannels

```
const size_t sound4::bigvoice::dyn::OutputChannels = 1 * 1
```

Number of float values in an output frame

Definition at line 484 of file sound4.bigvoice.cl_dyn.hpp.

9.4.2.5 OutputSampleSize

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

Number of float values for an output chunk

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

9.4.2.6 SampleRate

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

Sample rate of library

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

9.5 sound4::bigvoice::dyn::helper Namespace Reference

Classes

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

Typedefs

- using DynLib_t = HMODULE
- using DynFunc_t = FARPROC

9.5.1 Detailed Description

helper functions for unicode strings in Windows

9.5.2 Typedef Documentation

9.5.2.1 DynFunc_t

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

Definition at line 83 of file sound4.bigvoice.cl_dyn.hpp.

9.5.2.2 DynLib_t

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

Definition at line 82 of file sound4.bigvoice.cl_dyn.hpp.

9.6 sound4::bigvoice::helper Namespace Reference

Classes

- struct SampleFormat

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

9.6.1 Detailed Description

helper functions for unicode strings in Windows

9.7 sound4::helper Namespace Reference

Classes

- class CDynLoader

  Helper to load dynamic library.
- class DynFuncHolder

  Helper to wrap a function pointer.
- struct SampleFormat

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

Typedefs

- using DynLib_t = HMODULE
- using DynFunc_t = FARPROC

9.7.1 Detailed Description

helper functions for unicode strings in Windows

9.7.2 Typedef Documentation

9.7.2.1 DynFunc_t

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

Definition at line 112 of file sound4cl.hpp.

9.7.2.2 DynLib_t

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

Definition at line 111 of file sound4cl.hpp.

9.8 std Namespace Reference

Chapter 10

Class Documentation

10.1 bigvoice_CBus Struct Reference

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

10.1.1 Detailed Description

Abstract holder for a instances' bus

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

- sound4.bigvoice.cl.h

10.2 bigvoice_CClientInstance Struct Reference

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

10.2.1 Detailed Description

Abstract pointer for a client instance

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

- sound4.bigvoice.cl.h

10.3 bigvoice_CInstance Struct Reference

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

10.3.1 Detailed Description

Abstract holder for a process instance

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

- sound4.bigvoice.cl.h

10.4 bigvoice_CParameters Struct Reference

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

10.4.1 Detailed Description

Abstract holder for a process instance parameters

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

- sound4.bigvoice.cl.h

10.5 sound4::bigvoice::CBus Class Reference

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

Public Member Functions

- CBus ()
- ~CBus ()
- bigvoice_CBus * Get () const

10.5.1 Detailed Description

Definition at line 348 of file sound4.bigvoice.cl.hpp.

10.5.2 Constructor & Destructor Documentation

10.5.2.1 CBus()

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

Definition at line 350 of file sound4.bigvoice.cl.hpp.

10.5.2.2 ~CBus()

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

Definition at line 352 of file sound4.bigvoice.cl.hpp.

10.5.3 Member Function Documentation

10.5.3.1 Get()

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

Definition at line 357 of file sound4.bigvoice.cl.hpp.

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

- sound4.bigvoice.cl.hpp

10.6 sound4::bigvoice::dyn::CBus Class Reference

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

Public Member Functions

- CBus (CDynLib &dynlib)
- ~CBus ()
- bigvoice_CBus * Get () const

10.6.1 Detailed Description

Definition at line 771 of file sound4.bigvoice.cl_dyn.hpp.

10.6.2 Constructor & Destructor Documentation

10.6.2.1 CBus()

```
sound4::bigvoice::dyn::CBus::CBus (
            CDynLib & dynlib )  [inline]
```

Definition at line 773 of file sound4.bigvoice.cl_dyn.hpp.

10.6.2.2 ~CBus()

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

Definition at line 780 of file sound4.bigvoice.cl_dyn.hpp.

10.6.3 Member Function Documentation

10.6.3.1 Get()

```
bigvoice_CBus * sound4::bigvoice::dyn::CBus::Get ( ) const  [inline]
```

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

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

- sound4.bigvoice.cl_dyn.hpp

10.7 sound4::CBus Class Reference

Shared Bus.

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

Public Member Functions

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

10.7.1 Detailed Description

Some processor can share a bus for realtime data exchange

Definition at line 713 of file sound4cl.hpp.

10.7.2 Constructor & Destructor Documentation

10.7.2.1 CBus()

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

Definition at line 715 of file sound4cl.hpp.

10.7.2.2 ~CBus()

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

Definition at line 722 of file sound4cl.hpp.

10.7.3 Member Function Documentation

10.7.3.1 Get()

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

Definition at line 729 of file sound4cl.hpp.

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

- sound4cl.hpp

10.8 sound4::bigvoice::CInstance::CClient Class Reference

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

Public Member Functions

- CClient (bigvoice_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.

10.8.1 Detailed Description

Low-level JSON client.

See also

JSON syntax

Definition at line 827 of file sound4.bigvoice.cl.hpp.

10.8.2 Constructor & Destructor Documentation

10.8.2.1 CClient() [1/2]

```
sound4::bigvoice::CInstance::CClient::CClient (
            bigvoice_CInstance * instance )  [inline]
```

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

10.8.2.2 CClient() [2/2]

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

10.8.2.3 ~CClient()

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

Definition at line 838 of file sound4.bigvoice.cl.hpp.

10.8.3 Member Function Documentation

10.8.3.1 operator=()

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

10.8.3.2 ProcessJson()

```
std::string sound4::bigvoice::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 852 of file sound4.bigvoice.cl.hpp.

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

- sound4.bigvoice.cl.hpp

10.9 sound4::bigvoice::dyn::CInstance::CClient Class Reference

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

Public Member Functions

- CClient (bigvoice_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.

10.9.1 Detailed Description

Low-level JSON client.

See also

JSON syntax

Definition at line 1282 of file sound4.bigvoice.cl_dyn.hpp.

10.9.2 Constructor & Destructor Documentation

10.9.2.1 CClient() [1/2]

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

Definition at line 1284 of file sound4.bigvoice.cl_dyn.hpp.

10.9.2.2 CClient() [2/2]

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

10.9.2.3 ~CClient()

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

Definition at line 1294 of file sound4.bigvoice.cl_dyn.hpp.

10.9.3 Member Function Documentation

10.9.3.1 operator=()

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

10.9.3.2 ProcessJson()

```
std::string sound4::bigvoice::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 1308 of file sound4.bigvoice.cl_dyn.hpp.

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

- sound4.bigvoice.cl_dyn.hpp

10.10 sound4::CInstance::CClient Class Reference

`#include <sound4cl.hpp>`

Public Member Functions

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

  Process a JSON request and returns the answer.

10.10.1 Detailed Description

Low-level JSON client.

See also

JSON syntax

Definition at line 1225 of file sound4cl.hpp.

10.10.2 Constructor & Destructor Documentation

10.10.2.1 CClient() [1/2]

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

Definition at line 1227 of file sound4cl.hpp.

10.10.2.2 CClient() [2/2]

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

10.10.2.3 ~CClient()

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

Definition at line 1237 of file sound4cl.hpp.

10.10.3 Member Function Documentation

10.10.3.1 operator=()

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

10.10.3.2 ProcessJson()

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

Parameters

request	the JSON request.

See also

JSON syntax

Parameters

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

Returns

the JSON answer

Definition at line 1251 of file sound4cl.hpp.

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

- sound4cl.hpp

10.11 sound4::bigvoice::dyn::CDynLib Class Reference

Dynamic library interface.

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

Public Member Functions

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

  Loads the library.
- bool IsOk () const

  Check if the library was loaded correctly.

Public Attributes

- helper::DynFuncHolder< decltype(bigvoice_GetVersion) > GetVersion
- helper::DynFuncHolder< decltype(bigvoice_GetChunkSizeInFrames) > GetChunkSizeInFrames
- helper::DynFuncHolder< decltype(bigvoice_GetChannelCount) > GetChannelCount
- helper::DynFuncHolder< decltype(bigvoice_GetAudioInputCount) > GetAudioInputCount
- helper::DynFuncHolder< decltype(bigvoice_GetAudioOutputCount) > GetAudioOutputCount
- helper::DynFuncHolder< decltype(bigvoice_GetSampleRate) > GetSampleRate
- helper::DynFuncHolder< decltype(bigvoice_SetLoggerCallback) > SetLoggerCallback
- helper::DynFuncHolder< decltype(bigvoice_SetLogSeverity) > SetLogSeverity
- helper::DynFuncHolder< decltype(bigvoice_NewParameters) > NewParameters
- helper::DynFuncHolder< decltype(bigvoice_FreeParameters) > FreeParameters
- helper::DynFuncHolder< decltype(bigvoice_SetParameter) > SetParameter
- helper::DynFuncHolder< decltype(bigvoice_GetParameter) > GetParameter
- helper::DynFuncHolder< decltype(bigvoice_FreeParameterValue) > FreeParameterValue
- helper::DynFuncHolder< decltype(bigvoice_InitProcess) > InitProcess
- helper::DynFuncHolder< decltype(bigvoice_InitProcess2) > InitProcess2
- helper::DynFuncHolder< decltype(bigvoice_TerminateProcess) > TerminateProcess
- helper::DynFuncHolder< decltype(bigvoice_ExitProcess) > ExitProcess
- helper::DynFuncHolder< decltype(bigvoice_StartUpdateThread) > StartUpdateThread
- helper::DynFuncHolder< decltype(bigvoice_StopUpdateThread) > StopUpdateThread
- helper::DynFuncHolder< decltype(bigvoice_WaitUpdateThreadReady) > WaitUpdateThreadReady
- helper::DynFuncHolder< decltype(bigvoice_ProcessAudio) > ProcessAudio
- helper::DynFuncHolder< decltype(bigvoice_ProcessAudio_Planar) > ProcessAudio_Planar
- helper::DynFuncHolder< decltype(bigvoice_GetBufferIn) > GetBufferIn
- helper::DynFuncHolder< decltype(bigvoice_GetBufferOut) > GetBufferOut
- helper::DynFuncHolder< decltype(bigvoice_GetEstimatedDelay) > GetEstimatedDelay

- helper::DynFuncHolder< decltype(bigvoice_GetFormatName) > GetFormatName
- helper::DynFuncHolder< decltype(bigvoice_GetFormatFromName) > GetFormatFromName
- helper::DynFuncHolder< decltype(bigvoice_GetBytesFromFormat) > GetBytesFromFormat
- helper::DynFuncHolder< decltype(bigvoice_GetMaxPacketFrame) > GetMaxPacketFrame
- helper::DynFuncHolder< decltype(bigvoice_AddAudio) > AddAudio
- helper::DynFuncHolder< decltype(bigvoice_AddPadAudio) > AddPadAudio
- helper::DynFuncHolder< decltype(bigvoice_GetOutputCount) > GetOutputCount
- helper::DynFuncHolder< decltype(bigvoice_GetAudio) > GetAudio
- helper::DynFuncHolder< decltype(bigvoice_AudioConvertFrom) > AudioConvertFrom
- helper::DynFuncHolder< decltype(bigvoice_AudioConvertTo) > AudioConvertTo
- helper::DynFuncHolder< decltype(bigvoice_StereoToMono) > StereoToMono
- helper::DynFuncHolder< decltype(bigvoice_MonoToStereo) > MonoToStereo
- helper::DynFuncHolder< decltype(bigvoice_AudioMonoFromLiveStereo) > AudioMonoFromLiveStereo
- helper::DynFuncHolder< decltype(bigvoice_AudioMonoToLiveStereo) > AudioMonoToLiveStereo
- helper::DynFuncHolder< decltype(bigvoice_NewClient) > NewClient
- helper::DynFuncHolder< decltype(bigvoice_DeleteClient) > DeleteClient
- helper::DynFuncHolder< decltype(bigvoice_ProcessJson) > ProcessJson
- helper::DynFuncHolder< decltype(bigvoice_FreeJsonAnswer) > FreeJsonAnswer
- helper::DynFuncHolder< decltype(bigvoice_SaveState) > SaveState
- helper::DynFuncHolder< decltype(bigvoice_Webserver_tcp) > Webserver_tcp
- helper::DynFuncHolder< decltype(bigvoice_Webserver_tcp2) > Webserver_tcp2
- helper::DynFuncHolder< decltype(bigvoice_Webserver) > Webserver
- helper::DynFuncHolder< decltype(bigvoice_Webserver_Stop) > Webserver_Stop
- helper::DynFuncHolder< decltype(bigvoice_Webserver_Status) > Webserver_Status
- helper::DynFuncHolder< decltype(bigvoice_StereoToMono_Planar) > StereoToMono_Planar
- helper::DynFuncHolder< decltype(bigvoice_MonoToStereo_Planar) > MonoToStereo_Planar
- helper::DynFuncHolder< decltype(bigvoice_SetMetadata) > SetMetadata
- helper::DynFuncHolder< decltype(bigvoice_GetMetadataInfos) > GetMetadataInfos
- helper::DynFuncHolder< decltype(bigvoice_FreeMetadataInfos) > FreeMetadataInfos
- helper::DynFuncHolder< decltype(bigvoice_SetPresetManager) > SetPresetManager
- helper::DynFuncHolder< decltype(bigvoice_PresetManager_InformChange) > PresetManager_InformChange
- helper::DynFuncHolder< decltype(bigvoice_GetPossibleChunkSizeInFrames) > GetPossibleChunkSizeInFrames
- helper::DynFuncHolder< decltype(bigvoice_GetProcessChunkFrames) > GetProcessChunkFrames
- helper::DynFuncHolder< decltype(bigvoice_InitProcess3) > InitProcess3
- helper::DynFuncHolder< decltype(bigvoice_NewBus) > NewBus
- helper::DynFuncHolder< decltype(bigvoice_FreeBus) > FreeBus
- helper::DynFuncHolder< decltype(bigvoice_SetInstanceBus) > SetInstanceBus
- helper::DynFuncHolder< decltype(bigvoice_SetMetadataMulti) > SetMetadataMulti
- helper::DynFuncHolder< decltype(bigvoice_Webserver_SetAppHealth) > Webserver_SetAppHealth
- helper::DynFuncHolder< decltype(bigvoice_Webserver_GetAppHealth) > Webserver_GetAppHealth
- helper::DynFuncHolder< decltype(bigvoice_Webserver_FreeString) > Webserver_FreeString

10.11.1 Detailed Description

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

Definition at line 216 of file sound4.bigvoice.cl_dyn.hpp.

10.11.2 Constructor & Destructor Documentation

10.11.2.1 CDynLib()

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

10.11.3 Member Function Documentation

10.11.3.1 IsOk()

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

Returns

true if OK.

false if not OK.

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

10.11.3.2 Load()

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

Parameters

path	the path of the library to load.

Returns

true if library was loaded correctly.

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

Definition at line 318 of file sound4.bigvoice.cl_dyn.hpp.

10.11.4 Member Data Documentation

10.11.4.1 AddAudio

```
helper::DynFuncHolder< decltype(bigvoice_AddAudio ) > sound4::bigvoice::dyn::CDynLib::AddAudio
```

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

10.11.4.2 AddPadAudio

```
helper::DynFuncHolder< decltype(bigvoice_AddPadAudio ) > sound4::bigvoice::dyn::CDynLib::Add↩
PadAudio
```

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

10.11.4.3 AudioConvertFrom

helper::DynFuncHolder< decltype(bigvoice_AudioConvertFrom) > sound4::bigvoice::dyn::CDynLib↩
::AudioConvertFrom

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

10.11.4.4 AudioConvertTo

helper::DynFuncHolder< decltype(bigvoice_AudioConvertTo) > sound4::bigvoice::dyn::CDynLib::↩
AudioConvertTo

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

10.11.4.5 AudioMonoFromLiveStereo

helper::DynFuncHolder< decltype(bigvoice_AudioMonoFromLiveStereo) > sound4::bigvoice::dyn::↩
CDynLib::AudioMonoFromLiveStereo

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

10.11.4.6 AudioMonoToLiveStereo

helper::DynFuncHolder< decltype(bigvoice_AudioMonoToLiveStereo) > sound4::bigvoice::dyn::↩
CDynLib::AudioMonoToLiveStereo

Definition at line 262 of file sound4.bigvoice.cl_dyn.hpp.

10.11.4.7 DeleteClient

helper::DynFuncHolder< decltype(bigvoice_DeleteClient) > sound4::bigvoice::dyn::CDynLib::↩
DeleteClient

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

10.11.4.8 ExitProcess

helper::DynFuncHolder< decltype(bigvoice_ExitProcess) > sound4::bigvoice::dyn::CDynLib::↩
ExitProcess

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

10.11.4.9 FreeBus

helper::DynFuncHolder< decltype(bigvoice_FreeBus) > sound4::bigvoice::dyn::CDynLib::FreeBus

Definition at line 297 of file sound4.bigvoice.cl_dyn.hpp.

10.11.4.10 FreeJsonAnswer

```
helper::DynFuncHolder< decltype(bigvoice_FreeJsonAnswer ) > sound4::bigvoice::dyn::CDynLib::↩
FreeJsonAnswer
```

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

10.11.4.11 FreeMetadataInfos

```
helper::DynFuncHolder< decltype(bigvoice_FreeMetadataInfos ) > sound4::bigvoice::dyn::CDyn↩
Lib::FreeMetadataInfos
```

Definition at line 283 of file sound4.bigvoice.cl_dyn.hpp.

10.11.4.12 FreeParameters

```
helper::DynFuncHolder< decltype(bigvoice_FreeParameters ) > sound4::bigvoice::dyn::CDynLib::↩
FreeParameters
```

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

10.11.4.13 FreeParameterValue

```
helper::DynFuncHolder< decltype(bigvoice_FreeParameterValue ) > sound4::bigvoice::dyn::CDyn↩
Lib::FreeParameterValue
```

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

10.11.4.14 GetAudio

```
helper::DynFuncHolder< decltype(bigvoice_GetAudio ) > sound4::bigvoice::dyn::CDynLib::GetAudio
```

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

10.11.4.15 GetAudioInputCount

```
helper::DynFuncHolder< decltype(bigvoice_GetAudioInputCount ) > sound4::bigvoice::dyn::CDyn↩
Lib::GetAudioInputCount
```

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

10.11.4.16 GetAudioOutputCount

```
helper::DynFuncHolder< decltype(bigvoice_GetAudioOutputCount ) > sound4::bigvoice::dyn::CDyn↩
Lib::GetAudioOutputCount
```

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

10.11.4.17 GetBufferIn

```
helper::DynFuncHolder< decltype(bigvoice_GetBufferIn ) > sound4::bigvoice::dyn::CDynLib::Get↩
BufferIn
```

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

10.11.4.18 GetBufferOut

```
helper::DynFuncHolder< decltype(bigvoice_GetBufferOut ) > sound4::bigvoice::dyn::CDynLib::↩
GetBufferOut
```

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

10.11.4.19 GetBytesFromFormat

```
helper::DynFuncHolder< decltype(bigvoice_GetBytesFromFormat ) > sound4::bigvoice::dyn::CDyn↩
Lib::GetBytesFromFormat
```

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

10.11.4.20 GetChannelCount

```
helper::DynFuncHolder< decltype(bigvoice_GetChannelCount ) > sound4::bigvoice::dyn::CDynLib↩
::GetChannelCount
```

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

10.11.4.21 GetChunkSizeInFrames

```
helper::DynFuncHolder< decltype(bigvoice_GetChunkSizeInFrames ) > sound4::bigvoice::dyn::↩
CDynLib::GetChunkSizeInFrames
```

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

10.11.4.22 GetEstimatedDelay

```
helper::DynFuncHolder< decltype(bigvoice_GetEstimatedDelay ) > sound4::bigvoice::dyn::CDyn↩
Lib::GetEstimatedDelay
```

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

10.11.4.23 GetFormatFromName

```
helper::DynFuncHolder< decltype(bigvoice_GetFormatFromName ) > sound4::bigvoice::dyn::CDyn↩
Lib::GetFormatFromName
```

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

10.11.4.24 GetFormatName

helper::DynFuncHolder< decltype(bigvoice_GetFormatName) > sound4::bigvoice::dyn::CDynLib::↩
GetFormatName

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

10.11.4.25 GetMaxPacketFrame

helper::DynFuncHolder< decltype(bigvoice_GetMaxPacketFrame) > sound4::bigvoice::dyn::CDyn↩
Lib::GetMaxPacketFrame

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

10.11.4.26 GetMetadataInfos

helper::DynFuncHolder< decltype(bigvoice_GetMetadataInfos) > sound4::bigvoice::dyn::CDynLib↩
::GetMetadataInfos

Definition at line 282 of file sound4.bigvoice.cl_dyn.hpp.

10.11.4.27 GetOutputCount

helper::DynFuncHolder< decltype(bigvoice_GetOutputCount) > sound4::bigvoice::dyn::CDynLib::↩
GetOutputCount

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

10.11.4.28 GetParameter

helper::DynFuncHolder< decltype(bigvoice_GetParameter) > sound4::bigvoice::dyn::CDynLib::↩
GetParameter

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

10.11.4.29 GetPossibleChunkSizeInFrames

helper::DynFuncHolder< decltype(bigvoice_GetPossibleChunkSizeInFrames) > sound4::bigvoice↩
::dyn::CDynLib::GetPossibleChunkSizeInFrames

Definition at line 290 of file sound4.bigvoice.cl_dyn.hpp.

10.11.4.30 GetProcessChunkFrames

helper::DynFuncHolder< decltype(bigvoice_GetProcessChunkFrames) > sound4::bigvoice::dyn::↩
CDynLib::GetProcessChunkFrames

Definition at line 291 of file sound4.bigvoice.cl_dyn.hpp.

10.11.4.31 GetSampleRate

helper::DynFuncHolder< decltype(bigvoice_GetSampleRate) > sound4::bigvoice::dyn::CDynLib::↩
GetSampleRate

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

10.11.4.32 GetVersion

helper::DynFuncHolder< decltype(bigvoice_GetVersion) > sound4::bigvoice::dyn::CDynLib::Get↩
Version

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

10.11.4.33 InitProcess

helper::DynFuncHolder< decltype(bigvoice_InitProcess) > sound4::bigvoice::dyn::CDynLib::↩
InitProcess

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

10.11.4.34 InitProcess2

helper::DynFuncHolder< decltype(bigvoice_InitProcess2) > sound4::bigvoice::dyn::CDynLib::↩
InitProcess2

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

10.11.4.35 InitProcess3

helper::DynFuncHolder< decltype(bigvoice_InitProcess3) > sound4::bigvoice::dyn::CDynLib::↩
InitProcess3

Definition at line 292 of file sound4.bigvoice.cl_dyn.hpp.

10.11.4.36 MonoToStereo

helper::DynFuncHolder< decltype(bigvoice_MonoToStereo) > sound4::bigvoice::dyn::CDynLib::↩
MonoToStereo

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

10.11.4.37 MonoToStereo_Planar

helper::DynFuncHolder< decltype(bigvoice_MonoToStereo_Planar) > sound4::bigvoice::dyn::CDyn↩
Lib::MonoToStereo_Planar

Definition at line 278 of file sound4.bigvoice.cl_dyn.hpp.

10.11.4.38 NewBus

```
helper::DynFuncHolder< decltype(bigvoice_NewBus ) > sound4::bigvoice::dyn::CDynLib::NewBus
```

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

10.11.4.39 NewClient

```
helper::DynFuncHolder< decltype(bigvoice_NewClient ) > sound4::bigvoice::dyn::CDynLib::New↩
Client
```

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

10.11.4.40 NewParameters

```
helper::DynFuncHolder< decltype(bigvoice_NewParameters ) > sound4::bigvoice::dyn::CDynLib::↩
NewParameters
```

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

10.11.4.41 PresetManager_InformChange

```
helper::DynFuncHolder< decltype(bigvoice_PresetManager_InformChange ) > sound4::bigvoice↩
::dyn::CDynLib::PresetManager_InformChange
```

Definition at line 287 of file sound4.bigvoice.cl_dyn.hpp.

10.11.4.42 ProcessAudio

```
helper::DynFuncHolder< decltype(bigvoice_ProcessAudio ) > sound4::bigvoice::dyn::CDynLib::↩
ProcessAudio
```

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

10.11.4.43 ProcessAudio_Planar

```
helper::DynFuncHolder< decltype(bigvoice_ProcessAudio_Planar ) > sound4::bigvoice::dyn::CDyn↩
Lib::ProcessAudio_Planar
```

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

10.11.4.44 ProcessJson

```
helper::DynFuncHolder< decltype(bigvoice_ProcessJson ) > sound4::bigvoice::dyn::CDynLib::↩
ProcessJson
```

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

10.11.4.45 SaveState

helper::DynFuncHolder< decltype(bigvoice_SaveState) > sound4::bigvoice::dyn::CDynLib::SaveState

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

10.11.4.46 SetInstanceBus

helper::DynFuncHolder< decltype(bigvoice_SetInstanceBus) > sound4::bigvoice::dyn::CDynLib::SetInstanceBus

Definition at line 298 of file sound4.bigvoice.cl_dyn.hpp.

10.11.4.47 SetLoggerCallback

helper::DynFuncHolder< decltype(bigvoice_SetLoggerCallback) > sound4::bigvoice::dyn::CDynLib::SetLoggerCallback

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

10.11.4.48 SetLogSeverity

helper::DynFuncHolder< decltype(bigvoice_SetLogSeverity) > sound4::bigvoice::dyn::CDynLib::SetLogSeverity

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

10.11.4.49 SetMetadata

helper::DynFuncHolder< decltype(bigvoice_SetMetadata) > sound4::bigvoice::dyn::CDynLib::SetMetadata

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

10.11.4.50 SetMetadataMulti

helper::DynFuncHolder< decltype(bigvoice_SetMetadataMulti) > sound4::bigvoice::dyn::CDynLib::SetMetadataMulti

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

10.11.4.51 SetParameter

helper::DynFuncHolder< decltype(bigvoice_SetParameter) > sound4::bigvoice::dyn::CDynLib::SetParameter

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

10.11.4.52 SetPresetManager

```
helper::DynFuncHolder< decltype(bigvoice_SetPresetManager ) > sound4::bigvoice::dyn::CDynLib↩
::SetPresetManager
```

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

10.11.4.53 StartUpdateThread

```
helper::DynFuncHolder< decltype(bigvoice_StartUpdateThread ) > sound4::bigvoice::dyn::CDyn↩
Lib::StartUpdateThread
```

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

10.11.4.54 StereoToMono

```
helper::DynFuncHolder< decltype(bigvoice_StereoToMono ) > sound4::bigvoice::dyn::CDynLib::↩
StereoToMono
```

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

10.11.4.55 StereoToMono_Planar

```
helper::DynFuncHolder< decltype(bigvoice_StereoToMono_Planar ) > sound4::bigvoice::dyn::CDyn↩
Lib::StereoToMono_Planar
```

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

10.11.4.56 StopUpdateThread

```
helper::DynFuncHolder< decltype(bigvoice_StopUpdateThread ) > sound4::bigvoice::dyn::CDynLib↩
::StopUpdateThread
```

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

10.11.4.57 TerminateProcess

```
helper::DynFuncHolder< decltype(bigvoice_TerminateProcess ) > sound4::bigvoice::dyn::CDynLib↩
::TerminateProcess
```

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

10.11.4.58 WaitUpdateThreadReady

```
helper::DynFuncHolder< decltype(bigvoice_WaitUpdateThreadReady ) > sound4::bigvoice::dyn::↩
CDynLib::WaitUpdateThreadReady
```

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

10.11.4.59 Webserver

helper::DynFuncHolder< decltype(bigvoice_Webserver) > sound4::bigvoice::dyn::CDynLib::↩
Webserver

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

10.11.4.60 Webserver_FreeString

helper::DynFuncHolder< decltype(bigvoice_Webserver_FreeString) > sound4::bigvoice::dyn::↩
CDynLib::Webserver_FreeString

Definition at line 308 of file sound4.bigvoice.cl_dyn.hpp.

10.11.4.61 Webserver_GetAppHealth

helper::DynFuncHolder< decltype(bigvoice_Webserver_GetAppHealth) > sound4::bigvoice::dyn::↩
CDynLib::Webserver_GetAppHealth

Definition at line 307 of file sound4.bigvoice.cl_dyn.hpp.

10.11.4.62 Webserver_SetAppHealth

helper::DynFuncHolder< decltype(bigvoice_Webserver_SetAppHealth) > sound4::bigvoice::dyn::↩
CDynLib::Webserver_SetAppHealth

Definition at line 306 of file sound4.bigvoice.cl_dyn.hpp.

10.11.4.63 Webserver_Status

helper::DynFuncHolder< decltype(bigvoice_Webserver_Status) > sound4::bigvoice::dyn::CDynLib↩
::Webserver_Status

Definition at line 273 of file sound4.bigvoice.cl_dyn.hpp.

10.11.4.64 Webserver_Stop

helper::DynFuncHolder< decltype(bigvoice_Webserver_Stop) > sound4::bigvoice::dyn::CDynLib::↩
Webserver_Stop

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

10.11.4.65 Webserver_tcp

helper::DynFuncHolder< decltype(bigvoice_Webserver_tcp) > sound4::bigvoice::dyn::CDynLib::↩
Webserver_tcp

Definition at line 269 of file sound4.bigvoice.cl_dyn.hpp.

10.11.4.66 Webserver_tcp2

```
helper::DynFuncHolder< decltype(bigvoice_Webserver_tcp2 ) > sound4::bigvoice::dyn::CDynLib::↩
Webserver_tcp2
```

Definition at line 270 of file sound4.bigvoice.cl_dyn.hpp.

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

- sound4.bigvoice.cl_dyn.hpp

10.12 sound4::bigvoice::dyn::helper::CDynLoader Class Reference

Helper to load dynamic library.

```
#include <sound4.bigvoice.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_LIBRAR
- 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)

10.12.1 Detailed Description

Definition at line 107 of file sound4.bigvoice.cl_dyn.hpp.

10.12.2 Constructor & Destructor Documentation

10.12.2.1 CDynLoader() [1/3]

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

10.12.2.2 CDynLoader() [2/3]

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

10.12.2.3 CDynLoader() [3/3]

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

10.12.2.4 ~CDynLoader()

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

Definition at line 204 of file sound4.bigvoice.cl_dyn.hpp.

10.12.3 Member Function Documentation

10.12.3.1 Close()

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

Definition at line 165 of file sound4.bigvoice.cl_dyn.hpp.

10.12.3.2 GetSymbol()

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

Definition at line 178 of file sound4.bigvoice.cl_dyn.hpp.

10.12.3.3 GetSymbol_safe()

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

Definition at line 186 of file sound4.bigvoice.cl_dyn.hpp.

10.12.3.4 GetThisLibraryPath()

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

Definition at line 190 of file sound4.bigvoice.cl_dyn.hpp.

10.12.3.5 IsOk()

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

Definition at line 119 of file sound4.bigvoice.cl_dyn.hpp.

10.12.3.6 Load()

```
bool sound4::bigvoice::dyn::helper::CDynLoader::Load (
            const std::filesystem::path & dynlib,
            uint32_t loadflags = LOAD_LIBRARY_SEARCH_DLL_LOAD_DIR|LOAD_LIBRARY_SEARCH_APPLICATION_DIR|LOAD_LIB
)  [inline]
```

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

10.12.3.7 operator bool()

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

Definition at line 116 of file sound4.bigvoice.cl_dyn.hpp.

10.12.3.8 operator=()

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

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

- sound4.bigvoice.cl_dyn.hpp

10.13 sound4::helper::CDynLoader Class Reference

Helper to load dynamic library.

`#include <sound4cl.hpp>`

Public Member Functions

- CDynLoader ()=default
- CDynLoader (CDynLoader &&)=default
- CDynLoader (const CDynLoader &)=delete
- CDynLoader & operator= (CDynLoader const &)=delete
- operator bool () const
- bool IsOk () const
- bool Load (const std::filesystem::path &dynlib, uint32_t loadflags=LOAD_LIBRARY_SEARCH_DLL_LOAD_DIR|LOAD_LIBRAF
- 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)

10.13.1 Detailed Description

Definition at line 136 of file sound4cl.hpp.

10.13.2 Constructor & Destructor Documentation

10.13.2.1 CDynLoader() [1/3]

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

10.13.2.2 CDynLoader() [2/3]

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

10.13.2.3 CDynLoader() [3/3]

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

10.13.2.4 ~CDynLoader()

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

Definition at line 233 of file sound4cl.hpp.

10.13.3 Member Function Documentation

10.13.3.1 Close()

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

Definition at line 194 of file sound4cl.hpp.

10.13.3.2 GetSymbol()

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

Definition at line 207 of file sound4cl.hpp.

10.13.3.3 GetSymbol_safe()

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

Definition at line 215 of file sound4cl.hpp.

10.13.3.4 GetThisLibraryPath()

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

Definition at line 219 of file sound4cl.hpp.

10.13.3.5 IsOk()

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

Definition at line 148 of file sound4cl.hpp.

10.13.3.6 Load()

```
bool sound4::helper::CDynLoader::Load (
            const std::filesystem::path & dynlib,
            uint32_t loadflags = LOAD_LIBRARY_SEARCH_DLL_LOAD_DIR|LOAD_LIBRARY_SEARCH_APPLICATION_DIR|LOAD_LIB
)  [inline]
```

Definition at line 188 of file sound4cl.hpp.

10.13.3.7 operator bool()

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

Definition at line 145 of file sound4cl.hpp.

10.13.3.8 operator=()

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

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

- sound4cl.hpp

10.14 sound4::bigvoice::CInstance Class Reference

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

Classes

- class CClient

Public Member Functions

- CInstance ()
- CInstance (const CInstance &)=delete
- CInstance & operator= (const CInstance &)=delete
- virtual ~CInstance ()
- bool IsOk () const
- void SetParam (const std::string &name, const std::string &value)
- std::string GetParam (const std::string &name)
- void SetPresetManager (CPresetLoader *preset_manager)
- void SetBus (const CBus &bus)
- 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=12)
- unsigned int GetChunkFrames ()
- void PresetManager_InformChange (const std::filesystem::path &relative_path, PresetChange_Kind change_kind)
- void SetMetadata (const std::string &key, const char *value)
- void SetMetadataMulti (const std::unordered_map< std::string, const char * > &list)
- std::vector< std::tuple< std::string, std::string > > GetMetadataInfos ()
- virtual void OnUpdateThreadStart ()
- virtual void OnUpdateThreadStop ()
- bool StartWebServer (int http_port, int https_port=0)
- void StopWebServer ()
- void SetWebServerAppHealth (int httpcode, const std::string &contenttype, const std::string &content)
- void GetWebServerAppHealth (int &httpcode, std::string &contenttype, std::string &content)
- int Terminate ()
- void Stop ()
- unsigned int GetEstimatedDelay ()
- std::array< float, InputSampleSize > & GetBufferIn ()
- std::array< float, OutputSampleSize > & GetBufferOut ()
- void ProcessAudio ()
- void ProcessAudio_Planar (float const *const *input, float *const *output)
- template<typename T >
  std::vector< T > ProcessAnyAudio (const std::vector< T > input)
- std::shared_ptr< CClient > NewClient ()

Protected Member Functions

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

10.14.1 Detailed Description

Instance handling class

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

10.14.2 Constructor & Destructor Documentation

10.14.2.1 CInstance() [1/2]

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

Creates the instance, with default parameters

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

10.14.2.2 CInstance() [2/2]

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

10.14.2.3 ~CInstance()

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

Destructor, stopping any running tasks

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

10.14.3 Member Function Documentation

10.14.3.1 AddAudio()

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

Definition at line 883 of file sound4.bigvoice.cl.hpp.

10.14.3.2 Create()

```
bool sound4::bigvoice::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 = 12 )  [inline]
```

Creates and initialize the instance, and starts its update thread

See also

bigvoice_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 bigvoice_GetPossibleChunkSizeInFrames, or 0 to set default

Returns

false if failed

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

10.14.3.3 GetAudio()

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

Definition at line 889 of file sound4.bigvoice.cl.hpp.

10.14.3.4 GetBufferIn()

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

Gets the input buffer used in ProcessAudio

Input has GetAudioInputCount() * GetChannelCount() channels

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

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

Definition at line 738 of file sound4.bigvoice.cl.hpp.

10.14.3.5 GetBufferOut()

```
std::array< float, OutputSampleSize > & sound4::bigvoice::CInstance::GetBufferOut ( )  [inline]
```

Gets the output buffer used in ProcessAudio

Output has GetAudioOutputCount() * GetChannelCount() channels

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

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

Definition at line 755 of file sound4.bigvoice.cl.hpp.

10.14.3.6 GetChunkFrames()

```
unsigned int sound4::bigvoice::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.bigvoice.cl.hpp.

10.14.3.7 GetEstimatedDelay()

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

Gets an estimated delay of current processing

See also

bigvoice_GetEstimatedDelay

Definition at line 722 of file sound4.bigvoice.cl.hpp.

10.14.3.8 GetMetadataInfos()

```
std::vector< std::tuple< std::string, std::string > > sound4::bigvoice::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 602 of file sound4.bigvoice.cl.hpp.

10.14.3.9 GetParam()

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

Gets a parameter value

See also

bigvoice_GetParameter

Definition at line 427 of file sound4.bigvoice.cl.hpp.

10.14.3.10 GetWebServerAppHealth()

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

Get the web server /apphealth status info

Definition at line 673 of file sound4.bigvoice.cl.hpp.

10.14.3.11 IsOk()

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

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

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

10.14.3.12 NewClient()

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

Creates a low-level JSON client

See also

JSON syntax

Returns

a new client to communicate with the processing

Definition at line 875 of file sound4.bigvoice.cl.hpp.

10.14.3.13 OnUpdateThreadStart()

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

Definition at line 622 of file sound4.bigvoice.cl.hpp.

10.14.3.14 OnUpdateThreadStop()

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

Definition at line 624 of file sound4.bigvoice.cl.hpp.

10.14.3.15 operator=()

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

10.14.3.16 PresetManager_InformChange()

```
void sound4::bigvoice::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.bigvoice.cl.hpp.

10.14.3.17 ProcessAnyAudio()

```
template<typename T >
std::vector< T > sound4::bigvoice::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 BIGVOICE_AUDIOFRAME_COUNT

Template Parameters

T	the C type for one audio sample

Parameters

input	audio to process

Returns

processed audio

Definition at line 803 of file sound4.bigvoice.cl.hpp.

10.14.3.18 ProcessAudio()

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

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

See also

bigvoice_ProcessAudio

Definition at line 766 of file sound4.bigvoice.cl.hpp.

10.14.3.19 ProcessAudio_Planar()

```
void sound4::bigvoice::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

bigvoice_ProcessAudio_Planar

Parameters

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

Definition at line 785 of file sound4.bigvoice.cl.hpp.

10.14.3.20 SetBus()

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

Set the instance bus

Definition at line 467 of file sound4.bigvoice.cl.hpp.

10.14.3.21 SetMetadata()

```
void sound4::bigvoice::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.bigvoice.cl.hpp.

10.14.3.22 SetMetadataMulti()

```
void sound4::bigvoice::CInstance::SetMetadataMulti (
            const std::unordered_map< std::string, const char * > & list )  [inline]
```

Adds multiple metadata to the process.

This works like CInstance::SetMetadata but with multiple metadata at once.

See also

CInstance::SetMetadata

Parameters

list	a map of key/value pair.

Definition at line 572 of file sound4.bigvoice.cl.hpp.

10.14.3.23 SetParam()

```
void sound4::bigvoice::CInstance::SetParam (
            const std::string & name,
            const std::string & value )  [inline]
```

Set a parameter value

Must be done before Create

See also

bigvoice_SetParameter

Definition at line 417 of file sound4.bigvoice.cl.hpp.

10.14.3.24 SetPresetManager()

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

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

See also

bigvoice_SetPresetManager

Must be done before Create

Parameters

preset_manager	The custom preset manager

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

10.14.3.25 SetWebServerAppHealth()

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

Set the web server /apphealth status info

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

10.14.3.26 StartWebServer()

```
bool sound4::bigvoice::CInstance::StartWebServer (
            int http_port,
            int https_port = 0 )  [inline]
```

Starts a Web server on requested ports

See also

bigvoice_Webserver

Definition at line 632 of file sound4.bigvoice.cl.hpp.

10.14.3.27 Stop()

```
void sound4::bigvoice::CInstance::Stop ( )  [inline]
```

Stops the instance and its web server and its update thread.

See also

bigvoice_ExitProcess,
bigvoice_StopUpdateThread and
bigvoice_Webserver_Stop

Definition at line 703 of file sound4.bigvoice.cl.hpp.

10.14.3.28 StopWebServer()

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

Stops the Web server

See also

bigvoice_Webserver_Stop

Definition at line 654 of file sound4.bigvoice.cl.hpp.

10.14.3.29 Terminate()

```
int sound4::bigvoice::CInstance::Terminate ( )  [inline]
```

Terminate the processing. This is useful only with file processing, to get the final status.

See also

bigvoice_TerminateProcess

Definition at line 692 of file sound4.bigvoice.cl.hpp.

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

- sound4.bigvoice.cl.hpp

10.15 sound4::bigvoice::dyn::CInstance Class Reference

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

Classes

- class CClient

Public Member Functions

- CInstance (CDynLib &dynlib)
- CInstance (const CInstance &)=delete
- CInstance & operator= (const CInstance &)=delete
- virtual ~CInstance ()
- bool IsOk () const
- void SetParam (const std::string &name, const std::string &value)
- std::string GetParam (const std::string &name)
- void SetBus (const CBus &bus)
- void SetPresetManager (CPresetLoader *preset_manager)
- bool Create (const std::string &LoginKey, const std::string &RadioName, const std::string &Access_Key_ID, const std::string &Access_Key_Secret, const std::filesystem::path &save_path, int json_port=0, unsigned int frames_per_chunk=12)
- unsigned int GetChunkFrames ()
- void PresetManager_InformChange (const std::filesystem::path &relative_path, PresetChange_Kind change_kind)
- void SetMetadata (const std::string &key, const char *value)
- void SetMetadataMulti (const std::unordered_map< std::string, const char * > &list)
- std::vector< std::tuple< std::string, std::string > > GetMetadataInfos ()
- virtual void OnUpdateThreadStart ()
- virtual void OnUpdateThreadStop ()
- bool StartWebServer (int http_port, int https_port=0)
- void StopWebServer ()
- void SetWebServerAppHealth (int httpcode, const std::string &contenttype, const std::string &content)
- void GetWebServerAppHealth (int &httpcode, std::string &contenttype, std::string &content)
- int Terminate ()
- void Stop ()
- unsigned int GetEstimatedDelay ()
- std::array< float, InputSampleSize > & GetBufferIn ()
- std::array< float, OutputSampleSize > & GetBufferOut ()
- void ProcessAudio ()
- void ProcessAudio_Planar (float const *const *input, float *const *output)
- template<typename T >
  std::vector< T > ProcessAnyAudio (const std::vector< T > input)
- std::shared_ptr< CClient > NewClient ()

Protected Member Functions

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

10.15.1 Detailed Description

Instance handling class

Definition at line 816 of file sound4.bigvoice.cl_dyn.hpp.

10.15.2 Constructor & Destructor Documentation

10.15.2.1 CInstance() [1/2]

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

Creates the instance, with default parameters

Definition at line 821 of file sound4.bigvoice.cl_dyn.hpp.

10.15.2.2 CInstance() [2/2]

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

10.15.2.3 ~CInstance()

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

Destructor, stopping any running tasks

Definition at line 833 of file sound4.bigvoice.cl_dyn.hpp.

10.15.3 Member Function Documentation

10.15.3.1 AddAudio()

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

Definition at line 1340 of file sound4.bigvoice.cl_dyn.hpp.

10.15.3.2 Create()

```
bool sound4::bigvoice::dyn::CInstance::Create (
            const std::string & LoginKey,
            const std::string & RadioName,
            const std::string & Access_Key_ID,
            const std::string & Access_Key_Secret,
            const std::filesystem::path & save_path,
            int json_port = 0,
            unsigned int frames_per_chunk = 12 )  [inline]
```

Creates and initialize the instance, and starts its update thread

See also

bigvoice_InitProcess2

Parameters

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

Returns

false if failed

Definition at line 921 of file sound4.bigvoice.cl_dyn.hpp.

10.15.3.3 GetAudio()

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

Definition at line 1346 of file sound4.bigvoice.cl_dyn.hpp.

10.15.3.4 GetBufferIn()

```
std::array< float, InputSampleSize > & sound4::bigvoice::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 1193 of file sound4.bigvoice.cl_dyn.hpp.

10.15.3.5 GetBufferOut()

```
std::array< float, OutputSampleSize > & sound4::bigvoice::dyn::CInstance::GetBufferOut ( )
[inline]
```

Gets the output buffer used in ProcessAudio

Output has GetAudioOutputCount() * GetChannelCount() channels

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

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

Definition at line 1210 of file sound4.bigvoice.cl_dyn.hpp.

10.15.3.6 GetChunkFrames()

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

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

The value reflects the frames_per_chunk parameter passed to Create

Returns

the number of audio frames in each chunk of processing

Definition at line 964 of file sound4.bigvoice.cl_dyn.hpp.

10.15.3.7 GetEstimatedDelay()

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

Gets an estimated delay of current processing

See also

bigvoice_GetEstimatedDelay

Definition at line 1177 of file sound4.bigvoice.cl_dyn.hpp.

10.15.3.8 GetMetadataInfos()

```
std::vector< std::tuple< std::string, std::string > > sound4::bigvoice::dyn::CInstance::Get↩
MetadataInfos ( )  [inline]
```

Get information on known metadata keys.

Returns

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

Definition at line 1053 of file sound4.bigvoice.cl_dyn.hpp.

10.15.3.9 GetParam()

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

Gets a parameter value

See also

bigvoice_GetParameter

Definition at line 859 of file sound4.bigvoice.cl_dyn.hpp.

10.15.3.10 GetWebServerAppHealth()

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

Get the web server /apphealth status info

Definition at line 1127 of file sound4.bigvoice.cl_dyn.hpp.

10.15.3.11 IsOk()

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

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

Definition at line 841 of file sound4.bigvoice.cl_dyn.hpp.

10.15.3.12 NewClient()

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

Creates a low-level JSON client

See also

JSON syntax

Returns

a new client to communicate with the processing

Definition at line 1332 of file sound4.bigvoice.cl_dyn.hpp.

10.15.3.13 OnUpdateThreadStart()

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

Definition at line 1075 of file sound4.bigvoice.cl_dyn.hpp.

10.15.3.14 OnUpdateThreadStop()

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

Definition at line 1077 of file sound4.bigvoice.cl_dyn.hpp.

10.15.3.15 operator=()

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

10.15.3.16 PresetManager_InformChange()

```
void sound4::bigvoice::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 978 of file sound4.bigvoice.cl_dyn.hpp.

10.15.3.17 ProcessAnyAudio()

```
template<typename T >
std::vector< T > sound4::bigvoice::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 BIGVOICE_AUDIOFRAME_COUNT

Template Parameters

T	the C type for one audio sample

Parameters

input	audio to process

Returns

processed audio

Definition at line 1258 of file sound4.bigvoice.cl_dyn.hpp.

10.15.3.18 ProcessAudio()

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

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

See also

bigvoice_ProcessAudio

Definition at line 1221 of file sound4.bigvoice.cl_dyn.hpp.

10.15.3.19 ProcessAudio_Planar()

```
void sound4::bigvoice::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

bigvoice_ProcessAudio_Planar

Parameters

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

Definition at line 1240 of file sound4.bigvoice.cl_dyn.hpp.

10.15.3.20 SetBus()

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

Set the instance bus

Definition at line 871 of file sound4.bigvoice.cl_dyn.hpp.

10.15.3.21 SetMetadata()

```
void sound4::bigvoice::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 996 of file sound4.bigvoice.cl_dyn.hpp.

10.15.3.22 SetMetadataMulti()

```
void sound4::bigvoice::dyn::CInstance::SetMetadataMulti (
            const std::unordered_map< std::string, const char * > & list )  [inline]
```

Adds multiple metadata to the process.

This works like CInstance::SetMetadata but with multiple metadata at once.

See also

CInstance::SetMetadata

Parameters

list	a map of key/value pair.

Definition at line 1017 of file sound4.bigvoice.cl_dyn.hpp.

10.15.3.23 SetParam()

```
void sound4::bigvoice::dyn::CInstance::SetParam (
            const std::string & name,
            const std::string & value )  [inline]
```

Set a parameter value

Must be done before Create

See also

bigvoice_SetParameter

Definition at line 849 of file sound4.bigvoice.cl_dyn.hpp.

10.15.3.24 SetPresetManager()

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

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

See also

bigvoice_SetPresetManager

Must be done before Create

Parameters

preset_manager	The custom preset manager

Definition at line 888 of file sound4.bigvoice.cl_dyn.hpp.

10.15.3.25 SetWebServerAppHealth()

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

Set the web server /apphealth status info

Definition at line 1118 of file sound4.bigvoice.cl_dyn.hpp.

10.15.3.26 StartWebServer()

```
bool sound4::bigvoice::dyn::CInstance::StartWebServer (
            int http_port,
            int https_port = 0 )  [inline]
```

Starts a Web server on requested ports

See also

bigvoice_Webserver

Definition at line 1085 of file sound4.bigvoice.cl_dyn.hpp.

10.15.3.27 Stop()

```
void sound4::bigvoice::dyn::CInstance::Stop ( )  [inline]
```

Stops the instance and its web server and its update thread.

See also

bigvoice_ExitProcess,
bigvoice_StopUpdateThread and
bigvoice_Webserver_Stop

Definition at line 1158 of file sound4.bigvoice.cl_dyn.hpp.

10.15.3.28 StopWebServer()

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

Stops the Web server

See also

bigvoice_Webserver_Stop

Definition at line 1107 of file sound4.bigvoice.cl_dyn.hpp.

10.15.3.29 Terminate()

```
int sound4::bigvoice::dyn::CInstance::Terminate ( )  [inline]
```

Terminate the processing. This is useful only with file processing, to get the final status.

See also

bigvoice_TerminateProcess

Definition at line 1147 of file sound4.bigvoice.cl_dyn.hpp.

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

- sound4.bigvoice.cl_dyn.hpp

10.16 sound4::CInstance Class Reference

Instance handling class.

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

Classes

- class CClient

Public Member Functions

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

Protected Member Functions

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

10.16.1 Detailed Description

Holds an instance of processing

Definition at line 760 of file sound4cl.hpp.

10.16.2 Constructor & Destructor Documentation

10.16.2.1 CInstance() [1/2]

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

Creates the instance, with default parameters

Definition at line 765 of file sound4cl.hpp.

10.16.2.2 CInstance() [2/2]

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

10.16.2.3 ~CInstance()

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

Destructor, stopping any running tasks

Definition at line 777 of file sound4cl.hpp.

10.16.3 Member Function Documentation

10.16.3.1 AddAudio()

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

Definition at line 1283 of file sound4cl.hpp.

10.16.3.2 Create()

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

Creates and initialize the instance, and starts its update thread

See also

sound4cl_InitProcess2

Parameters

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

Returns

false if failed

Definition at line 864 of file sound4cl.hpp.

10.16.3.3 GetAudio()

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

Definition at line 1289 of file sound4cl.hpp.

10.16.3.4 GetBufferIn()

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

Gets the input buffer used in ProcessAudio

Input has GetAudioInputCount() * GetChannelCount() channels

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

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

Definition at line 1138 of file sound4cl.hpp.

10.16.3.5 GetBufferOut()

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

Gets the output buffer used in ProcessAudio

Output has GetAudioOutputCount() * GetChannelCount() channels

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

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

Definition at line 1154 of file sound4cl.hpp.

10.16.3.6 GetChunkFrames()

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

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

The value reflects the frames_per_chunk parameter passed to Create

Returns

the number of audio frames in each chunk of processing

Definition at line 910 of file sound4cl.hpp.

10.16.3.7 GetEstimatedDelay()

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

Gets an estimated delay of current processing

See also

sound4cl_GetEstimatedDelay

Definition at line 1122 of file sound4cl.hpp.

10.16.3.8 GetMetadataInfos()

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

Get information on known metadata keys.

Returns

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

Definition at line 999 of file sound4cl.hpp.

10.16.3.9 GetParam()

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

Gets a parameter value

See also

sound4cl_GetParameter

Definition at line 803 of file sound4cl.hpp.

10.16.3.10 GetWebServerAppHealth()

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

Get the web server /apphealth status info

Definition at line 1076 of file sound4cl.hpp.

10.16.3.11 IsOk()

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

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

Definition at line 785 of file sound4cl.hpp.

10.16.3.12 NewClient()

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

Creates a low-level JSON client

See also

JSON syntax

Returns

a new client to communicate with the processing

Definition at line 1275 of file sound4cl.hpp.

10.16.3.13 OnUpdateThreadStart()

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

Definition at line 1021 of file sound4cl.hpp.

10.16.3.14 OnUpdateThreadStop()

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

Definition at line 1023 of file sound4cl.hpp.

10.16.3.15 operator=()

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

10.16.3.16 PresetManager_InformChange()

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

Inform that a change happened in the storage. This must be used only in 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 924 of file sound4cl.hpp.

10.16.3.17 ProcessAnyAudio()

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

Process audio (different formats, less optimized)

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

Attention

the return count is independent from the input count

Template Parameters

T	the C type for one audio sample

Parameters

input	audio to process

Returns

processed audio

Definition at line 1201 of file sound4cl.hpp.

10.16.3.18 ProcessAudio()

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

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

See also

sound4cl_ProcessAudio

Definition at line 1164 of file sound4cl.hpp.

10.16.3.19 ProcessAudio_Planar()

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

Process one audio chunk, but with planar audio

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

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

See also

sound4cl_ProcessAudio_Planar

Parameters

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

Definition at line 1183 of file sound4cl.hpp.

10.16.3.20 SetBus()

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

Set the instance bus

Definition at line 815 of file sound4cl.hpp.

10.16.3.21 SetMetadata()

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

Adds the specific metadata to the process.

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

Parameters

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 942 of file sound4cl.hpp.

10.16.3.22 SetMetadataMulti()

```
void sound4::CInstance::SetMetadataMulti (
            const std::unordered_map< std::string, const char * > & list )  [inline]
```

Adds multiple metadata to the process.

This works like CInstance::SetMetadata but with multiple metadata at once.

See also

CInstance::SetMetadata

Parameters

list	a map of key/value pair.

Definition at line 963 of file sound4cl.hpp.

10.16.3.23 SetParam()

```
void sound4::CInstance::SetParam (
            const std::string & name,
            const std::string & value )  [inline]
```

Set a parameter value

Must be done before Create

See also

sound4cl_SetParameter

Definition at line 793 of file sound4cl.hpp.

10.16.3.24 SetPresetManager()

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

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

See also

sound4cl_SetPresetManager

Must be done before Create

Parameters

preset_manager	The custom preset manager

Definition at line 831 of file sound4cl.hpp.

10.16.3.25 SetWebServerAppHealth()

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

Set the web server /apphealth status info

Definition at line 1067 of file sound4cl.hpp.

10.16.3.26 StartWebServer()

```
bool sound4::CInstance::StartWebServer (
            int http_port,
            int https_port = 0 )  [inline]
```

Starts a Web server on requested ports

See also

sound4cl_Webserver

Definition at line 1030 of file sound4cl.hpp.

10.16.3.27 Stop()

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

Stops the instance and its web server and its update thread.

See also

sound4cl_ExitProcess,
sound4cl_StopUpdateThread and
sound4cl_Webserver_Stop

Definition at line 1105 of file sound4cl.hpp.

10.16.3.28 StopWebServer()

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

Stops the Web server

See also

sound4cl_Webserver_Stop

Definition at line 1055 of file sound4cl.hpp.

10.16.3.29 Terminate()

```
int sound4::CInstance::Terminate ( )  [inline]
```

Terminate the processing. This is useful only with file processing, to get the final status.

See also

sound4cl_TerminateProcess

Definition at line 1094 of file sound4cl.hpp.

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

- sound4cl.hpp

10.17 sound4::bigvoice::CPresetLoader Class Reference

```
#include <sound4.bigvoice.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

10.17.1 Detailed Description

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

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

10.17.2 Constructor & Destructor Documentation

10.17.2.1 CPresetLoader()

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

10.17.2.2 ~CPresetLoader()

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

10.17.3 Member Function Documentation

10.17.3.1 Exists()

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

10.17.3.2 GetAll()

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

10.17.3.3 IsReadOnly()

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

10.17.3.4 Read()

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

10.17.3.5 Remove()

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

10.17.3.6 Rename()

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

10.17.3.7 Write()

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

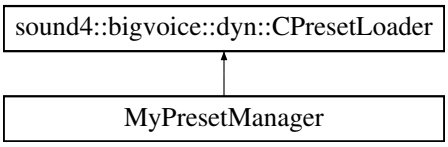
The documentation for this class was generated from the following file:

- sound4.bigvoice.cl.hpp

10.18 sound4::bigvoice::dyn::CPresetLoader Class Reference

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

Inheritance diagram for sound4::bigvoice::dyn::CPresetLoader:



Public Member Functions

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

10.18.1 Detailed Description

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

Definition at line 798 of file sound4.bigvoice.cl_dyn.hpp.

10.18.2 Constructor & Destructor Documentation

10.18.2.1 CPresetLoader()

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

10.18.2.2 ~CPresetLoader()

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

10.18.3 Member Function Documentation

10.18.3.1 Exists()

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

Implemented in MyPresetManager, and MyPresetManager.

10.18.3.2 GetAll()

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

Implemented in MyPresetManager, and MyPresetManager.

10.18.3.3 IsReadOnly()

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

Implemented in MyPresetManager, and MyPresetManager.

10.18.3.4 Read()

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

Implemented in MyPresetManager, and MyPresetManager.

10.18.3.5 Remove()

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

Implemented in MyPresetManager, and MyPresetManager.

10.18.3.6 Rename()

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

Implemented in MyPresetManager, and MyPresetManager.

10.18.3.7 Write()

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

Implemented in MyPresetManager, and MyPresetManager.

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

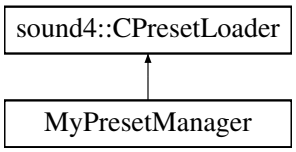
- sound4.bigvoice.cl_dyn.hpp

10.19 sound4::CPresetLoader Class Reference

Custom preset handler helper.

`#include <sound4cl.hpp>`

Inheritance diagram for sound4::CPresetLoader:



Public Member Functions

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

10.19.1 Detailed Description

Derive from this class and call CInstance::SetPresetManager

Definition at line 740 of file sound4cl.hpp.

10.19.2 Constructor & Destructor Documentation

10.19.2.1 CPresetLoader()

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

10.19.2.2 ~CPresetLoader()

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

10.19.3 Member Function Documentation

10.19.3.1 Exists()

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

Implemented in MyPresetManager, and MyPresetManager.

10.19.3.2 GetAll()

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

Implemented in MyPresetManager, and MyPresetManager.

10.19.3.3 IsReadOnly()

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

Implemented in MyPresetManager, and MyPresetManager.

10.19.3.4 Read()

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

Implemented in MyPresetManager, and MyPresetManager.

10.19.3.5 Remove()

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

Implemented in MyPresetManager, and MyPresetManager.

10.19.3.6 Rename()

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

Implemented in MyPresetManager, and MyPresetManager.

10.19.3.7 Write()

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

Implemented in MyPresetManager, and MyPresetManager.

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

- sound4cl.hpp

10.20 sound4::CProcessor Class Reference

Dynamic library interface.

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

Public Member Functions

- CProcessor ()=default
- bool Load (const std::filesystem::path &filepath)

  Loads the library.
- bool IsOk () const

  Check if the library was loaded correctly.
- void AudioConvertFrom (const uint8_t *payload, float *output, size_t nSpl, sound4cl_SampleFormat fmt)
- void AudioConvertTo (const float *input, uint8_t *payload, size_t nSpl, sound4cl_SampleFormat fmt)
- void StereoToMono (const float *input, float *output, size_t nSpl)
- void MonoToStereo (const float *input, float *output, size_t nSpl)
- void StereoToMono_Planar (const float *inputL, const float *inputR, float *output, size_t nSpl)
- void MonoToStereo_Planar (const float *input, float *outputL, float *outputR, size_t nSpl)
- void AudioMonoFromLiveStereo (const uint8_t *payload, float *output)
- void AudioMonoToLiveStereo (const float *input, uint8_t *payload)
- std::string GetVersion ()
- unsigned int GetChunkSizeInFrames ()
- std::vector< unsigned int > GetPossibleChunkSizeInFrames ()
- unsigned int GetChannelCount ()
- unsigned int GetAudioInputCount ()
- unsigned int GetAudioOutputCount ()
- unsigned int GetSampleRate ()
- std::string GetFormatName (const sound4cl_SampleFormat fmt)
- sound4cl_SampleFormat GetFormatFromName (const std::string &name)
- unsigned int GetBytesFromFormat (const sound4cl_SampleFormat fmt)
- void SetLogSeverity (LogSeverity severity)
- void SetLoggerCallback (log_cb_t cb)

Protected Member Functions

- template<typename T >
  helper::DynFuncHolder< T > GetPrefixSymbol (const std::string &name)

Protected Attributes

- helper::DynFuncHolder< decltype(sound4cl_GetVersion) > fnGetVersion
- helper::DynFuncHolder< decltype(sound4cl_GetChunkSizeInFrames) > fnGetChunkSizeInFrames
- helper::DynFuncHolder< decltype(sound4cl_GetChannelCount) > fnGetChannelCount
- helper::DynFuncHolder< decltype(sound4cl_GetAudioInputCount) > fnGetAudioInputCount
- helper::DynFuncHolder< decltype(sound4cl_GetAudioOutputCount) > fnGetAudioOutputCount
- helper::DynFuncHolder< decltype(sound4cl_GetSampleRate) > fnGetSampleRate
- helper::DynFuncHolder< decltype(sound4cl_SetLoggerCallback) > fnSetLoggerCallback
- helper::DynFuncHolder< decltype(sound4cl_SetLogSeverity) > fnSetLogSeverity
- helper::DynFuncHolder< decltype(sound4cl_NewParameters) > fnNewParameters
- helper::DynFuncHolder< decltype(sound4cl_FreeParameters) > fnFreeParameters
- helper::DynFuncHolder< decltype(sound4cl_SetParameter) > fnSetParameter
- helper::DynFuncHolder< decltype(sound4cl_GetParameter) > fnGetParameter
- helper::DynFuncHolder< decltype(sound4cl_FreeParameterValue) > fnFreeParameterValue
- helper::DynFuncHolder< decltype(sound4cl_InitProcess) > fnInitProcess
- helper::DynFuncHolder< decltype(sound4cl_InitProcess2) > fnInitProcess2
- helper::DynFuncHolder< decltype(sound4cl_TerminateProcess) > fnTerminateProcess
- helper::DynFuncHolder< decltype(sound4cl_ExitProcess) > fnExitProcess
- helper::DynFuncHolder< decltype(sound4cl_StartUpdateThread) > fnStartUpdateThread
- helper::DynFuncHolder< decltype(sound4cl_StopUpdateThread) > fnStopUpdateThread
- helper::DynFuncHolder< decltype(sound4cl_WaitUpdateThreadReady) > fnWaitUpdateThreadReady
- helper::DynFuncHolder< decltype(sound4cl_ProcessAudio) > fnProcessAudio
- helper::DynFuncHolder< decltype(sound4cl_ProcessAudio_Planar) > fnProcessAudio_Planar
- helper::DynFuncHolder< decltype(sound4cl_GetBufferIn) > fnGetBufferIn
- helper::DynFuncHolder< decltype(sound4cl_GetBufferOut) > fnGetBufferOut
- helper::DynFuncHolder< decltype(sound4cl_GetEstimatedDelay) > fnGetEstimatedDelay
- helper::DynFuncHolder< decltype(sound4cl_GetFormatName) > fnGetFormatName
- helper::DynFuncHolder< decltype(sound4cl_GetFormatFromName) > fnGetFormatFromName
- helper::DynFuncHolder< decltype(sound4cl_GetBytesFromFormat) > fnGetBytesFromFormat
- helper::DynFuncHolder< decltype(sound4cl_GetMaxPacketFrame) > fnGetMaxPacketFrame
- helper::DynFuncHolder< decltype(sound4cl_AddAudio) > fnAddAudio
- helper::DynFuncHolder< decltype(sound4cl_AddPadAudio) > fnAddPadAudio
- helper::DynFuncHolder< decltype(sound4cl_GetOutputCount) > fnGetOutputCount
- helper::DynFuncHolder< decltype(sound4cl_GetAudio) > fnGetAudio
- helper::DynFuncHolder< decltype(sound4cl_AudioConvertFrom) > fnAudioConvertFrom
- helper::DynFuncHolder< decltype(sound4cl_AudioConvertTo) > fnAudioConvertTo
- helper::DynFuncHolder< decltype(sound4cl_StereoToMono) > fnStereoToMono
- helper::DynFuncHolder< decltype(sound4cl_MonoToStereo) > fnMonoToStereo
- helper::DynFuncHolder< decltype(sound4cl_AudioMonoFromLiveStereo) > fnAudioMonoFromLiveStereo
- helper::DynFuncHolder< decltype(sound4cl_AudioMonoToLiveStereo) > fnAudioMonoToLiveStereo
- helper::DynFuncHolder< decltype(sound4cl_NewClient) > fnNewClient
- helper::DynFuncHolder< decltype(sound4cl_DeleteClient) > fnDeleteClient
- helper::DynFuncHolder< decltype(sound4cl_ProcessJson) > fnProcessJson
- helper::DynFuncHolder< decltype(sound4cl_FreeJsonAnswer) > fnFreeJsonAnswer
- helper::DynFuncHolder< decltype(sound4cl_SaveState) > fnSaveState
- helper::DynFuncHolder< decltype(sound4cl_Webserver_tcp) > fnWebserver_tcp
- helper::DynFuncHolder< decltype(sound4cl_Webserver_tcp2) > fnWebserver_tcp2
- helper::DynFuncHolder< decltype(sound4cl_Webserver) > fnWebserver
- helper::DynFuncHolder< decltype(sound4cl_Webserver_Stop) > fnWebserver_Stop
- helper::DynFuncHolder< decltype(sound4cl_Webserver_Status) > fnWebserver_Status
- helper::DynFuncHolder< decltype(sound4cl_StereoToMono_Planar) > fnStereoToMono_Planar
- helper::DynFuncHolder< decltype(sound4cl_MonoToStereo_Planar) > fnMonoToStereo_Planar
- helper::DynFuncHolder< decltype(sound4cl_SetMetadata) > fnSetMetadata
- helper::DynFuncHolder< decltype(sound4cl_GetMetadataInfos) > fnGetMetadataInfos

- helper::DynFuncHolder< decltype(sound4cl_FreeMetadataInfos) > fnFreeMetadataInfos
- helper::DynFuncHolder< decltype(sound4cl_SetPresetManager) > fnSetPresetManager
- helper::DynFuncHolder< decltype(sound4cl_PresetManager_InformChange) > fnPresetManager_InformChange
- helper::DynFuncHolder< decltype(sound4cl_GetPossibleChunkSizeInFrames) > fnGetPossibleChunkSizeInFrames
- helper::DynFuncHolder< decltype(sound4cl_GetProcessChunkFrames) > fnGetProcessChunkFrames
- helper::DynFuncHolder< decltype(sound4cl_InitProcess3) > fnInitProcess3
- helper::DynFuncHolder< decltype(sound4cl_NewBus) > fnNewBus
- helper::DynFuncHolder< decltype(sound4cl_FreeBus) > fnFreeBus
- helper::DynFuncHolder< decltype(sound4cl_SetInstanceBus) > fnSetInstanceBus
- helper::DynFuncHolder< decltype(sound4cl_SetMetadataMulti) > fnSetMetadataMulti
- helper::DynFuncHolder< decltype(sound4cl_Webserver_SetAppHealth) > fnWebserver_SetAppHealth
- helper::DynFuncHolder< decltype(sound4cl_Webserver_GetAppHealth) > fnWebserver_GetAppHealth
- helper::DynFuncHolder< decltype(sound4cl_Webserver_FreeString) > fnWebserver_FreeString

Friends

- class CBus
- class CInstance

10.20.1 Detailed Description

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

Definition at line 276 of file sound4cl.hpp.

10.20.2 Constructor & Destructor Documentation

10.20.2.1 CProcessor()

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

10.20.3 Member Function Documentation

10.20.3.1 AudioConvertFrom()

```
void sound4::CProcessor::AudioConvertFrom (
            const uint8_t * payload,
            float * output,
            size_t nSpl,
            sound4cl_SampleFormat fmt )  [inline]
```

Conversion helpers

Definition at line 573 of file sound4cl.hpp.

10.20.3.2 AudioConvertTo()

```
void sound4::CProcessor::AudioConvertTo (
            const float * input,
            uint8_t * payload,
            size_t nSpl,
            sound4cl_SampleFormat fmt )  [inline]
```

Definition at line 575 of file sound4cl.hpp.

10.20.3.3 AudioMonoFromLiveStereo()

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

Definition at line 585 of file sound4cl.hpp.

10.20.3.4 AudioMonoToLiveStereo()

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

Definition at line 587 of file sound4cl.hpp.

10.20.3.5 GetAudioInputCount()

```
unsigned int sound4::CProcessor::GetAudioInputCount ( )  [inline]
```

Get the number of inputs for this process

Remarks

All inputs have sound4cl_GetChannelCount() channels

Returns

The process input count

Definition at line 645 of file sound4cl.hpp.

10.20.3.6 GetAudioOutputCount()

```
unsigned int sound4::CProcessor::GetAudioOutputCount ( )  [inline]
```

Get the number of outputs for this process

Remarks

All outputs have sound4cl_GetChannelCount() channels

Returns

The process outputs count

Definition at line 655 of file sound4cl.hpp.

10.20.3.7 GetBytesFromFormat()

```
unsigned int sound4::CProcessor::GetBytesFromFormat (
            const sound4cl_SampleFormat fmt )  [inline]
```

Get the format byte size

Parameters

fmt	the format

Returns

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

Definition at line 692 of file sound4cl.hpp.

10.20.3.8 GetChannelCount()

```
unsigned int sound4::CProcessor::GetChannelCount ( )  [inline]
```

Get the number of channels needed by this process

Returns

The process channels count

Definition at line 635 of file sound4cl.hpp.

10.20.3.9 GetChunkSizeInFrames()

```
unsigned int sound4::CProcessor::GetChunkSizeInFrames ( )  [inline]
```

Get the number of frames in a process chunk.

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

Note

A Frame is a set of sound4cl_GetChannelCount samples

Returns

The process chunk size in frame

Definition at line 606 of file sound4cl.hpp.

10.20.3.10 GetFormatFromName()

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

Convert string to format

Parameters

name	the format string

Returns

the matching format, or INVALID_FORMAT if invalid.

Definition at line 683 of file sound4cl.hpp.

10.20.3.11 GetFormatName()

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

Convert format to string

Parameters

fmt	the format

Returns

the string name of the format

Definition at line 674 of file sound4cl.hpp.

10.20.3.12 GetPossibleChunkSizeInFrames()

```
std::vector< unsigned int > sound4::CProcessor::GetPossibleChunkSizeInFrames ( )  [inline]
```

Get the possibles number of frames in a process chunk.

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

Note

A Frame is a set of sound4cl_GetChannelCount samples

Returns

An vector of possible process chunk size in frame

Definition at line 617 of file sound4cl.hpp.

10.20.3.13 GetPrefixSymbol()

```
template<typename T >
helper::DynFuncHolder< T > sound4::CProcessor::GetPrefixSymbol (
            const std::string & name )  [inline], [protected]
```

Definition at line 370 of file sound4cl.hpp.

10.20.3.14 GetSampleRate()

```
unsigned int sound4::CProcessor::GetSampleRate ( )  [inline]
```

Get the processing sample-rate

Remarks

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

Returns

The process sample-rate in Hz

Definition at line 665 of file sound4cl.hpp.

10.20.3.15 GetVersion()

```
std::string sound4::CProcessor::GetVersion ( )  [inline]
```

Get the process version string

Returns

version string

Definition at line 595 of file sound4cl.hpp.

10.20.3.16 IsOk()

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

Returns

true if OK.

false if not OK.

Definition at line 565 of file sound4cl.hpp.

10.20.3.17 Load()

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

Parameters

filepath	the path of the library to load.

Returns

true if library was loaded correctly.

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

Definition at line 385 of file sound4cl.hpp.

10.20.3.18 MonoToStereo()

```
void sound4::CProcessor::MonoToStereo (
            const float * input,
            float * output,
            size_t nSpl )  [inline]
```

Definition at line 579 of file sound4cl.hpp.

10.20.3.19 MonoToStereo_Planar()

```
void sound4::CProcessor::MonoToStereo_Planar (
            const float * input,
            float * outputL,
            float * outputR,
            size_t nSpl )  [inline]
```

Definition at line 583 of file sound4cl.hpp.

10.20.3.20 SetLoggerCallback()

```
void sound4::CProcessor::SetLoggerCallback (
            log_cb_t cb )  [inline]
```

Sets the log callback. If not set, console output will be used.

Definition at line 704 of file sound4cl.hpp.

10.20.3.21 SetLogSeverity()

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

Sets the log severity (global)

Definition at line 698 of file sound4cl.hpp.

10.20.3.22 StereoToMono()

```
void sound4::CProcessor::StereoToMono (
            const float * input,
            float * output,
            size_t nSpl )  [inline]
```

Definition at line 577 of file sound4cl.hpp.

10.20.3.23 StereoToMono_Planar()

```
void sound4::CProcessor::StereoToMono_Planar (
            const float * inputL,
            const float * inputR,
            float * output,
            size_t nSpl )  [inline]
```

Definition at line 581 of file sound4cl.hpp.

10.20.4 Friends And Related Symbol Documentation

10.20.4.1 CBus

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

Definition at line 282 of file sound4cl.hpp.

10.20.4.2 CInstance

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

Definition at line 283 of file sound4cl.hpp.

10.20.5 Member Data Documentation

10.20.5.1 fnAddAudio

```
helper::DynFuncHolder< decltype(sound4cl_AddAudio ) > sound4::CProcessor::fnAddAudio  [protected]
```

Definition at line 315 of file sound4cl.hpp.

10.20.5.2 fnAddPadAudio

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

Definition at line 316 of file sound4cl.hpp.

10.20.5.3 fnAudioConvertFrom

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

Definition at line 319 of file sound4cl.hpp.

10.20.5.4 fnAudioConvertTo

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

Definition at line 320 of file sound4cl.hpp.

10.20.5.5 fnAudioMonoFromLiveStereo

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

Definition at line 323 of file sound4cl.hpp.

10.20.5.6 fnAudioMonoToLiveStereo

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

Definition at line 324 of file sound4cl.hpp.

10.20.5.7 fnDeleteClient

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

Definition at line 326 of file sound4cl.hpp.

10.20.5.8 fnExitProcess

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

Definition at line 302 of file sound4cl.hpp.

10.20.5.9 fnFreeBus

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

Definition at line 358 of file sound4cl.hpp.

10.20.5.10 fnFreeJsonAnswer

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

Definition at line 328 of file sound4cl.hpp.

10.20.5.11 fnFreeMetadataInfos

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

Definition at line 345 of file sound4cl.hpp.

10.20.5.12 fnFreeParameters

```
helper::DynFuncHolder< decltype(sound4cl_FreeParameters ) > sound4::CProcessor::fnFreeParameters
[protected]
```

Definition at line 295 of file sound4cl.hpp.

10.20.5.13 fnFreeParameterValue

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

Definition at line 298 of file sound4cl.hpp.

10.20.5.14 fnGetAudio

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

Definition at line 318 of file sound4cl.hpp.

10.20.5.15 fnGetAudioInputCount

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

Definition at line 289 of file sound4cl.hpp.

10.20.5.16 fnGetAudioOutputCount

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

Definition at line 290 of file sound4cl.hpp.

10.20.5.17 fnGetBufferIn

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

Definition at line 308 of file sound4cl.hpp.

10.20.5.18 fnGetBufferOut

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

Definition at line 309 of file sound4cl.hpp.

10.20.5.19 fnGetBytesFromFormat

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

Definition at line 313 of file sound4cl.hpp.

10.20.5.20 fnGetChannelCount

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

Definition at line 288 of file sound4cl.hpp.

10.20.5.21 fnGetChunkSizeInFrames

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

Definition at line 287 of file sound4cl.hpp.

10.20.5.22 fnGetEstimatedDelay

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

Definition at line 310 of file sound4cl.hpp.

10.20.5.23 fnGetFormatFromName

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

Definition at line 312 of file sound4cl.hpp.

10.20.5.24 fnGetFormatName

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

Definition at line 311 of file sound4cl.hpp.

10.20.5.25 fnGetMaxPacketFrame

helper::DynFuncHolder< decltype(sound4cl_GetMaxPacketFrame) > sound4::CProcessor::fnGetMaxPacketFrame [protected]

Definition at line 314 of file sound4cl.hpp.

10.20.5.26 fnGetMetadataInfos

helper::DynFuncHolder< decltype(sound4cl_GetMetadataInfos) > sound4::CProcessor::fnGetMetadataInfos [protected]

Definition at line 344 of file sound4cl.hpp.

10.20.5.27 fnGetOutputCount

helper::DynFuncHolder< decltype(sound4cl_GetOutputCount) > sound4::CProcessor::fnGetOutputCount [protected]

Definition at line 317 of file sound4cl.hpp.

10.20.5.28 fnGetParameter

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

Definition at line 297 of file sound4cl.hpp.

10.20.5.29 fnGetPossibleChunkSizeInFrames

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

Definition at line 352 of file sound4cl.hpp.

10.20.5.30 fnGetProcessChunkFrames

helper::DynFuncHolder< decltype(sound4cl_GetProcessChunkFrames) > sound4::CProcessor::fnGetProcessChunkFrames [protected]

Definition at line 353 of file sound4cl.hpp.

10.20.5.31 fnGetSampleRate

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

Definition at line 291 of file sound4cl.hpp.

10.20.5.32 fnGetVersion

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

Definition at line 286 of file sound4cl.hpp.

10.20.5.33 fnInitProcess

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

Definition at line 299 of file sound4cl.hpp.

10.20.5.34 fnInitProcess2

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

Definition at line 300 of file sound4cl.hpp.

10.20.5.35 fnInitProcess3

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

Definition at line 354 of file sound4cl.hpp.

10.20.5.36 fnMonoToStereo

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

Definition at line 322 of file sound4cl.hpp.

10.20.5.37 fnMonoToStereo_Planar

helper::DynFuncHolder< decltype(sound4cl_MonoToStereo_Planar) > sound4::CProcessor::fnMono↩ ToStereo_Planar [protected]

Definition at line 340 of file sound4cl.hpp.

10.20.5.38 fnNewBus

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

Definition at line 357 of file sound4cl.hpp.

10.20.5.39 fnNewClient

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

Definition at line 325 of file sound4cl.hpp.

10.20.5.40 fnNewParameters

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

Definition at line 294 of file sound4cl.hpp.

10.20.5.41 fnPresetManager_InformChange

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

Definition at line 349 of file sound4cl.hpp.

10.20.5.42 fnProcessAudio

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

Definition at line 306 of file sound4cl.hpp.

10.20.5.43 fnProcessAudio_Planar

```
helper::DynFuncHolder< decltype(sound4cl_ProcessAudio_Planar ) > sound4::CProcessor::fn↩
ProcessAudio_Planar  [protected]
```

Definition at line 307 of file sound4cl.hpp.

10.20.5.44 fnProcessJson

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

Definition at line 327 of file sound4cl.hpp.

10.20.5.45 fnSaveState

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

Definition at line 329 of file sound4cl.hpp.

10.20.5.46 fnSetInstanceBus

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

Definition at line 359 of file sound4cl.hpp.

10.20.5.47 fnSetLoggerCallback

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

Definition at line 292 of file sound4cl.hpp.

10.20.5.48 fnSetLogSeverity

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

Definition at line 293 of file sound4cl.hpp.

10.20.5.49 fnSetMetadata

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

Definition at line 343 of file sound4cl.hpp.

10.20.5.50 fnSetMetadataMulti

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

Definition at line 362 of file sound4cl.hpp.

10.20.5.51 fnSetParameter

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

Definition at line 296 of file sound4cl.hpp.

10.20.5.52 fnSetPresetManager

helper::DynFuncHolder< decltype(sound4cl_SetPresetManager) > sound4::CProcessor::fnSetPresetManager [protected]

Definition at line 348 of file sound4cl.hpp.

10.20.5.53 fnStartUpdateThread

helper::DynFuncHolder< decltype(sound4cl_StartUpdateThread) > sound4::CProcessor::fnStartUpdateThread [protected]

Definition at line 303 of file sound4cl.hpp.

10.20.5.54 fnStereoToMono

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

Definition at line 321 of file sound4cl.hpp.

10.20.5.55 fnStereoToMono_Planar

helper::DynFuncHolder< decltype(sound4cl_StereoToMono_Planar) > sound4::CProcessor::fnStereoToMono_Planar [protected]

Definition at line 339 of file sound4cl.hpp.

10.20.5.56 fnStopUpdateThread

helper::DynFuncHolder< decltype(sound4cl_StopUpdateThread) > sound4::CProcessor::fnStopUpdateThread [protected]

Definition at line 304 of file sound4cl.hpp.

10.20.5.57 fnTerminateProcess

helper::DynFuncHolder< decltype(sound4cl_TerminateProcess) > sound4::CProcessor::fnTerminateProcess [protected]

Definition at line 301 of file sound4cl.hpp.

10.20.5.58 fnWaitUpdateThreadReady

helper::DynFuncHolder< decltype(sound4cl_WaitUpdateThreadReady) > sound4::CProcessor::fnWaitUpdateThreadReady [protected]

Definition at line 305 of file sound4cl.hpp.

10.20.5.59 fnWebserver

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

Definition at line 334 of file sound4cl.hpp.

10.20.5.60 fnWebserver_FreeString

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

Definition at line 367 of file sound4cl.hpp.

10.20.5.61 fnWebserver_GetAppHealth

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

Definition at line 366 of file sound4cl.hpp.

10.20.5.62 fnWebserver_SetAppHealth

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

Definition at line 365 of file sound4cl.hpp.

10.20.5.63 fnWebserver_Status

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

Definition at line 336 of file sound4cl.hpp.

10.20.5.64 fnWebserver_Stop

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

Definition at line 335 of file sound4cl.hpp.

10.20.5.65 fnWebserver_tcp

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

Definition at line 332 of file sound4cl.hpp.

10.20.5.66 fnWebserver_tcp2

```
helper::DynFuncHolder< decltype(sound4cl_Webserver_tcp2 ) > sound4::CProcessor::fnWebserver_↩
tcp2  [protected]
```

Definition at line 333 of file sound4cl.hpp.

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

- sound4cl.hpp

10.21 sound4::bigvoice::dyn::helper::DynFuncHolder< T > Class Template Reference

Helper to wrap a function pointer.

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

Public Member Functions

- DynFuncHolder ()=default
- DynFuncHolder (DynFunc_t a_ptr)
- operator bool () const
- bool IsOk () const
- operator T* () const

10.21.1 Detailed Description

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

Parameters

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

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

10.21.2 Constructor & Destructor Documentation

10.21.2.1 DynFuncHolder() [1/2]

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

10.21.2.2 DynFuncHolder() [2/2]

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

Definition at line 97 of file sound4.bigvoice.cl_dyn.hpp.

10.21.3 Member Function Documentation

10.21.3.1 IsOk()

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

Definition at line 99 of file sound4.bigvoice.cl_dyn.hpp.

10.21.3.2 operator bool()

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

Definition at line 98 of file sound4.bigvoice.cl_dyn.hpp.

10.21.3.3 operator T*()

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

Definition at line 100 of file sound4.bigvoice.cl_dyn.hpp.

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

- sound4.bigvoice.cl_dyn.hpp

10.22 sound4::helper::DynFuncHolder< T > Class Template Reference

Helper to wrap a function pointer.

`#include <sound4cl.hpp>`

Public Member Functions

- DynFuncHolder ()=default
- DynFuncHolder (DynFunc_t a_ptr)
- operator bool () const
- bool IsOk () const
- operator T* () const

10.22.1 Detailed Description

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

Parameters

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

Definition at line 121 of file sound4cl.hpp.

10.22.2 Constructor & Destructor Documentation

10.22.2.1 DynFuncHolder() [1/2]

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

10.22.2.2 DynFuncHolder() [2/2]

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

Definition at line 126 of file sound4cl.hpp.

10.22.3 Member Function Documentation

10.22.3.1 IsOk()

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

Definition at line 128 of file sound4cl.hpp.

10.22.3.2 operator bool()

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

Definition at line 127 of file sound4cl.hpp.

10.22.3.3 operator T*()

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

Definition at line 129 of file sound4cl.hpp.

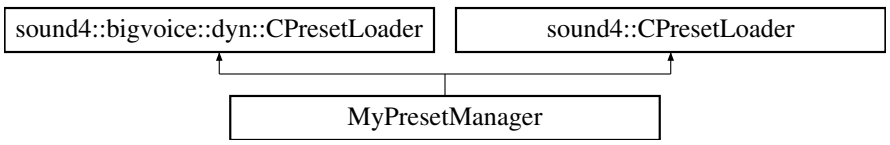
The documentation for this class was generated from the following file:

- sound4cl.hpp

10.23 MyPresetManager Class Reference

[Custom preset manager]

Inheritance diagram for MyPresetManager:



Public Member Functions

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

Public Member Functions inherited from sound4::bigvoice::dyn::CPresetLoader

- CPresetLoader ()=default
- virtual ~CPresetLoader ()=default

Public Member Functions inherited from sound4::CPresetLoader

- CPresetLoader ()=default
- virtual ~CPresetLoader ()=default

10.23.1 Detailed Description

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

10.23.2 Constructor & Destructor Documentation

10.23.2.1 MyPresetManager() [1/2]

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

10.23.2.2 ~MyPresetManager() [1/2]

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

10.23.2.3 MyPresetManager() [2/2]

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

10.23.2.4 ~MyPresetManager() [2/2]

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

10.23.3 Member Function Documentation

10.23.3.1 Exists() [1/2]

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

Implements sound4::bigvoice::dyn::CPresetLoader.

Definition at line 137 of file sound4.bigvoice.cl-pipe-dyn.cpp.

10.23.3.2 Exists() [2/2]

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

Implements sound4::bigvoice::dyn::CPresetLoader.

Definition at line 121 of file sound4cl-pipe.cpp.

10.23.3.3 GetAll() [1/2]

```
virtual std::vector< std::filesystem::path > MyPresetManager::GetAll ( )  [inline], [override],
[virtual]
```

Implements sound4::bigvoice::dyn::CPresetLoader.

Definition at line 153 of file sound4.bigvoice.cl-pipe-dyn.cpp.

10.23.3.4 GetAll() [2/2]

```
virtual std::vector< std::filesystem::path > MyPresetManager::GetAll ( )  [inline], [override],
[virtual]
```

Implements sound4::bigvoice::dyn::CPresetLoader.

Definition at line 137 of file sound4cl-pipe.cpp.

10.23.3.5 IsReadOnly() [1/2]

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

Implements sound4::bigvoice::dyn::CPresetLoader.

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

10.23.3.6 IsReadOnly() [2/2]

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

Implements sound4::bigvoice::dyn::CPresetLoader.

Definition at line 120 of file sound4cl-pipe.cpp.

10.23.3.7 Read() [1/2]

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

Implements sound4::bigvoice::dyn::CPresetLoader.

Definition at line 167 of file sound4.bigvoice.cl-pipe-dyn.cpp.

10.23.3.8 Read() [2/2]

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

Implements sound4::bigvoice::dyn::CPresetLoader.

Definition at line 151 of file sound4cl-pipe.cpp.

10.23.3.9 Remove() [1/2]

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

Implements sound4::bigvoice::dyn::CPresetLoader.

Definition at line 142 of file sound4.bigvoice.cl-pipe-dyn.cpp.

10.23.3.10 Remove() [2/2]

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

Implements sound4::bigvoice::dyn::CPresetLoader.

Definition at line 126 of file sound4cl-pipe.cpp.

10.23.3.11 Rename() [1/2]

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

Implements sound4::bigvoice::dyn::CPresetLoader.

Definition at line 147 of file sound4.bigvoice.cl-pipe-dyn.cpp.

10.23.3.12 Rename() [2/2]

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

Implements sound4::bigvoice::dyn::CPresetLoader.

Definition at line 131 of file sound4cl-pipe.cpp.

10.23.3.13 SetupPath() [1/2]

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

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

10.23.3.14 SetupPath() [2/2]

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

Definition at line 108 of file sound4cl-pipe.cpp.

10.23.3.15 Write() [1/2]

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

Implements sound4::bigvoice::dyn::CPresetLoader.

Definition at line 177 of file sound4.bigvoice.cl-pipe-dyn.cpp.

10.23.3.16 Write() [2/2]

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

Implements sound4::bigvoice::dyn::CPresetLoader.

Definition at line 161 of file sound4cl-pipe.cpp.

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

- sound4.bigvoice.cl-pipe-dyn.cpp
- sound4cl-pipe.cpp

10.24 sound4::bigvoice::dyn::helper::SampleFormat< T > Struct Template Reference

Helper for Sample format types.

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

10.24.1 Detailed Description

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

Template Parameters

T	the C type for one audio sample

Definition at line 535 of file sound4.bigvoice.cl_dyn.hpp.

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

- sound4.bigvoice.cl_dyn.hpp

10.25 sound4::bigvoice::helper::SampleFormat< T > Struct Template Reference

Helper for Sample format types.

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

10.25.1 Detailed Description

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

Template Parameters

T	the C type for one audio sample

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

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

- sound4.bigvoice.cl.hpp

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

Helper for Sample format types.

`#include <sound4cl.hpp>`

10.26.1 Detailed Description

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

Template Parameters

T	the C type for one audio sample

Definition at line 91 of file sound4cl.hpp.

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

- sound4cl.hpp

10.27 sound4::bigvoice::dyn::helper::SampleFormat< float > Struct Reference

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

Public Attributes

- const bigvoice_SampleFormat format = F32_NATIVE

10.27.1 Detailed Description

Definition at line 546 of file sound4.bigvoice.cl_dyn.hpp.

10.27.2 Member Data Documentation

10.27.2.1 format

```
const bigvoice_SampleFormat sound4::bigvoice::dyn::helper::SampleFormat< float >::format =
F32_NATIVE
```

Definition at line 547 of file sound4.bigvoice.cl_dyn.hpp.

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

- sound4.bigvoice.cl_dyn.hpp

10.28 sound4::bigvoice::helper::SampleFormat< float > Struct Reference

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

Public Attributes

- const bigvoice_SampleFormat format = F32_NATIVE

10.28.1 Detailed Description

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

10.28.2 Member Data Documentation

10.28.2.1 format

```
const bigvoice_SampleFormat sound4::bigvoice::helper::SampleFormat< float >::format = F32_NATIVE
```

Definition at line 128 of file sound4.bigvoice.cl.hpp.

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

- sound4.bigvoice.cl.hpp

10.29 sound4::helper::SampleFormat< float > Struct Reference

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

Public Attributes

- const sound4cl_SampleFormat format = F32_NATIVE

10.29.1 Detailed Description

Definition at line 102 of file sound4cl.hpp.

10.29.2 Member Data Documentation

10.29.2.1 format

```
const sound4cl_SampleFormat sound4::helper::SampleFormat< float >::format = F32_NATIVE
```

Definition at line 103 of file sound4cl.hpp.

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

- sound4cl.hpp

10.30 sound4::bigvoice::dyn::helper::SampleFormat< int16_t > Struct Reference

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

Public Attributes

- const bigvoice_SampleFormat format = S16_NATIVE

10.30.1 Detailed Description

Definition at line 538 of file sound4.bigvoice.cl_dyn.hpp.

10.30.2 Member Data Documentation

10.30.2.1 format

```
const bigvoice_SampleFormat sound4::bigvoice::dyn::helper::SampleFormat< int16_t >::format =
S16_NATIVE
```

Definition at line 539 of file sound4.bigvoice.cl_dyn.hpp.

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

- sound4.bigvoice.cl_dyn.hpp

10.31 sound4::bigvoice::helper::SampleFormat< int16_t > Struct Reference

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

Public Attributes

- const bigvoice_SampleFormat format = S16_NATIVE

10.31.1 Detailed Description

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

10.31.2 Member Data Documentation

10.31.2.1 format

`const bigvoice_SampleFormat sound4::bigvoice::helper::SampleFormat< int16_t >::format = S16_NATIVE`

Definition at line 120 of file sound4.bigvoice.cl.hpp.

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

- sound4.bigvoice.cl.hpp

10.32 sound4::helper::SampleFormat< int16_t > Struct Reference

`#include <sound4cl.hpp>`

Public Attributes

- const sound4cl_SampleFormat format = S16_NATIVE

10.32.1 Detailed Description

Definition at line 94 of file sound4cl.hpp.

10.32.2 Member Data Documentation

10.32.2.1 format

`const sound4cl_SampleFormat sound4::helper::SampleFormat< int16_t >::format = S16_NATIVE`

Definition at line 95 of file sound4cl.hpp.

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

- sound4cl.hpp

10.33 sound4::bigvoice::dyn::helper::SampleFormat< int32_t > Struct Reference

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

Public Attributes

- const bigvoice_SampleFormat format = S32_NATIVE

10.33.1 Detailed Description

Definition at line 542 of file sound4.bigvoice.cl_dyn.hpp.

10.33.2 Member Data Documentation

10.33.2.1 format

`const bigvoice_SampleFormat sound4::bigvoice::dyn::helper::SampleFormat< int32_t >::format = S32_NATIVE`

Definition at line 543 of file sound4.bigvoice.cl_dyn.hpp.

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

- sound4.bigvoice.cl_dyn.hpp

10.34 sound4::bigvoice::helper::SampleFormat< int32_t > Struct Reference

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

Public Attributes

- const bigvoice_SampleFormat format = S32_NATIVE

10.34.1 Detailed Description

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

10.34.2 Member Data Documentation

10.34.2.1 format

```
const bigvoice_SampleFormat sound4::bigvoice::helper::SampleFormat< int32_t >::format = S32_NATIVE
```

Definition at line 124 of file sound4.bigvoice.cl.hpp.

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

- sound4.bigvoice.cl.hpp

10.35 sound4::helper::SampleFormat< int32_t > Struct Reference

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

Public Attributes

- const sound4cl_SampleFormat format = S32_NATIVE

10.35.1 Detailed Description

Definition at line 98 of file sound4cl.hpp.

10.35.2 Member Data Documentation

10.35.2.1 format

```
const sound4cl_SampleFormat sound4::helper::SampleFormat< int32_t >::format = S32_NATIVE
```

Definition at line 99 of file sound4cl.hpp.

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

- sound4cl.hpp

10.36 SOUND4_ProcessInfo Struct Reference

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

Public Attributes

- size_t struct_size

  The size of total this structure, for compatibility checking.
- const char * name

  The name of the processor.
- const char * prefix

  The prefix of this processor functions.
- const char * shortname

  A short name of the processor, single word.
- const char * version

  Library version.

10.36.1 Detailed Description

Information on this processor.

See also

SOUND4_GetProcessInfo

Definition at line 203 of file sound4.bigvoice.cl.h.

10.36.2 Member Data Documentation

10.36.2.1 name

```
const char* SOUND4_ProcessInfo::name
```

Definition at line 205 of file sound4.bigvoice.cl.h.

10.36.2.2 prefix

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

Definition at line 206 of file sound4.bigvoice.cl.h.

10.36.2.3 shortname

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

Definition at line 207 of file sound4.bigvoice.cl.h.

10.36.2.4 struct_size

```
size_t SOUND4_ProcessInfo::struct_size
```

Definition at line 204 of file sound4.bigvoice.cl.h.

10.36.2.5 version

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

Definition at line 208 of file sound4.bigvoice.cl.h.

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

- sound4.bigvoice.cl.h

10.37 sound4cl_SOUND4_ProcessInfo Struct Reference

```
#include <sound4cl_cdef.h>
```

Public Attributes

- size_t struct_size
  The size of total this structure, for compatibility checking.
- const char * name
  The name of the processor.
- const char * prefix
  The prefix of this processor functions.
- const char * shortname
  A short name of the processor, single word.
- const char * version
  Library version.

10.37.1 Detailed Description

Definition at line 34 of file sound4cl_cdef.h.

10.37.2 Member Data Documentation

10.37.2.1 name

```
const char* sound4cl_SOUND4_ProcessInfo::name
```

Definition at line 36 of file sound4cl_cdef.h.

10.37.2.2 prefix

```
const char* sound4cl_SOUND4_ProcessInfo::prefix
```

Definition at line 37 of file sound4cl_cdef.h.

10.37.2.3 shortname

```
const char* sound4cl_SOUND4_ProcessInfo::shortname
```

Definition at line 38 of file sound4cl_cdef.h.

10.37.2.4 struct_size

```
size_t sound4cl_SOUND4_ProcessInfo::struct_size
```

Definition at line 35 of file sound4cl_cdef.h.

10.37.2.5 version

```
const char* sound4cl_SOUND4_ProcessInfo::version
```

Definition at line 39 of file sound4cl_cdef.h.

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

- sound4cl_cdef.h

Chapter 11

File Documentation

11.1 documentation.h File Reference

11.2 documentation.h

Go to the documentation of this file.
00001

11.3 sound4.bigvoice.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.bigvoice.cl.hpp"
```

Functions

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

11.3.1 Function Documentation

11.3.1.1 GetEnv()

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

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

11.3.1.2 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.bigvoice.cl-pipe-cxx.cpp.

11.3.1.3 MyLogger()

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

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

11.3.1.4 ProcessLoop()

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

[Processing Loop]

[Processing Loop]

Definition at line 48 of file sound4.bigvoice.cl-pipe-cxx.cpp.

11.3.1.5 Usage()

```
void Usage (
            const char * progname )
```

Definition at line 79 of file sound4.bigvoice.cl-pipe-cxx.cpp.

11.4 sound4.bigvoice.cl-pipe-cxx.cpp

Go to the documentation of this file.

```
00001
00004 #include <unistd.h>
00005 #include <iostream>
00006 #include <strings.h>
00007 #include <sys/types.h>
00008 #include <filesystem>
00009 #if defined(_WIN32) || defined(WIN32)
00010     #include <io.h>
00011     #include <fcntl.h>
00012 #endif
00013
00014 #include "sound4.bigvoice.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::bigvoice::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::bigvoice::InputSampleSize, stdin) != sound4::bigvoice::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::bigvoice::OutputSampleSize,
      stdout) != sound4::bigvoice::OutputSampleSize) {
00072             std::cerr « "Failed to write audio output" « std::endl;
00073             stopthread=true;
00074         }
00075     }
00077 }
00078
00079 void Usage(const char *progname)
00080 {
00081     std::cerr « "Usage: " « progname « std::endl;;
00082     std::cerr « "  Reads audio from stdin and return processed audio in stdout" « std::endl;
00083     std::cerr « "Environment:"« std::endl;;
00084     std::cerr « " - S4STORE_PATH            : path to store config [default: (no storage)]"«
      std::endl;;
00085     std::cerr « " - S4LOGINKEY              : SOUND4 Login key (or S4USERID)"« std::endl;;
00086     std::cerr « " - RADIO_NAME              : SOUND4 Radio Name"« std::endl;;
00087     std::cerr « " - ACCESS_KEY_ID           : SOUND4 Access Key ID"« std::endl;;
00088     std::cerr « " - ACCESS_KEY_SECRET       : SOUND4 Access Key Secret"« std::endl;;
00089     std::cerr « std::endl;
00090     std::cerr « "Example:"« std::endl;
```

```
00091     std::cerr « "ffmpeg -re -i file.wav -ar "«sound4::bigvoice::GetSampleRate()«" -ac
      "«sound4::bigvoice::GetChannelCount()«" -f f32le - | "«progname«" | ffplay -f f32le -ar
      "«sound4::bigvoice::GetSampleRate()«" -ac "«sound4::bigvoice::GetChannelCount()«" -i -" « std::endl;;
00092     #if defined(_WIN32) || defined(WIN32)
00093         std::cerr « "ffmpeg -f dshow -i audio=\"Microphone\" -ar "«sound4::bigvoice::GetSampleRate()«"
      -ac "«sound4::bigvoice::GetChannelCount()«" -f f32le - | "«progname«" | ffplay -f f32le -ar
      "«sound4::bigvoice::GetSampleRate()«" -ac "«sound4::bigvoice::GetChannelCount()«" -i -" « std::endl;;
00094     #else
00095         std::cerr « "ffmpeg -f pulse -i default -ar "«sound4::bigvoice::GetSampleRate()«" -ac
      "«sound4::bigvoice::GetChannelCount()«" -f f32le - | "«progname«" | ffplay -f f32le -ar
      "«sound4::bigvoice::GetSampleRate()«" -ac "«sound4::bigvoice::GetChannelCount()«" -i -" « std::endl;;
00096     #endif
00097 }
00098
00099 void MyLogger(sound4::bigvoice::LogSeverity severity,const std::string& msg)
00100 {
00101     std::cerr « msg « std::endl;
00102 }
00103
00104 int main(int argc, const char **argv)
00105 {
00106 #if (BIGVOICE_HAS_WEBSERVER)
00107     int listen_port = 8080;
00108 #endif // (BIGVOICE_HAS_WEBSERVER)
00109     auto store_path_str=GetEnv("S4STORE_PATH");
00110
00111     // Get credentials from environment
00112     auto LoginKey=GetEnv("S4LOGINKEY");
00113     if (LoginKey.empty()) {
00114         LoginKey=GetEnv("S4USERID");
00115     }
00116     auto RadioName=GetEnv("RADIO_NAME");
00117     auto KeyId=GetEnv("ACCESS_KEY_ID");
00118     auto AccessKey=GetEnv("ACCESS_KEY_SECRET");
00119
00120     const char *progname=argv[0];
00121
00122     if (argc>1) {
00123         if ((strcasecmp(argv[1],"-h")==0) || (strcasecmp(argv[1],"--help")==0)) {
00124             Usage(progname);
00125             return 0;
00126         }
00127     }
00128     if (LoginKey.empty() || RadioName.empty() || KeyId.empty() || AccessKey.empty()) {
00129         std::cerr « "Missing credentials in environment" « std::endl;
00130         Usage(progname);
00131         return 1;
00132     }
00133     std::filesystem::path store_path;
00134     if (!store_path_str.empty()) {
00135         std::error_code ec;
00136         store_path = std::filesystem::path(store_path_str);
00137         if (!std::filesystem::exists(store_path,ec) || !std::filesystem::is_directory(store_path,ec))
      {
00138             std::cerr « "Storage folder '"«store_path.c_str()«"' is not a valid directory" «
      std::endl;
00139             Usage(progname);
00140             return 1;
00141         }
00142     }
00143
00145     sound4::bigvoice::SetLogSeverity(verbose5);
00146     sound4::bigvoice::SetLoggerCallback(MyLogger);
00147
00148     sound4::bigvoice::CInstance instance;
00149     instance.SetParam("ADMIN_USER","admin");
00150     instance.SetParam("ADMIN_SECRET","admin");
00152
00153     std::cerr « "Creating" « std::endl;
00155     if (!instance.Create(LoginKey, RadioName, KeyId, AccessKey, store_path)) {
00156         std::cerr « "Failed to create the processing instance" « std::endl;
00157         return 1;
00158     }
00160
00161 #if (BIGVOICE_HAS_WEBSERVER)
00162     std::cerr « "Starting web server" « std::endl;
00164     if (!instance.StartWebServer(listen_port,0)) {
00165         std::cerr « "Failed to create the web server on port " « listen_port « std::endl;
00166         return 1;
00167     }
00169 #endif // (BIGVOICE_HAS_WEBSERVER)
00170
00171     {
00173         auto client = instance.NewClient();
00174         auto answer = client->ProcessJson(R"JSON({"get":{"processorname":null}})JSON");
00175         std::cerr « "Request for processor name returned "«answer«std::endl;
00177     }
```

```
00178
00179     ProcessLoop(instance);
00180
00181     std::cerr « "Stopping now" « std::endl;
00182     instance.Stop();
00183
00184     std::cerr « "End" « std::endl;
00185     return 0;
00186 }
```

11.5 sound4.bigvoice.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.bigvoice.cl_dyn.hpp"
```

Classes

- class MyPresetManager

  [Custom preset manager]

Functions

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

  [Custom preset manager]

11.5.1 Function Documentation

11.5.1.1 GetEnv()

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

Definition at line 51 of file sound4.bigvoice.cl-pipe-dyn.cpp.

11.5.1.2 main()

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

[Dynamic loading]

[Dynamic loading]

[Setup and sanity checks]

[Setup and sanity checks]

[Custom preset manager loading]

[Custom preset manager loading]

[Create the processing instance]

[Create the processing instance]

[Communicating with the instance]

[Communicating with the instance]

Definition at line 198 of file sound4.bigvoice.cl-pipe-dyn.cpp.

11.5.1.3 MyLogger()

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

Definition at line 113 of file sound4.bigvoice.cl-pipe-dyn.cpp.

11.5.1.4 ProcessLoop()

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

[Processing Loop]

[Processing Loop]

Definition at line 60 of file sound4.bigvoice.cl-pipe-dyn.cpp.

11.5.1.5 Usage()

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

Definition at line 91 of file sound4.bigvoice.cl-pipe-dyn.cpp.

11.6 sound4.bigvoice.cl-pipe-dyn.cpp

Go to the documentation of this file.

```
00001
00004 #include <unistd.h>
00005 #include <iostream>
00006 #include <strings.h>
00007 #include <sys/types.h>
00008 #include <filesystem>
00009 #include <fstream>
00010 #include <sstream>
00011 #if defined(_WIN32) || defined(WIN32)
00012     #include <io.h>
00013     #include <fcntl.h>
00014 #endif
00015
00016 #include "sound4.bigvoice.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::bigvoice::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::bigvoice::dyn::InputSampleSize, stdin) != sound4::bigvoice::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::bigvoice::dyn::OutputSampleSize, stdout) != sound4::bigvoice::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::bigvoice::dyn::CDynLib &dynlib)
00092 {
00093     std::cerr « "Usage: " « progname « std::endl;;
00094     std::cerr « "  Reads audio from stdin and return processed audio in stdout" « std::endl;
00095     std::cerr « "Environment:"« std::endl;;
00096     std::cerr « " - S4STORE_PATH            : path to store config [default: (no storage)]"«
      std::endl;;
00097     std::cerr « " - S4LOGINKEY              : SOUND4 Login key (or S4USERID)"« std::endl;;
00098     std::cerr « " - RADIO_NAME              : SOUND4 Radio Name"« std::endl;;
00099     std::cerr « " - ACCESS_KEY_ID           : SOUND4 Access Key ID"« std::endl;;
00100     std::cerr « " - ACCESS_KEY_SECRET       : SOUND4 Access Key Secret"« std::endl;;
00101     if (dynlib.IsOk()) {
00102         std::cerr « std::endl;
00103         std::cerr « "Example:"« std::endl;
00104         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;;
00105         #if defined(_WIN32) || defined(WIN32)
00106             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;;
00107         #else
00108             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;;
00109         #endif
```

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

```
00198 int main(int argc, const char **argv)
00199 {
00200 #if (BIGVOICE_HAS_WEBSERVER)
00201     int listen_port = 8080;
00202 #endif // (BIGVOICE_HAS_WEBSERVER)
00203     auto store_path_str=GetEnv("S4STORE_PATH");
00204 
00205     // Get credentials from environment
00206     auto LoginKey=GetEnv("S4LOGINKEY");
00207     if (LoginKey.empty()) {
00208         LoginKey=GetEnv("S4USERID");
00209     }
00210     auto RadioName=GetEnv("RADIO_NAME");
00211     auto KeyId=GetEnv("ACCESS_KEY_ID");
00212     auto AccessKey=GetEnv("ACCESS_KEY_SECRET");
00213 
00214     const char *progname=argv[0];
00215 
00216 
00218     sound4::bigvoice::dyn::CDynLib dynlib;
00219     if (!dynlib.Load()) {
00220         std::cerr « "Failed to load processing library" « std::endl;
00221         return 1;
00222     }
00224 
00225     if (argc>1) {
00226         if ((strcasecmp(argv[1],"-h")==0) || (strcasecmp(argv[1],"--help")==0)) {
00227             Usage(progname,dynlib);
00228             return 0;
00229         }
00230     }
00231     if (LoginKey.empty() || RadioName.empty() || KeyId.empty() || AccessKey.empty()) {
00232         std::cerr « "Missing credentials in environment" « std::endl;
00233         Usage(progname,dynlib);
00234         return 1;
00235     }
00236     std::filesystem::path store_path;
00237     if (!store_path_str.empty()) {
00238         std::error_code ec;
00239         store_path = std::filesystem::path(store_path_str);
00240         if (!std::filesystem::exists(store_path,ec) || !std::filesystem::is_directory(store_path,ec))
    {
00241             std::cerr « "Storage folder '"«store_path.c_str()«"' is not a valid directory" «
    std::endl;
00242             Usage(progname,dynlib);
00243             return 1;
00244         }
00245     }
00246 
00248     sound4::bigvoice::dyn::SetLogSeverity(dynlib, verbose5);
00249     sound4::bigvoice::dyn::SetLoggerCallback(dynlib, MyLogger);
00250 
00251     sound4::bigvoice::dyn::CInstance instance(dynlib);
00252     instance.SetParam("ADMIN_USER","admin");
00253     instance.SetParam("ADMIN_SECRET","admin");
00255 
00257     MyPresetManager preset_manager;
00258     if (!store_path.empty()) {
00259         if (!preset_manager.SetupPath(store_path)) {
00260             std::cerr « "Failed to setup the custom preset manager on path " « store_path.c_str() «
    std::endl;
00261             return 1;
00262         }
00263         instance.SetPresetManager(&preset_manager);
00264     }
00266 
00267     std::cerr « "Creating" « std::endl;
00269     if (!instance.Create(LoginKey, RadioName, KeyId, AccessKey, {})) {
00270         std::cerr « "Failed to create the processing instance" « std::endl;
00271         return 1;
00272     }
00274 
00275 #if (BIGVOICE_HAS_WEBSERVER)
00276     std::cerr « "Starting web server" « std::endl;
00278     if (!instance.StartWebServer(listen_port,0)) {
00279         std::cerr « "Failed to create the web server on port " « listen_port « std::endl;
00280         return 1;
00281     }
00283 #endif // (BIGVOICE_HAS_WEBSERVER)
00284 
00285     {
00287         auto client = instance.NewClient();
00288         auto answer = client->ProcessJson(R"JSON({"get":{"processorname":null}})JSON");
00289         std::cerr « "Request for processor name returned "«answer«std::endl;
00291     }
00292 
00293     ProcessLoop(instance);
```

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

11.7 sound4.bigvoice.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.bigvoice.cl.h>
```

Namespaces

- namespace std

Macros

- #define SAMPLE_RATE 48000
- #define CHUNK_SIZE 64
- #define CHANNEL_COUNT 2

Functions

- void ProcessLoop (struct bigvoice_CInstance *pInstance)
- void Usage (const char *progname)
- void MyLogger (bigvoice_LogSeverity severe, const char *str)
- int main (int argc, char *const argv[])

11.7.1 Macro Definition Documentation

11.7.1.1 CHANNEL_COUNT

```
#define CHANNEL_COUNT 2
```

Definition at line 28 of file sound4.bigvoice.cl-pipe.cpp.

11.7.1.2 CHUNK_SIZE

```
#define CHUNK_SIZE 64
```

Definition at line 27 of file sound4.bigvoice.cl-pipe.cpp.

11.7.1.3 SAMPLE_RATE

```
#define SAMPLE_RATE 48000
```

Definition at line 26 of file sound4.bigvoice.cl-pipe.cpp.

11.7.2 Function Documentation

11.7.2.1 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.bigvoice.cl-pipe.cpp.

11.7.2.2 MyLogger()

```
void MyLogger (
            bigvoice_LogSeverity severe,
            const char * str )
```

Definition at line 84 of file sound4.bigvoice.cl-pipe.cpp.

11.7.2.3 ProcessLoop()

```
void ProcessLoop (
            struct bigvoice_CInstance * pInstance )
```

[Processing Loop]

[Processing Loop]

Definition at line 31 of file sound4.bigvoice.cl-pipe.cpp.

11.7.2.4 Usage()

```
void Usage (
            const char * progname )
```

Definition at line 64 of file sound4.bigvoice.cl-pipe.cpp.

11.8 sound4.bigvoice.cl-pipe.cpp

Go to the documentation of this file.

```
00001
00004 #include <unistd.h>
00005 #include <iostream>
00006 #include <strings.h>
00007 #include <sys/types.h>
00008 #include <thread>           // NOTE: needs pthread link
00009 // Choose good filesystem version depending on compiler.
00010 #if __has_include(<filesystem>)
00011     #include <filesystem>
00012 #else //if __cpp_lib_experimental_filesystem
00013     // We need the alias from std::experimental::filesystem to std::filesystem
00014     #include <experimental/filesystem>
00015     namespace std {
00016         namespace filesystem = experimental::filesystem;
00017     }
00018 #endif
00019 #if defined(_WIN32) || defined(WIN32)
00020     #include <io.h>
00021     #include <fcntl.h>
00022     #include <stringapiset.h>   // UTF-8 conversion
00023 #endif
00024 #include <sound4.bigvoice.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 bigvoice_CInstance *pInstance)
00032 {
00034 bool stopthread=false;
00035 alignas(64) float audio_in[CHUNK_SIZE*CHANNEL_COUNT];
00036 alignas(64) float audio_out[CHUNK_SIZE*CHANNEL_COUNT];
00037
00038     #if defined(_WIN32) || defined(WIN32)
00039         // Windows: The stdin, stdout, and stderr streams always open in text mode by default
00040         _setmode(_fileno(stdin), _O_BINARY);
00041         _setmode(_fileno(stdout), _O_BINARY);
00042     #endif
00043
00044     while (!stopthread)
00045     {
00046         // Read a chunk from stdin
00047         if (std::fread(&audio_in[0], sizeof(float), CHUNK_SIZE*CHANNEL_COUNT, stdin) !=
      CHUNK_SIZE*CHANNEL_COUNT) {
00048             std::cerr « "ProcessLoop: input stopped"« std::endl;
00049             stopthread=true;
00050             break;
00051         }
00052         if (stopthread) break;
00053         // Process it
00054         bigvoice_ProcessAudio(pInstance, audio_in, audio_out);
00055         // Write it to stdout
00056         if (fwrite(&audio_out[0], sizeof(float), CHUNK_SIZE*CHANNEL_COUNT, stdout) !=
      CHUNK_SIZE*CHANNEL_COUNT) {
00057             std::cerr « "Failed to write audio output" « std::endl;
00058             stopthread=true;
00059         }
00060     }
00062 }
00063
00064 void Usage(const char *progname)
00065 {
00066     std::cerr « "Usage: " « progname « std::endl;;
00067     std::cerr « "  Reads audio from stdin and return processed audio in stdout" « std::endl;
00068     std::cerr « "Environment:"« std::endl;;
00069     std::cerr « " - S4STORE_PATH            : path to store config [default: ./state]"« std::endl;;
00070     std::cerr « " - S4LOGINKEY              : SOUND4 Login key (or S4USERID)"« std::endl;;
```

```
00071     std::cerr « " - RADIO_NAME              : SOUND4 Radio Name"« std::endl;;
00072     std::cerr « " - ACCESS_KEY_ID           : SOUND4 Access Key ID"« std::endl;;
00073     std::cerr « " - ACCESS_KEY_SECRET       : SOUND4 Access Key Secret"« std::endl;;
00074     std::cerr « std::endl;
00075     std::cerr « "Example:"« std::endl;
00076     std::cerr « "ffmpeg -re -i file.wav -ar "«SAMPLE_RATE«" -ac "«CHANNEL_COUNT«" -f f32le - |
      "«progname«" | ffplay -f f32le -ar "«SAMPLE_RATE«" -ac "«CHANNEL_COUNT«" -i -" « std::endl;;
00077     #if defined(_WIN32) || defined(WIN32)
00078         std::cerr « "ffmpeg -f dshow -i audio=\"Microphone\" -ar "«SAMPLE_RATE«" -ac "«CHANNEL_COUNT«"
      -f f32le - | "«progname«" | ffplay -f f32le -ar "«SAMPLE_RATE«" -ac "«CHANNEL_COUNT«" -i -" «
      std::endl;;
00079     #else
00080         std::cerr « "ffmpeg -f pulse -i default -ar "«SAMPLE_RATE«" -ac "«CHANNEL_COUNT«" -f f32le - |
      "«progname«" | ffplay -f f32le -ar "«SAMPLE_RATE«" -ac "«CHANNEL_COUNT«" -i -" « std::endl;;
00081     #endif
00082 }
00083
00084 void MyLogger(bigvoice_LogSeverity severe [[maybe_unused]], const char *str)
00085 {
00086     std::cerr « str « std::endl;
00087 }
00088
00089
00090 int main(int argc, char *const argv[])
00091 {
00092     struct bigvoice_CInstance *pInstance = NULL;
00093     const char *store_path_str=getenv("S4STORE_PATH");
00094     // Get credentials from environment
00095     char *pLoginKey=getenv("S4LOGINKEY");
00096     if (!pLoginKey) {
00097         pLoginKey=getenv("S4USERID");
00098     }
00099     char *pRadioName=getenv("RADIO_NAME");
00100     char *pKeyId=getenv("ACCESS_KEY_ID");
00101     char *pAccessKey=getenv("ACCESS_KEY_SECRET");
00102
00103     const char *progname=argv[0];
00104
00105     if (argc>1) {
00106         if ((strcasecmp(argv[1],"-h")==0) || (strcasecmp(argv[1],"--help")==0)) {
00107             Usage(progname);
00108             return 0;
00109         }
00110     }
00111     if (!pLoginKey || !pRadioName || !pKeyId || !pAccessKey) {
00112         std::cerr « "Missing credentials in environment" « std::endl;
00113         Usage(progname);
00114         return 1;
00115     }
00116     std::error_code ec;
00117     std::filesystem::path fs_store_path;
00118     if (!store_path_str || store_path_str[0]==0) {
00119         fs_store_path=std::filesystem::current_path(ec) / "state";
00120     } else {
00121         fs_store_path = std::filesystem::path(store_path_str);
00122     }
00123     if (!std::filesystem::exists(fs_store_path,ec) ||
      !std::filesystem::is_directory(fs_store_path,ec)) {
00124         std::cerr « "Storage folder '"«fs_store_path.c_str()«"' is not a valid directory" « std::endl;
00125         Usage(progname);
00126         return 1;
00127     }
00128     std::string store_path;
00129     #ifdef _WIN32
00130     {
00131         // Convert path to UTF-8
00132         std::wstring wstr(fs_store_path.wstring());
00133         if (wstr.empty()) return {};
00134         int size = WideCharToMultiByte(CP_UTF8, 0, &wstr[0], (int)wstr.size(), NULL, 0, NULL, NULL);
00135         std::string str(size, 0);
00136         WideCharToMultiByte(CP_UTF8, 0, &wstr[0], (int)wstr.size(), &str[0], size, NULL, NULL);
00137         store_path=str;
00138     }
00139     #else // !_WIN32
00140         store_path=fs_store_path;
00141     #endif // !_WIN32
00142
00144     // Optional: Use our own logging, instead of default color output to stderr.
00145     //bigvoice_SetLoggerCallback(&MyLogger);
00146
00147     //bigvoice_SetLogSeverity(bigvoice_LogSeverity::info);
00148     bigvoice_SetLogSeverity(bigvoice_LogSeverity::verbose5);
00149
00150     std::cerr « "Process version: " « bigvoice_GetVersion() « std::endl;
00151     // Do some sanity checks
00152     if (bigvoice_GetSampleRate()!=SAMPLE_RATE) {
00153       std::cerr « "Bad library sampling rate, expecting '"«SAMPLE_RATE«"' but has "«
```

```
    bigvoice_GetSampleRate() « std::endl;
00154       return 1;
00155     }
00156     if (bigvoice_GetChunkSizeInFrames()!=CHUNK_SIZE) {
00157       std::cerr « "Bad library frame size, expecting '"«CHUNK_SIZE«"' but has "«
    bigvoice_GetChunkSizeInFrames() « std::endl;
00158       return 1;
00159     }
00160     if (bigvoice_GetChannelCount()!=CHANNEL_COUNT) {
00161       std::cerr « "Bad library channel count, expecting '"«CHANNEL_COUNT«"' but has "«
    bigvoice_GetChannelCount() « std::endl;
00162       return 1;
00163     }
00165
00167     // Setup parameters
00168     struct bigvoice_CParameters *params = bigvoice_NewParameters();
00169     // Optional: activate web authentication, set admin password
00170     // This creates the authentication only for a new storage path
00171     bigvoice_SetParameter(params,"ADMIN_USER", "admin");
00172     bigvoice_SetParameter(params,"ADMIN_SECRET", "admin");
00173
00174     // Create the processing instance
00175     pInstance=bigvoice_InitProcess2(pLoginKey,pRadioName,pKeyId,pAccessKey,store_path.c_str(),
    params);
00176     bigvoice_FreeParameters(params);
00177     if (!pInstance) {
00178       std::cerr « "Failed to create the process instance"« std::endl;
00179       return 1;
00180     }
00181
00182     // Start the update thread, without JSON server (low priority)
00183     std::thread updater([pInstance](){
00184        // will returned when stopped or destroyed
00185        std::cerr « "bigvoice_StartUpdateThread (no json port)" « std::endl;
00186        bigvoice_StartUpdateThread(pInstance, 0);
00187        std::cerr « "bigvoice_StartUpdateThread END" « std::endl;
00188      });
00189
00190     // This is not mandatory, but better be sure we are ready right now
00191     std::cerr « "Waiting update thread" « std::endl;
00192     if (bigvoice_WaitUpdateThreadReady(pInstance, 5000)<0) {
00193         std::cerr « "Timeout to create the process instance"« std::endl;
00194     }
00195     std::cerr « "Waiting update thread DONE" « std::endl;
00196
00197 #if (BIGVOICE_HAS_WEBSERVER)
00199     // Starts a web server, listening on port http://localhost:8080, https://localhost:8443
00200     std::cerr « "Starting web server" « std::endl;
00201     int listenport= 8080;
00202     int listenport_secure= 8443;
00203     uint64_t webserver= SOUND4_INVALID_WEBSERVER_ID;
00204     if (listenport || listenport_secure) {
00205         webserver = bigvoice_Webserver(listenport,listenport_secure,pInstance);
00206         if (webserver==SOUND4_INVALID_WEBSERVER_ID) {
00207             std::cerr « "Starting web server FAILED" « std::endl;
00208         } else {
00209             int web_status=bigvoice_Webserver_Status(webserver);
00210             if (listenport) {
00211                 if (web_status & SOUND4_WEBSERVER_HTTP_OK) {
00212                     std::cerr « "Web server listening on port " « listenport;
00213                 } else {
00214                     std::cerr « "Web server failed to listen on port " « listenport;
00215                 }
00216             }
00217             if (listenport_secure) {
00218                 if (web_status & SOUND4_WEBSERVER_HTTPS_OK) {
00219                     std::cerr « "Web server listening on secure port " « listenport_secure;
00220                 } else {
00221                     std::cerr « "Web server failed to listen on secure port " « listenport_secure;
00222                 }
00223             }
00224             std::cerr « "Starting web server DONE" « std::endl;
00225         }
00226     }
00228 #endif // (BIGVOICE_HAS_WEBSERVER)
00229
00230     // Force load a factory preset now
00231     // Could be used instead of setting a store_path
00232     /*{
00233         bigvoice_CClientInstance *pClient=bigvoice_NewClient(pInstance);
00234         const char *answer=bigvoice_ProcessJson(pClient, "{ \"factoryload\": \"All Programs\" }",
    NULL);
00235         std::cerr « "Loaded factory preset answer: " « answer« std::endl;
00236         bigvoice_FreeJsonAnswer(answer);
00237         bigvoice_DeleteClient(pClient);
00238     }*/
00240
```

```
00241     std::cerr « "Estimated delay " « ( bigvoice_GetEstimatedDelay(pInstance)*1000.0/SAMPLE_RATE ) « "
      ms" « std::endl;
00242
00243     ProcessLoop(pInstance);
00244
00246 #if (BIGVOICE_HAS_WEBSERVER)
00247     // Stopping the webserver
00248     std::cerr « "Stopping web server" « std::endl;
00249     bigvoice_Webserver_Stop(webserver, 1000);
00250     std::cerr « "Stopping web server DONE" « std::endl;
00251 #endif // (BIGVOICE_HAS_WEBSERVER)
00252
00253     // Stopping the processing instance
00254     std::cerr « "Stopping Update thread" « std::endl;
00255     bigvoice_StopUpdateThread(pInstance); // this will exit the update thread
00256     std::cerr « "Stopping Update thread DONE" « std::endl;
00257
00258     updater.join();
00259
00260     bigvoice_ExitProcess(pInstance);
00262
00263     return 0;
00264 }
```

11.9 sound4cl-pipe.cpp File Reference

C++ sample using C++ interface wrapper with dynamic library loading Works with any SOUND4 processor provided.

```
#include <unistd.h>
#include <iostream>
#include <strings.h>
#include <sys/types.h>
#include <filesystem>
#include <fstream>
#include <sstream>
#include <io.h>
#include <fcntl.h>
#include "sound4cl.hpp"
```

Classes

- class MyPresetManager

  [Custom preset manager]

Functions

- std::string GetEnv (const std::string &name)
- void ProcessLoop (sound4::CInstance &instance, sound4::CProcessor &processor)
- int main (int argc, const char **argv)

  [Custom preset manager]

11.9.1 Function Documentation

11.9.1.1 GetEnv()

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

Definition at line 60 of file sound4cl-pipe.cpp.

11.9.1.2 main()

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

[Dynamic loading config]

[Dynamic loading config]

[Dynamic loading]

[Dynamic loading]

[Setup and configure]

[Setup and configure]

[Custom preset manager loading]

[Custom preset manager loading]

[Create the processing instance]

[Create the processing instance]

[Starting a web-server]

[Starting a web-server]

[Communicating with the instance]

[Communicating with the instance]

Definition at line 182 of file sound4cl-pipe.cpp.

11.9.1.3 ProcessLoop()

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

[Processing Loop]

[Processing Loop]

Definition at line 69 of file sound4cl-pipe.cpp.

11.10 sound4cl-pipe.cpp

Go to the documentation of this file.

```
00001
00006 #include <unistd.h>
00007 #include <iostream>
00008 #include <strings.h>
00009 #include <sys/types.h>
00010 #include <filesystem>
00011 #include <fstream>
00012 #include <sstream>
00013 #if defined(_WIN32) || defined(WIN32)
00014     #include <io.h>
00015     #include <fcntl.h>
00016 #endif
00017
00018 #include "sound4cl.hpp"
00019
00020 // ------------------------------------------------------------
00060 std::string GetEnv(const std::string& name)
00061 {
00062     char *data=getenv(name.c_str());
00063     if (!data) return {};
00064     return std::string(data);
00065 }
00066
00067
00068 // Reads from stdin, write to stdout
00069 void ProcessLoop(sound4::CInstance &instance, sound4::CProcessor &processor)
00070 {
00071 bool stopthread=false;
00072
00073     #if defined(_WIN32) || defined(WIN32)
00074         // Windows: The stdin, stdout, and stderr streams always open in text mode by default
00075         _setmode(_fileno(stdin), _O_BINARY);
00076         _setmode(_fileno(stdout), _O_BINARY);
00077     #endif
00078
00080     size_t input_sample_count = instance.GetChunkFrames() * processor.GetChannelCount();
00081     size_t output_sample_count = instance.GetChunkFrames() * processor.GetChannelCount();
00082     while (!stopthread)
00083     {
00084         // Read a chunk from stdin
00085         if (std::fread(instance.GetBufferIn(), sizeof(float), input_sample_count, stdin) !=
      input_sample_count) {
00086             std::cerr « "ProcessLoop: input stopped"« std::endl;
00087             stopthread=true;
00088             break;
00089         }
00090         if (stopthread) break;
00091         // Process it
00092         instance.ProcessAudio();
00093         // Write it to stdout
00094         if (fwrite(instance.GetBufferOut(), sizeof(float), output_sample_count, stdout) !=
      output_sample_count) {
00095             std::cerr « "Failed to write audio output" « std::endl;
00096             stopthread=true;
00097         }
00098     }
00100 }
00101
00103 class MyPresetManager : public sound4::CPresetLoader {
00104 public:
00105     MyPresetManager() = default;
00106     virtual ~MyPresetManager() = default;
00107
00108     bool SetupPath(const std::filesystem::path& a_preset_path)
00109         {
00110             m_preset_path=a_preset_path;
00111             std::error_code ec;
00112             if ( !std::filesystem::exists(m_preset_path,ec ) ) {
00113                 return false;
00114             }
00115             if ( !std::filesystem::is_directory(m_preset_path, ec) ) {
00116                 return false;
00117             }
00118             return true;
00119         }
00120     virtual bool IsReadOnly() override { return false; };
00121     virtual bool Exists(const std::filesystem::path &name) override
00122         {
00123             std::error_code ec;
00124             return std::filesystem::exists(GetRealPath(name), ec);
00125         }
00126     virtual bool Remove(const std::filesystem::path &name) override
```

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

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

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

11.11 sound4.bigvoice.cl.h File Reference

C interface for library.

```
#include <stdint.h>
```

Classes

- struct SOUND4_ProcessInfo

Macros

- #define BIGVOICE_DECL __declspec(dllimport)
- #define BIGVOICE_HAS_WEBSERVER 1
- #define BIGVOICE_HAS_CLOUDBUS 1
- #define BIGVOICE_AUDIOFRAME_COUNT 12
- #define BIGVOICE_CHANNEL_COUNT 1
- #define BIGVOICE_AUDIO_INPUT_COUNT 1
- #define BIGVOICE_AUDIO_OUTPUT_COUNT 1
- #define BIGVOICE_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(* bigvoice_loggerfn) (enum bigvoice_LogSeverity, const char *)
- typedef wchar_t fs_char
- typedef char *(* bigvoice_storage_reader) (const fs_char *filename, void *userdata)
- typedef void(* bigvoice_storage_reader_free) (char *content, void *userdata)
- typedef int(* bigvoice_storage_writer) (const fs_char *filename, const char *content, void *userdata)
- typedef int(* bigvoice_storage_exists) (const fs_char *filename, void *userdata)
- typedef fs_char **(* bigvoice_storage_getall) (void *userdata)
- typedef void(* bigvoice_storage_getall_free) (fs_char **all, void *userdata)
- typedef int(* bigvoice_storage_remove) (const fs_char *filename, void *userdata)
- typedef int(* bigvoice_storage_rename) (const fs_char *from, const fs_char *to, void *userdata)

Enumerations

- enum bigvoice_LogSeverity {
  none = 0 , fatal = 1 , error = 2 , warning = 3 ,
  info = 4 , verbose = 5 , verbose2 = 6 , verbose3 = 7 ,
  verbose4 = 8 , verbose5 = 9 }
- enum bigvoice_PresetChange_Kind { change_kind_created = 1 , change_kind_modified , change_kind_deleted }
- enum bigvoice_SampleFormat {
  INVALID_FORMAT =0 , S16_LE , S16_BE , S24_LE ,
  S24_BE , S32_LE , S32_BE , F32_LE ,
  F32_BE , S16_NATIVE =S16_LE , S24_NATIVE =S24_LE , S32_NATIVE =S32_LE ,
  F32_NATIVE =F32_LE }

Functions

- const struct SOUND4_ProcessInfo * SOUND4_GetProcessInfo ()
- const char * bigvoice_GetVersion ()
- unsigned int bigvoice_GetChunkSizeInFrames ()
- unsigned int * bigvoice_GetPossibleChunkSizeInFrames ()
- unsigned int bigvoice_GetChannelCount ()
- unsigned int bigvoice_GetAudioInputCount ()
- unsigned int bigvoice_GetAudioOutputCount ()
- unsigned int bigvoice_GetSampleRate ()
- void bigvoice_SetLoggerCallback (bigvoice_loggerfn logger)
- void bigvoice_SetLogSeverity (enum bigvoice_LogSeverity severity)
- struct bigvoice_CParameters * bigvoice_NewParameters ()
- void bigvoice_FreeParameters (struct bigvoice_CParameters *params)
- void bigvoice_SetParameter (struct bigvoice_CParameters *params, const char *name, const char *value)
- const char * bigvoice_GetParameter (struct bigvoice_CParameters *params, const char *name)
- void bigvoice_FreeParameterValue (const char *value)
- struct bigvoice_CInstance * bigvoice_InitProcess (const char *LoginKey, const char *RadioName, const char *Access_Key_ID, const char *Access_Key_Secret, const char *save_path)
- struct bigvoice_CInstance * bigvoice_InitProcess2 (const char *LoginKey, const char *RadioName, const char *Access_Key_ID, const char *Access_Key_Secret, const char *save_path, const struct bigvoice_CParameters *parameters)
- struct bigvoice_CInstance * bigvoice_InitProcess3 (const char *LoginKey, const char *RadioName, const char *Access_Key_ID, const char *Access_Key_Secret, const char *save_path, const struct bigvoice_CParameters *parameters, unsigned int frames_per_chunk)
- unsigned int bigvoice_GetProcessChunkFrames (struct bigvoice_CInstance *instance)
- int bigvoice_TerminateProcess (struct bigvoice_CInstance *instance)
- void bigvoice_ExitProcess (struct bigvoice_CInstance *instance)
- struct bigvoice_CBus * bigvoice_NewBus ()
- void bigvoice_FreeBus (struct bigvoice_CBus *bus)
- void bigvoice_SetInstanceBus (struct bigvoice_CParameters *params, struct bigvoice_CBus *bus)
- void bigvoice_SetPresetManager (struct bigvoice_CParameters *params, bigvoice_storage_reader reader, bigvoice_storage_reader_free, bigvoice_storage_writer writer, bigvoice_storage_exists exists, bigvoice_storage_getall getall, bigvoice_storage_getall_free getall_free, bigvoice_storage_remove remove, bigvoice_storage_rename rename, int IsReadOnly, void *userdata)
- void bigvoice_PresetManager_InformChange (struct bigvoice_CInstance *instance, const fs_char *relative↩ _path, enum bigvoice_PresetChange_Kind change_kind)
- void bigvoice_SetMetadata (struct bigvoice_CInstance *instance, const char *key, const char *value)
- void bigvoice_SetMetadataMulti (struct bigvoice_CInstance *instance, const char **keyvalue)
- const char ** bigvoice_GetMetadataInfos (struct bigvoice_CInstance *instance)
- void bigvoice_FreeMetadataInfos (struct bigvoice_CInstance *instance, const char **infos)

- void bigvoice_StartUpdateThread (struct bigvoice_CInstance *instance, unsigned int port)
- void bigvoice_StopUpdateThread (struct bigvoice_CInstance *instance)
- int bigvoice_WaitUpdateThreadReady (struct bigvoice_CInstance *instance, int milliseconds)
- void bigvoice_ProcessAudio (struct bigvoice_CInstance *instance, const float *input, float *output)
- void bigvoice_ProcessAudio_Planar (struct bigvoice_CInstance *instance, float const *const *input, float *const *output)
- float * bigvoice_GetBufferIn (struct bigvoice_CInstance *instance)
- float * bigvoice_GetBufferOut (struct bigvoice_CInstance *instance)
- unsigned int bigvoice_GetEstimatedDelay (struct bigvoice_CInstance *instance)
- const char * bigvoice_GetFormatName (const enum bigvoice_SampleFormat fmt)
- enum bigvoice_SampleFormat bigvoice_GetFormatFromName (const char *name)
- unsigned int bigvoice_GetBytesFromFormat (const enum bigvoice_SampleFormat fmt)
- unsigned int bigvoice_GetMaxPacketFrame (struct bigvoice_CInstance *instance)
- unsigned int bigvoice_AddAudio (struct bigvoice_CInstance *instance, const uint8_t *payload, unsigned int nFrame, enum bigvoice_SampleFormat fmt)
- unsigned int bigvoice_AddPadAudio (struct bigvoice_CInstance *instance)
- unsigned int bigvoice_GetOutputCount (struct bigvoice_CInstance *instance)
- unsigned int bigvoice_GetAudio (struct bigvoice_CInstance *instance, uint8_t *payload, unsigned int max↩ _nFrame, enum bigvoice_SampleFormat fmt)
- void bigvoice_AudioConvertFrom (const uint8_t *payload, float *output, size_t nSpl, enum bigvoice_SampleFormat fmt)
- void bigvoice_AudioConvertTo (const float *input, uint8_t *payload, size_t nSpl, enum bigvoice_SampleFormat fmt)
- void bigvoice_StereoToMono (const float *input, float *output, size_t nFrame)
- void bigvoice_MonoToStereo (const float *input, float *output, size_t nFrame)
- void bigvoice_StereoToMono_Planar (const float *inputL, const float *inputR, float *output, size_t nFrame)
- void bigvoice_MonoToStereo_Planar (const float *input, float *outputL, float *outputR, size_t nFrame)
- void bigvoice_AudioMonoFromLiveStereo (const uint8_t *payload, float *output)
- void bigvoice_AudioMonoToLiveStereo (const float *input, uint8_t *payload)
- struct bigvoice_CClientInstance * bigvoice_NewClient (struct bigvoice_CInstance *instance)
- void bigvoice_DeleteClient (struct bigvoice_CClientInstance *client)
- const char * bigvoice_ProcessJson (struct bigvoice_CClientInstance *client, const char *json_str, int *need_save)
- void bigvoice_FreeJsonAnswer (const char *json_str)
- int bigvoice_SaveState (struct bigvoice_CInstance *instance)
- uint64_t bigvoice_Webserver_tcp (unsigned int listenport, unsigned int listenport_secure, const char *socket_ip, unsigned int socket_port)
- uint64_t bigvoice_Webserver_tcp2 (unsigned int listenport, unsigned int listenport_secure, const char *socket_ip, unsigned int socket_port, const struct bigvoice_CParameters *parameters)
- uint64_t bigvoice_Webserver (unsigned int listenport, unsigned int listenport_secure, struct bigvoice_CInstance *instance)
- int bigvoice_Webserver_Stop (uint64_t id, int timeout_ms)
- int bigvoice_Webserver_Status (uint64_t id)
- void bigvoice_Webserver_SetAppHealth (struct bigvoice_CInstance *instance, int httpcode, const char *contenttype, const char *content)
- void bigvoice_Webserver_GetAppHealth (struct bigvoice_CInstance *instance, int *httpcode, char **contenttype, char **content)
- void bigvoice_Webserver_FreeString (char *str)

11.11.1 Macro Definition Documentation

11.11.1.1 BIGVOICE_DECL

```
#define BIGVOICE_DECL __declspec(dllimport)
```

Definition at line 14 of file sound4.bigvoice.cl.h.

11.11.1.2 BIGVOICE_HAS_CLOUDBUS

```
#define BIGVOICE_HAS_CLOUDBUS 1
```

Definition at line 22 of file sound4.bigvoice.cl.h.

11.11.1.3 BIGVOICE_HAS_WEBSERVER

```
#define BIGVOICE_HAS_WEBSERVER 1
```

Definition at line 21 of file sound4.bigvoice.cl.h.

11.12 sound4.bigvoice.cl.h

Go to the documentation of this file.

```
00001
00004 #pragma once
00005
00006 #include <stdint.h>
00007 #ifdef __unix__
00008     #include <sys/types.h>
00009     #define BIGVOICE_DECL       __attribute__ ((visibility ("default")))
00010 #elif defined(_WIN32) || defined(WIN32)
00011     #ifdef BUILDING_BIGVOICE_DLL
00012         #define BIGVOICE_DECL   __declspec(dllexport)
00013     #else
00014         #define BIGVOICE_DECL   __declspec(dllimport)
00015     #endif
00016 #elif defined(__APPLE__)
00017     #include <sys/types.h>
00018     #define BIGVOICE_DECL       __attribute__ ((visibility ("default")))
00019 #endif
00020
00021 #define BIGVOICE_HAS_WEBSERVER 1
00022 #define BIGVOICE_HAS_CLOUDBUS 1
00023
00024 #ifdef __cplusplus
00025 extern "C" {
00026 #endif // __cplusplus
00027
00073 // ------------------------------------------------------------
00092 // ------------------------------------------------------------
00158 // ------------------------------------------------------------
00159
00160 // ------------------------------------------------------------
00172 #define BIGVOICE_AUDIOFRAME_COUNT      12
00178 #define BIGVOICE_CHANNEL_COUNT          1
00184 #define BIGVOICE_AUDIO_INPUT_COUNT      1
00190 #define BIGVOICE_AUDIO_OUTPUT_COUNT     1
00196 #define BIGVOICE_SAMPLE_RATE            48000
00197
00203 struct SOUND4_ProcessInfo {
00204     size_t struct_size;
00205     const char *name;
00206     const char *prefix;
00207     const char *shortname;
00208     const char *version;
00209 };
00210
00218 BIGVOICE_DECL const struct SOUND4_ProcessInfo* SOUND4_GetProcessInfo();
00219
00225 BIGVOICE_DECL const char *bigvoice_GetVersion();
00226
00239 BIGVOICE_DECL unsigned int bigvoice_GetChunkSizeInFrames();
00240
00249 BIGVOICE_DECL unsigned int* bigvoice_GetPossibleChunkSizeInFrames();
00250
00258 BIGVOICE_DECL unsigned int bigvoice_GetChannelCount();
00259
00268 BIGVOICE_DECL unsigned int bigvoice_GetAudioInputCount();
00269
00278 BIGVOICE_DECL unsigned int bigvoice_GetAudioOutputCount();
00279
```

```
00287 BIGVOICE_DECL unsigned int bigvoice_GetSampleRate();
00288
00292 enum bigvoice_LogSeverity {
00293     none = 0,
00294     fatal = 1,
00295     error = 2,
00296     warning = 3,
00297     info = 4,
00298     verbose = 5,
00299     verbose2 = 6,
00300     verbose3 = 7,
00301     verbose4 = 8,
00302     verbose5 = 9,
00303 };
00304
00311 typedef void (*bigvoice_loggerfn) (enum bigvoice_LogSeverity, const char *);
00312
00322 BIGVOICE_DECL void bigvoice_SetLoggerCallback( bigvoice_loggerfn logger );
00323
00331 BIGVOICE_DECL void bigvoice_SetLogSeverity( enum bigvoice_LogSeverity severity );
00332
00335 // ------------------------------------------------------------------
00341 // ------------------------------------------------------------------
00345 struct bigvoice_CInstance;
00346
00350 struct bigvoice_CParameters;
00351
00359 BIGVOICE_DECL struct bigvoice_CParameters *bigvoice_NewParameters();
00360
00364 BIGVOICE_DECL void bigvoice_FreeParameters(struct bigvoice_CParameters *params);
00365
00377 BIGVOICE_DECL void bigvoice_SetParameter(struct bigvoice_CParameters *params, const char *name, const
      char *value);
00378
00384 BIGVOICE_DECL const char *bigvoice_GetParameter(struct bigvoice_CParameters *params, const char
      *name);
00385
00391 BIGVOICE_DECL void bigvoice_FreeParameterValue(const char *value);
00392
00400 BIGVOICE_DECL struct bigvoice_CInstance *bigvoice_InitProcess(
00401         const char *LoginKey,
00402         const char *RadioName,
00403         const char *Access_Key_ID,
00404         const char *Access_Key_Secret,
00405         const char *save_path
00406     );
00407
00415 BIGVOICE_DECL struct bigvoice_CInstance *bigvoice_InitProcess2(
00416         const char *LoginKey,
00417         const char *RadioName,
00418         const char *Access_Key_ID,
00419         const char *Access_Key_Secret,
00420         const char *save_path,
00421         const struct bigvoice_CParameters *parameters
00422     );
00423
00468 BIGVOICE_DECL struct bigvoice_CInstance *bigvoice_InitProcess3(
00469         const char *LoginKey,
00470         const char *RadioName,
00471         const char *Access_Key_ID,
00472         const char *Access_Key_Secret,
00473         const char *save_path,
00474         const struct bigvoice_CParameters *parameters,
00475         unsigned int frames_per_chunk
00476     );
00477
00489 BIGVOICE_DECL unsigned int bigvoice_GetProcessChunkFrames(struct bigvoice_CInstance *instance);
00490
00500 BIGVOICE_DECL int bigvoice_TerminateProcess(struct bigvoice_CInstance *instance);
00501
00507 BIGVOICE_DECL void bigvoice_ExitProcess(struct bigvoice_CInstance *instance);
00508
00511 #if BIGVOICE_HAS_CLOUDBUS
00512 // ------------------------------------------------------------------
00521 struct bigvoice_CBus;
00522
00528 BIGVOICE_DECL struct bigvoice_CBus* bigvoice_NewBus();
00529
00535 BIGVOICE_DECL void bigvoice_FreeBus(struct bigvoice_CBus *bus);
00536
00546 BIGVOICE_DECL void bigvoice_SetInstanceBus(struct bigvoice_CParameters *params, struct bigvoice_CBus
      *bus);
00547
00548
00550 #endif // BIGVOICE_HAS_CLOUDBUS
00551
00552 // ------------------------------------------------------------------
```

```
00575 // ------------------------------------------------------------
00576
00577 // OS dependent filesystem path char type
00578 #ifdef _WIN32
00579     typedef wchar_t fs_char;
00580 #else // !_WIN32
00581     typedef char fs_char;
00582 #endif // !_WIN32
00583
00591 typedef char * (*bigvoice_storage_reader) (const fs_char *filename, void* userdata);
00592
00599 typedef void (*bigvoice_storage_reader_free) (char *content, void* userdata);
00600
00609 typedef int (*bigvoice_storage_writer) (const fs_char *filename, const char *content, void* userdata);
00610
00618 typedef int (*bigvoice_storage_exists) (const fs_char *filename, void* userdata);
00619
00627 typedef fs_char** (*bigvoice_storage_getall) (void* userdata);
00628
00635 typedef void (*bigvoice_storage_getall_free) (fs_char**all, void* userdata);
00636
00644 typedef int (*bigvoice_storage_remove) (const fs_char *filename, void* userdata);
00645
00654 typedef int (*bigvoice_storage_rename) (const fs_char *from, const fs_char *to, void* userdata);
00655
00673 BIGVOICE_DECL void bigvoice_SetPresetManager(struct bigvoice_CParameters *params
00674     , bigvoice_storage_reader reader, bigvoice_storage_reader_free
00675     , bigvoice_storage_writer writer
00676     , bigvoice_storage_exists exists
00677     , bigvoice_storage_getall getall, bigvoice_storage_getall_free getall_free
00678     , bigvoice_storage_remove remove
00679     , bigvoice_storage_rename rename
00680     , int IsReadOnly
00681     , void* userdata
00682     );
00683
00687 enum bigvoice_PresetChange_Kind {
00688     change_kind_created = 1,
00689     change_kind_modified   ,
00690     change_kind_deleted    ,
00691 };
00692
00701 BIGVOICE_DECL void bigvoice_PresetManager_InformChange(struct bigvoice_CInstance *instance, const
      fs_char *relative_path, enum bigvoice_PresetChange_Kind change_kind);
00702
00705 // ------------------------------------------------------------
00713 // ------------------------------------------------------------
00714
00731 BIGVOICE_DECL void bigvoice_SetMetadata(struct bigvoice_CInstance *instance, const char* key, const
      char* value);
00732
00743 BIGVOICE_DECL void bigvoice_SetMetadataMulti(struct bigvoice_CInstance *instance, const char**
      keyvalue);
00744
00752 BIGVOICE_DECL const char** bigvoice_GetMetadataInfos(struct bigvoice_CInstance *instance);
00753
00760 BIGVOICE_DECL void bigvoice_FreeMetadataInfos(struct bigvoice_CInstance *instance, const char**
      infos);
00761
00764 // ------------------------------------------------------------
00769 // ------------------------------------------------------------
00770
00783 BIGVOICE_DECL void bigvoice_StartUpdateThread(struct bigvoice_CInstance *instance, unsigned int port);
00784
00790 BIGVOICE_DECL void bigvoice_StopUpdateThread(struct bigvoice_CInstance *instance);
00791
00799 BIGVOICE_DECL int bigvoice_WaitUpdateThreadReady(struct bigvoice_CInstance *instance, int
      milliseconds);
00800
00803 // ------------------------------------------------------------
00808 // ------------------------------------------------------------
00830 BIGVOICE_DECL void bigvoice_ProcessAudio(struct bigvoice_CInstance *instance, const float *input,
      float *output );
00831
00850 BIGVOICE_DECL void bigvoice_ProcessAudio_Planar(struct bigvoice_CInstance *instance
00851     , float const * const *input
00852     , float * const *output);
00853
00866 BIGVOICE_DECL float *bigvoice_GetBufferIn(struct bigvoice_CInstance *instance);
00867
00880 BIGVOICE_DECL float *bigvoice_GetBufferOut(struct bigvoice_CInstance *instance);
00881
00889 BIGVOICE_DECL unsigned int bigvoice_GetEstimatedDelay(struct bigvoice_CInstance *instance);
00890
00893 // ------------------------------------------------------------
00900 // ------------------------------------------------------------
00901     #ifdef _MSC_VER
```

```
00902         #define __BYTE_ORDER__  REG_DWORD
00903         #define __ORDER_LITTLE_ENDIAN__ REG_DWORD_LITTLE_ENDIAN
00904         #define __ORDER_BIG_ENDIAN__    REG_DWORD_BIG_ENDIAN
00905     #endif
00906
00910 enum bigvoice_SampleFormat {
00911     INVALID_FORMAT=0,
00912     S16_LE,
00913     S16_BE,
00914     S24_LE,
00915     S24_BE,
00916     S32_LE,
00917     S32_BE,
00918     F32_LE,
00919     F32_BE,
00920
00921     // Native definitions, only links to the physical ones
00922     #if __BYTE_ORDER__ == __ORDER_LITTLE_ENDIAN__
00923         S16_NATIVE=S16_LE,
00924         S24_NATIVE=S24_LE,
00925         S32_NATIVE=S32_LE,
00926         F32_NATIVE=F32_LE,
00927     #else
00928         S16_NATIVE=S16_BE,
00929         S24_NATIVE=S24_BE,
00930         S32_NATIVE=S32_BE,
00931         F32_NATIVE=F32_BE,
00932     #endif
00933 };
00934
00941 BIGVOICE_DECL const char *bigvoice_GetFormatName(const enum bigvoice_SampleFormat fmt);
00942
00949 BIGVOICE_DECL enum bigvoice_SampleFormat bigvoice_GetFormatFromName(const char *name);
00950
00957 BIGVOICE_DECL unsigned int bigvoice_GetBytesFromFormat(const enum bigvoice_SampleFormat fmt);
00958
00959
00966 BIGVOICE_DECL unsigned int bigvoice_GetMaxPacketFrame(struct bigvoice_CInstance *instance);
00967
00978 BIGVOICE_DECL unsigned int bigvoice_AddAudio(struct bigvoice_CInstance *instance, const uint8_t
      *payload, unsigned int nFrame, enum bigvoice_SampleFormat fmt );
00979
00986 BIGVOICE_DECL unsigned int bigvoice_AddPadAudio(struct bigvoice_CInstance *instance);
00987
00994 BIGVOICE_DECL unsigned int bigvoice_GetOutputCount(struct bigvoice_CInstance *instance);
00995
01005 BIGVOICE_DECL unsigned int bigvoice_GetAudio(struct bigvoice_CInstance *instance, uint8_t *payload,
      unsigned int max_nFrame, enum bigvoice_SampleFormat fmt );
01006
01010 // ------------------------------------------------------------
01014 // ------------------------------------------------------------
01025 BIGVOICE_DECL void bigvoice_AudioConvertFrom(const uint8_t *payload, float *output, size_t nSpl, enum
      bigvoice_SampleFormat fmt);
01026
01037 BIGVOICE_DECL void bigvoice_AudioConvertTo(const float *input, uint8_t *payload, size_t nSpl, enum
      bigvoice_SampleFormat fmt);
01038
01048 BIGVOICE_DECL void bigvoice_StereoToMono(const float *input, float *output, size_t nFrame);
01049
01059 BIGVOICE_DECL void bigvoice_MonoToStereo(const float *input, float *output, size_t nFrame);
01060
01071 BIGVOICE_DECL void bigvoice_StereoToMono_Planar(const float *inputL, const float *inputR, float
      *output, size_t nFrame);
01072
01083 BIGVOICE_DECL void bigvoice_MonoToStereo_Planar(const float *input, float *outputL, float *outputR,
      size_t nFrame);
01084
01091 BIGVOICE_DECL void bigvoice_AudioMonoFromLiveStereo(const uint8_t *payload, float *output);
01092
01099 BIGVOICE_DECL void bigvoice_AudioMonoToLiveStereo(const float *input, uint8_t *payload);
01100
01103 // ------------------------------------------------------------
01108 // ------------------------------------------------------------
01112 struct bigvoice_CClientInstance;
01113
01122 BIGVOICE_DECL struct bigvoice_CClientInstance *bigvoice_NewClient(struct bigvoice_CInstance
      *instance);
01123
01129 BIGVOICE_DECL void bigvoice_DeleteClient(struct bigvoice_CClientInstance *client);
01130
01144 BIGVOICE_DECL const char *bigvoice_ProcessJson(struct bigvoice_CClientInstance *client, const char
      *json_str, int *need_save);
01145
01151 BIGVOICE_DECL void bigvoice_FreeJsonAnswer(const char *json_str);
01152
01164 BIGVOICE_DECL int bigvoice_SaveState(struct bigvoice_CInstance *instance);
01165
```

```
01461 #if BIGVOICE_HAS_WEBSERVER
01462 // ------------------------------------------------------------
01475 // ------------------------------------------------------------
01494 BIGVOICE_DECL uint64_t bigvoice_Webserver_tcp(unsigned int listenport, unsigned int listenport_secure,
      const char *socket_ip, unsigned int socket_port);
01495
01496 // ------------------------------------------------------------
01516 BIGVOICE_DECL uint64_t bigvoice_Webserver_tcp2(unsigned int listenport, unsigned int
      listenport_secure, const char *socket_ip, unsigned int socket_port, const struct bigvoice_CParameters
      *parameters);
01517
01535 BIGVOICE_DECL uint64_t bigvoice_Webserver(unsigned int listenport, unsigned int listenport_secure,
      struct bigvoice_CInstance *instance);
01536
01546 BIGVOICE_DECL int bigvoice_Webserver_Stop(uint64_t id, int timeout_ms);
01547
01556 BIGVOICE_DECL int bigvoice_Webserver_Status(uint64_t id);
01557
01568 BIGVOICE_DECL void bigvoice_Webserver_SetAppHealth(struct bigvoice_CInstance *instance, int httpcode,
      const char *contenttype, const char *content);
01569
01578 BIGVOICE_DECL void bigvoice_Webserver_GetAppHealth(struct bigvoice_CInstance *instance, int* httpcode,
      char** contenttype, char** content);
01579
01585 BIGVOICE_DECL void bigvoice_Webserver_FreeString(char *str);
01586
01591 #define SOUND4_INVALID_WEBSERVER_ID            ((uint64_t)-1)
01592
01596 #define SOUND4_WEBSERVER_HTTP_OK               (1<<0)
01600 #define SOUND4_WEBSERVER_HTTPS_OK              (1<<1)
01601
01603 #endif // BIGVOICE_HAS_WEBSERVER
01604
01605 #ifdef __cplusplus
01606 } // extern "C"
01607 #endif // __cplusplus
```

11.13 sound4.bigvoice.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.bigvoice.cl.h"
#include <stringapiset.h>
#include <wchar.h>
```

Classes

- struct sound4::bigvoice::helper::SampleFormat< T >

  Helper for Sample format types.
- struct sound4::bigvoice::helper::SampleFormat< int16_t >
- struct sound4::bigvoice::helper::SampleFormat< int32_t >
- struct sound4::bigvoice::helper::SampleFormat< float >
- class sound4::bigvoice::CBus
- class sound4::bigvoice::CPresetLoader
- class sound4::bigvoice::CInstance
- class sound4::bigvoice::CInstance::CClient

Namespaces

- namespace sound4
- namespace sound4::bigvoice
- namespace sound4::bigvoice::helper

Macros

- #define fs_strdup wcsdup

  helper for OS dependant file name copy

Typedefs

- using sound4::bigvoice::SampleFormat = bigvoice_SampleFormat
- using sound4::bigvoice::LogSeverity = bigvoice_LogSeverity
- using sound4::bigvoice::log_cb_t = std::function< void(LogSeverity, const std::string &)>
- using sound4::bigvoice::PresetChange_Kind = bigvoice_PresetChange_Kind

Variables

- const size_t sound4::bigvoice::ChunkFrames = 12
- const size_t sound4::bigvoice::InputChannels = 1 * 1
- const size_t sound4::bigvoice::OutputChannels = 1 * 1
- const size_t sound4::bigvoice::InputSampleSize =ChunkFrames*InputChannels
- const size_t sound4::bigvoice::OutputSampleSize =ChunkFrames*OutputChannels
- const size_t sound4::bigvoice::SampleRate = 48000

11.13.1 Macro Definition Documentation

11.13.1.1 fs_strdup

```
#define fs_strdup wcsdup
```

Definition at line 25 of file sound4.bigvoice.cl.hpp.

11.14 sound4.bigvoice.cl.hpp

Go to the documentation of this file.

```
00001
00004 #pragma once
00005
00006 #include <string>
00007 #include <exception>
00008 #include <thread>
00009 #include <functional>
00010 #include <memory>
00011 #include <array>
00012 #include <assert.h>
00013 #include <stdexcept>
00014 #include <filesystem>
00015 #include <string.h>
00016
00017 #include "sound4.bigvoice.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 *bigvoice_custom_reader(const fs_char *filename, void* userdata);
00034     static void bigvoice_custom_reader_free(char *content, void* userdata);
00035     static int bigvoice_custom_writer(const fs_char *filename, const char *content, void* userdata);
00036     static int bigvoice_custom_exists(const fs_char *filename, void* userdata);
00037     static fs_char** bigvoice_custom_getall(void* userdata);
00038     static void bigvoice_custom_getall_free(fs_char** all, void* userdata);
00039     static int bigvoice_custom_remove(const fs_char *filename, void* userdata);
00040     static int bigvoice_custom_rename(const fs_char *from, const fs_char *to, void* userdata);
00041 };
00042
00046 namespace sound4 {
00050 namespace bigvoice {
00051     static constexpr const char *process_name = "SOUND4 BIGVOICE.CL";
00052     static constexpr const char *process_shortname = "bigvoice";
00053     using SampleFormat = bigvoice_SampleFormat;
00057     const size_t ChunkFrames=BIGVOICE_AUDIOFRAME_COUNT;
00061     const size_t InputChannels=BIGVOICE_CHANNEL_COUNT*BIGVOICE_AUDIO_INPUT_COUNT;
00065     const size_t OutputChannels=BIGVOICE_CHANNEL_COUNT*BIGVOICE_AUDIO_OUTPUT_COUNT;
00069     const size_t InputSampleSize=ChunkFrames*InputChannels;
00073     const size_t OutputSampleSize=ChunkFrames*OutputChannels;
00077     const size_t SampleRate=BIGVOICE_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 bigvoice_SampleFormat format = S16_NATIVE;
00121         };
00122             template<>
00123         struct SampleFormat<int32_t> {
00124             const bigvoice_SampleFormat format = S32_NATIVE;
00125         };
00126             template<>
00127         struct SampleFormat<float> {
```

```
00128             const bigvoice_SampleFormat format = F32_NATIVE;
00129         };
00130
00131
00136         static inline void AudioConvertFrom(const uint8_t *payload, float *output, size_t nSpl,
      bigvoice_SampleFormat fmt) {
00137             bigvoice_AudioConvertFrom(payload, output, nSpl, fmt);
00138         }
00139         static inline void AudioConvertTo(const float *input, uint8_t *payload, size_t nSpl,
      bigvoice_SampleFormat fmt) {
00140             bigvoice_AudioConvertTo(input, payload, nSpl, fmt);
00141         }
00142         static inline void StereoToMono(const float *input, float *output, size_t nSpl) {
00143             bigvoice_StereoToMono(input, output, nSpl);
00144         }
00145         static inline void MonoToStereo(const float *input, float *output, size_t nSpl) {
00146             bigvoice_MonoToStereo(input, output, nSpl);
00147         }
00148         static inline void StereoToMono_Planar(const float *inputL, const float *inputR, float
      *output, size_t nSpl) {
00149             bigvoice_StereoToMono_Planar(inputL, inputR, output, nSpl);
00150         }
00151         static inline void MonoToStereo_Planar(const float *input, float *outputL, float *outputR,
      size_t nSpl) {
00152             bigvoice_MonoToStereo_Planar(input, outputL, outputR, nSpl);
00153         }
00154         static inline void AudioMonoFromLiveStereo(const uint8_t *payload, float *output) {
00155             bigvoice_AudioMonoFromLiveStereo(payload, output);
00156         }
00157         static inline void AudioMonoToLiveStereo(const float *input, uint8_t *payload) {
00158             bigvoice_AudioMonoToLiveStereo(input, payload);
00159         }
00160
00161     }; // namespace helper
00162
00168     static inline std::string GetVersion()
00169         {
00170             return std::string(bigvoice_GetVersion());
00171         }
00172
00183     static inline unsigned int GetChunkSizeInFrames()
00184         {
00185             return bigvoice_GetChunkSizeInFrames();
00186         }
00187
00196     static inline std::vector<unsigned int> GetPossibleChunkSizeInFrames()
00197         {
00198             std::vector<unsigned int> list;
00199             for (unsigned int* src=bigvoice_GetPossibleChunkSizeInFrames();*src;src++) {
00200                 list.push_back(*src);
00201             }
00202             return list;
00203         }
00204
00212     static inline unsigned int GetChannelCount()
00213         {
00214             return bigvoice_GetChannelCount();
00215         }
00216
00225     static inline unsigned int GetAudioInputCount()
00226         {
00227             return bigvoice_GetAudioInputCount();
00228         }
00229
00238     static inline unsigned int GetAudioOutputCount()
00239         {
00240             return bigvoice_GetAudioOutputCount();
00241         }
00242
00250     static inline unsigned int GetSampleRate()
00251         {
00252             return bigvoice_GetSampleRate();
00253         }
00254
00260     static inline void SanityCheck(bool a_bCheckFrames = true)
00261     {
00262         // Do some sanity checks
00263         if (bigvoice_GetSampleRate()!=BIGVOICE_SAMPLE_RATE) {
00264             throw std::runtime_error("Bad library sampling rate");
00265         }
00266         if (a_bCheckFrames && bigvoice_GetChunkSizeInFrames()!=ChunkFrames) {
00267             throw std::runtime_error("Bad library frame size");
00268         }
00269         if (bigvoice_GetChannelCount()!=BIGVOICE_CHANNEL_COUNT) {
00270             throw std::runtime_error("Bad library channel count");
00271         }
00272         if (bigvoice_GetAudioInputCount()!=BIGVOICE_AUDIO_INPUT_COUNT) {
```

```
00273             throw std::runtime_error("Bad library input count");
00274         }
00275         if (bigvoice_GetAudioOutputCount()!=BIGVOICE_AUDIO_OUTPUT_COUNT) {
00276             throw std::runtime_error("Bad library output count");
00277         }
00278     }
00279
00286     static inline std::string GetFormatName(const SampleFormat fmt)
00287         {
00288             return std::string(bigvoice_GetFormatName(fmt));
00289         }
00290
00297     static inline SampleFormat GetFormatFromName(const std::string& name)
00298         {
00299             return bigvoice_GetFormatFromName(name.c_str());
00300         }
00301
00308     static inline unsigned int GetBytesFromFormat(const SampleFormat fmt)
00309         {
00310             return bigvoice_GetBytesFromFormat(fmt);
00311         }
00312
00316     using LogSeverity = bigvoice_LogSeverity;
00320     using log_cb_t = std::function<void(LogSeverity,const std::string&)>;
00321
00325     static log_cb_t _log_cb;
00326     extern "C" {
00327         static inline void _log_cb_c(bigvoice_LogSeverity severity, const char *c_msg) {
00328             _log_cb(severity, std::string(c_msg));
00329         }
00330     }
00334     static inline void SetLogSeverity(LogSeverity severity)
00335         {
00336             bigvoice_SetLogSeverity(severity);
00337         }
00341     static inline void SetLoggerCallback(log_cb_t cb)
00342         {
00343             _log_cb=cb;
00344             bigvoice_SetLoggerCallback(&_log_cb_c);
00345         }
00346
00347 #if BIGVOICE_HAS_CLOUDBUS
00348     class CBus {
00349         public:
00350             CBus()
00351                 : m_bus(bigvoice_NewBus()) {}
00352             ~CBus()
00353                 {
00354                     bigvoice_FreeBus(m_bus);
00355                     m_bus = nullptr;
00356                 }
00357             bigvoice_CBus *Get() const { return m_bus; }
00358         private:
00359             bigvoice_CBus *m_bus;
00360     };
00361 #endif // BIGVOICE_HAS_CLOUDBUS
00362
00367     class CPresetLoader {
00368     public:
00369         CPresetLoader() = default;
00370         virtual ~CPresetLoader() = default;
00371
00372         virtual bool IsReadOnly() = 0;
00373         virtual bool Exists(const std::filesystem::path &name) = 0;
00374         virtual bool Remove(const std::filesystem::path &name) = 0;
00375         virtual bool Rename(const std::filesystem::path &from, const std::filesystem::path &to) = 0;
00376         virtual std::vector<std::filesystem::path> GetAll() = 0;
00377         virtual std::string Read(const std::filesystem::path &filename) = 0;
00378         virtual bool Write(const std::filesystem::path &filename, const std::string &content) =0;
00379     };
00380     using PresetChange_Kind = bigvoice_PresetChange_Kind;
00381
00385     class CInstance {
00386     public:
00390         CInstance()
00391             {
00392                 params = bigvoice_NewParameters();
00393                 if (!params) throw std::bad_alloc();
00394             }
00395         // non-copyable
00396         CInstance( const CInstance& ) = delete;
00397         CInstance& operator=( const CInstance& ) = delete;
00401         virtual ~CInstance()
00402             {
00403                 Stop();
00404                 if (params) bigvoice_FreeParameters(params);
00405             }
```

```
00409         bool IsOk() const { return instance!=nullptr; }
00417         void SetParam(const std::string& name, const std::string& value)
00418             {
00419                 assert(params);
00420                 bigvoice_SetParameter(params, name.c_str(), value.c_str());
00421             }
00427         std::string GetParam(const std::string& name)
00428             {
00429                 assert(params);
00430                 auto c_value=bigvoice_GetParameter(params, name.c_str());
00431                 std::string ret(c_value);
00432                 bigvoice_FreeParameterValue(c_value);
00433                 return ret;
00434             }
00435
00445         void SetPresetManager(CPresetLoader *preset_manager)
00446             {
00447                 if (!preset_manager) return;
00448                 assert(params);
00449                 bigvoice_SetPresetManager(params,
00450                     bigvoice_custom_reader,
00451                     bigvoice_custom_reader_free,
00452                     bigvoice_custom_writer,
00453                     bigvoice_custom_exists,
00454                     bigvoice_custom_getall,
00455                     bigvoice_custom_getall_free,
00456                     bigvoice_custom_remove,
00457                     bigvoice_custom_rename,
00458                     preset_manager->IsReadOnly(),
00459                     preset_manager
00460                 );
00461             }
00462
00463 #if BIGVOICE_HAS_CLOUDBUS
00467             void SetBus(const CBus& bus) {
00468                 assert(params);
00469                 bigvoice_SetInstanceBus(params,bus.Get());
00470             }
00471 #endif // BIGVOICE_HAS_CLOUDBUS
00472
00487         bool Create(const std::string& LoginKey, const std::string& RadioName, const std::string&
      AWS_Access_ID, const std::string& AWS_Access_Secret, const std::filesystem::path& save_path, int
      json_port = 0, unsigned int frames_per_chunk=12)
00488             {
00489                 assert(!instance);
00490                 std::string l_save_path;
00491                 if (!save_path.empty()) {
00492                     l_save_path = save_path.u8string();
00493                 }
00494                 instance = bigvoice_InitProcess3(LoginKey.c_str(), RadioName.c_str(),
      AWS_Access_ID.c_str(), AWS_Access_Secret.c_str(), l_save_path.c_str(), params, frames_per_chunk);
00495                 if (!instance) return false;
00496                 if (!init_metadata.empty()) {
00497                     for (auto&& [key,value]: init_metadata) {
00498                         bigvoice_SetMetadata(instance, key.c_str(), value.c_str());
00499                     }
00500                     init_metadata.clear();
00501                 }
00502                 update_thread = std::thread([this, json_port](){
00503                     OnUpdateThreadStart();
00504                     bigvoice_StartUpdateThread(instance, json_port);
00505                     OnUpdateThreadStop();
00506                 });
00507                 int timeout=5000;
00508                 #ifdef DEBUG
00509                     timeout=60*1000; // For debugger pauses
00510                 #endif
00511                 if (bigvoice_WaitUpdateThreadReady(instance, timeout)<0) {
00512                     return false;
00513                 }
00514                 return true;
00515             }
00516
00524         unsigned int GetChunkFrames() {
00525             if (instance) {
00526                 return bigvoice_GetProcessChunkFrames(instance);
00527             }
00528             return 0;
00529         }
00530
00538         void PresetManager_InformChange(const std::filesystem::path& relative_path, PresetChange_Kind
      change_kind)
00539             {
00540                 bigvoice_PresetManager_InformChange(instance, relative_path.c_str(),
      (bigvoice_PresetChange_Kind) change_kind);
00541             }
00542
```

```
00554         void SetMetadata(const std::string& key, const char* value)
00555             {
00556                 if (instance) {
00557                     bigvoice_SetMetadata(instance, key.c_str(), value);
00558                 } else {
00559                     init_metadata.push_back( {key, value} );
00560                 }
00561             }
00562 
00572         void SetMetadataMulti(const std::unordered_map<std::string, const char*>& list)
00573             {
00574                 if (instance) {
00575                     typedef const char* pchar;
00576                     pchar* keyvalue=new pchar[2*list.size()+1];
00577                     size_t n=0;
00578                     for (auto&& [key,value]: list) {
00579                         keyvalue[2*n+0]=key.c_str();
00580                         keyvalue[2*n+1]=value;
00581                         n++;
00582                     }
00583                     keyvalue[2*n+0]=nullptr;
00584 
00585                     bigvoice_SetMetadataMulti(instance, keyvalue);
00586 
00587                     delete[] keyvalue;
00588                 } else {
00589                     for (auto&& [key,value]: list) {
00590                         if (value) {
00591                             init_metadata.push_back( {key, value} );
00592                         }
00593                     }
00594                 }
00595             }
00596 
00602         std::vector< std::tuple<std::string,std::string> > GetMetadataInfos()
00603             {
00604                 std::vector< std::tuple<std::string,std::string> > values;
00605                 const char** c_values = bigvoice_GetMetadataInfos(instance);
00606                 if (c_values) {
00607                     for (const char** c_value=c_values; *c_value; ) {
00608                         std::string key(*c_value);
00609                         c_value++;
00610                         if (*c_value) {
00611                             std::string descr(*c_value);
00612                             c_value++;
00613                             values.push_back( {key,descr} );
00614                         }
00615                     }
00616                     bigvoice_FreeMetadataInfos(instance, c_values);
00617                 }
00618                 return values;
00619             }
00620 
00621         // Can be overloaded, called at start of UpdateThread in the thread
00622         virtual void OnUpdateThreadStart() {}
00623         // Can be overloaded to call at end of UpdateThread in the thread
00624         virtual void OnUpdateThreadStop() {}
00625 
00626 #if BIGVOICE_HAS_WEBSERVER
00632         bool StartWebServer(int http_port, int https_port=0)
00633             {
00634                 assert(instance);
00635                 assert(webserver==SOUND4_INVALID_WEBSERVER_ID);
00636                 webserver = bigvoice_Webserver(http_port,https_port,instance);
00637                 if (webserver==SOUND4_INVALID_WEBSERVER_ID) {
00638                     return false;
00639                 }
00640                 int web_status=bigvoice_Webserver_Status(webserver);
00641                 if (http_port && (web_status & SOUND4_WEBSERVER_HTTP_OK) == 0) {
00642                     return false;
00643                 }
00644                 if (https_port && (web_status & SOUND4_WEBSERVER_HTTPS_OK) == 0) {
00645                     return false;
00646                 }
00647                 return true;
00648             }
00654         void StopWebServer()
00655             {
00656                 if (webserver != SOUND4_INVALID_WEBSERVER_ID) {
00657                     bigvoice_Webserver_Stop(webserver, 1000);
00658                     webserver = SOUND4_INVALID_WEBSERVER_ID;
00659                 }
00660             }
00661 
00665         void SetWebServerAppHealth(int httpcode, const std::string& contenttype, const std::string&
    content)
00666             {
```

```
00667                 assert(instance);
00668                 bigvoice_Webserver_SetAppHealth(instance, httpcode, contenttype.c_str(),
    content.c_str());
00669             }
00673         void GetWebServerAppHealth(int &httpcode, std::string& contenttype, std::string& content)
00674             {
00675                 assert(instance);
00676                 char* c_contenttype=nullptr;
00677                 char* c_content=nullptr;
00678                 bigvoice_Webserver_GetAppHealth(instance, &httpcode, &c_contenttype, &c_content);
00679                 contenttype=c_contenttype;
00680                 content=c_content;
00681                 bigvoice_Webserver_FreeString(c_contenttype);
00682                 bigvoice_Webserver_FreeString(c_content);
00683         }
00684 #endif // BIGVOICE_HAS_WEBSERVER
00685
00692         int Terminate()
00693             {
00694                 if (!instance) return 0;
00695                 return bigvoice_TerminateProcess(instance);
00696             }
00697
00703         void Stop()
00704             {
00705                 if (!instance) return;
00706 #if BIGVOICE_HAS_WEBSERVER
00707                 StopWebServer();
00708 #endif // BIGVOICE_HAS_WEBSERVER
00709                 if (update_thread.joinable()) {
00710                     bigvoice_StopUpdateThread(instance); // this will exit the update thread
00711                     update_thread.join();
00712                 }
00713                 bigvoice_ExitProcess(instance);
00714                 instance=nullptr;
00715             }
00716
00722         unsigned int GetEstimatedDelay()
00723             {
00724                 assert(instance);
00725                 return bigvoice_GetEstimatedDelay(instance);
00726             }
00727
00738         std::array<float,InputSampleSize>& GetBufferIn()
00739             {
00740                 assert(instance);
00741                 float *buf=bigvoice_GetBufferIn(instance);
00742                 return reinterpret_cast< std::array<float,InputSampleSize>& >(*buf);
00743
00744             }
00755         std::array<float,OutputSampleSize>& GetBufferOut()
00756             {
00757                 assert(instance);
00758                 float *buf= bigvoice_GetBufferOut(instance);
00759                 return reinterpret_cast< std::array<float,OutputSampleSize>& >(*buf);
00760             }
00766         void ProcessAudio()
00767             {
00768                 assert(instance);
00769                 bigvoice_ProcessAudio(instance, bigvoice_GetBufferIn(instance),
    bigvoice_GetBufferOut(instance));
00770             }
00784         // void ProcessAudio_Planar(const std::array< const std::array<const float,ChunkFrames>,
    InputChannels>& input, std::array< std::array<float,ChunkFrames>, OutputChannels>& output)
00785         void ProcessAudio_Planar(float const * const *input, float * const *output)
00786             {
00787                 assert(instance);
00788                 bigvoice_ProcessAudio_Planar(instance, input, output);
00789             }
00790
00802             template<typename T>
00803         std::vector<T> ProcessAnyAudio(const std::vector<T> input)
00804             {
00805                 assert(instance);
00806                 std::vector<T> output;
00807                 unsigned int out_offset=0;
00808                 unsigned int in_offset=0;
00809                 unsigned int todo = input.size();
00810                 while (todo>0) {
00811                     unsigned int left = AddAudio(&input[in_offset], todo);
00812                     unsigned int out_avail = bigvoice_GetOutputCount(instance);
00813                     output.resize(out_offset + out_avail);
00814                     GetAudio(&output[out_offset], out_avail);
00815                     out_offset+=out_avail;
00816                     in_offset += todo-left;
00817                     todo=left;
00818                 }
```

```
00819                 return output;
00820             }
00821
00827         class CClient {
00828             public:
00829                 CClient(bigvoice_CInstance* instance)
00830                     {
00831                         assert(instance);
00832                         client=bigvoice_NewClient(instance);
00833                         if (!client) throw std::bad_alloc();
00834                     }
00835                 // non-copyable
00836                 CClient( const CInstance& ) = delete;
00837                 CClient& operator=( const CInstance& ) = delete;
00838                 ~CClient()
00839                     {
00840                         if (client) {
00841                             bigvoice_DeleteClient(client);
00842                             client=nullptr;
00843                         }
00844                     }
00852                 std::string ProcessJson(const std::string &request, bool *NeedSave = nullptr)
00853                     {
00854                         assert(client);
00855                         int need_save=0;
00856                         const char *canswer = bigvoice_ProcessJson(client, request.c_str(),
    &need_save);
00857                         if (!canswer) return {};
00858                         std::string answer(canswer);
00859                         bigvoice_FreeJsonAnswer (canswer);
00860                         if (NeedSave) {
00861                             *NeedSave=(need_save!=0);
00862                         }
00863                         return answer;
00864                     }
00865             private:
00866                 bigvoice_CClientInstance *client = nullptr;
00867         };
00875         std::shared_ptr<CClient> NewClient()
00876             {
00877                 assert(instance);
00878                 return std::make_shared<CClient>(instance);
00879             }
00880
00881     protected:
00882             template<typename T>
00883         unsigned int AddAudio(const T* payload, unsigned int nFrame)
00884             {
00885                 assert(instance);
00886                 return bigvoice_AddAudio(instance, reinterpret_cast<const T*>(payload), nFrame,
    helper::SampleFormat<T>::format);
00887             }
00888             template<typename T>
00889         unsigned int GetAudio(T* payload, unsigned int max_nFrame)
00890             {
00891                 assert(instance);
00892                 return bigvoice_GetAudio(instance, reinterpret_cast<T*>(payload), max_nFrame,
    helper::SampleFormat<T>::format);
00893             }
00894
00895     private:
00896         bigvoice_CParameters* params = nullptr;
00897         bigvoice_CInstance* instance = nullptr;
00898         std::thread update_thread;
00899         std::vector< std::pair<std::string,std::string> > init_metadata;
00900 #if BIGVOICE_HAS_WEBSERVER
00901         uint64_t webserver = SOUND4_INVALID_WEBSERVER_ID;
00902 #endif // BIGVOICE_HAS_WEBSERVER
00903     };
00904
00905 }; // namespace bigvoice
00906 }; // namespace sound4
00907
00908 // bridge C callbacks to CPresetLoader
00909 extern "C" {
00910     static char *bigvoice_custom_reader(const fs_char *filename, void* userdata) {
00911         sound4::bigvoice::CPresetLoader
    *preset_loader=reinterpret_cast<sound4::bigvoice::CPresetLoader*>(userdata);
00912         std::string content=preset_loader->Read(filename);
00913         return strdup(content.c_str());
00914     }
00915     static void bigvoice_custom_reader_free(char *content, void* userdata) {
00916         if (!content) return;
00917         free(content);
00918     }
00919     static int bigvoice_custom_writer(const fs_char *filename, const char *content, void* userdata) {
00920         sound4::bigvoice::CPresetLoader
```

```
      *preset_loader=reinterpret_cast<sound4::bigvoice::CPresetLoader*>(userdata);
00921         auto res=preset_loader->Write(filename,content);
00922         return res?0:-1;
00923     }
00924     static int bigvoice_custom_exists(const fs_char *filename, void* userdata) {
00925         sound4::bigvoice::CPresetLoader
      *preset_loader=reinterpret_cast<sound4::bigvoice::CPresetLoader*>(userdata);
00926         auto res=preset_loader->Exists(filename);
00927         return res?0:-1;
00928     }
00929     static fs_char** bigvoice_custom_getall(void* userdata) {
00930         sound4::bigvoice::CPresetLoader
      *preset_loader=reinterpret_cast<sound4::bigvoice::CPresetLoader*>(userdata);
00931         auto res=preset_loader->GetAll();
00932         fs_char**all=(fs_char**)malloc((res.size()+1)*sizeof(const fs_char*));
00933         for (size_t n=0;n<res.size();n++) {
00934             all[n]=fs_strdup(res[n].c_str());
00935         }
00936         all[res.size()]=nullptr;
00937         return all;
00938     }
00939     static void bigvoice_custom_getall_free(fs_char** all, void* userdata) {
00940         if (!all) return;
00941         for (fs_char** one=all;*one;one++) {
00942             free(*one);
00943         }
00944         free(all);
00945     }
00946     static int bigvoice_custom_remove(const fs_char *filename, void* userdata) {
00947         sound4::bigvoice::CPresetLoader
      *preset_loader=reinterpret_cast<sound4::bigvoice::CPresetLoader*>(userdata);
00948         auto res=preset_loader->Remove(filename);
00949         return res?0:-1;
00950     }
00951     static int bigvoice_custom_rename(const fs_char *from, const fs_char *to, void* userdata) {
00952         sound4::bigvoice::CPresetLoader
      *preset_loader=reinterpret_cast<sound4::bigvoice::CPresetLoader*>(userdata);
00953         auto res=preset_loader->Rename(from,to);
00954         return res?0:-1;
00955     }
00956
00957 };
```

11.15 sound4.bigvoice.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.bigvoice.cl.h"
#include <stringapiset.h>
#include <wchar.h>
```

Classes

- class sound4::bigvoice::dyn::helper::DynFuncHolder< T >

  Helper to wrap a function pointer.
- class sound4::bigvoice::dyn::helper::CDynLoader

Helper to load dynamic library.

- class sound4::bigvoice::dyn::CDynLib

  Dynamic library interface.

- struct sound4::bigvoice::dyn::helper::SampleFormat< T >

  Helper for Sample format types.

- struct sound4::bigvoice::dyn::helper::SampleFormat< int16_t >
- struct sound4::bigvoice::dyn::helper::SampleFormat< int32_t >
- struct sound4::bigvoice::dyn::helper::SampleFormat< float >
- class sound4::bigvoice::dyn::CBus
- class sound4::bigvoice::dyn::CPresetLoader
- class sound4::bigvoice::dyn::CInstance
- class sound4::bigvoice::dyn::CInstance::CClient

Namespaces

- namespace sound4
- namespace sound4::bigvoice
- namespace dyn
- namespace sound4::bigvoice::dyn
- namespace sound4::bigvoice::dyn::helper

Macros

- #define UNIXLIKE 0
- #define fs_strdup wcsdup

  helper for OS dependant file name copy

Typedefs

- using sound4::bigvoice::dyn::helper::DynLib_t = HMODULE
- using sound4::bigvoice::dyn::helper::DynFunc_t = FARPROC
- using sound4::bigvoice::dyn::SampleFormat = bigvoice_SampleFormat
- using sound4::bigvoice::dyn::LogSeverity = bigvoice_LogSeverity
- using sound4::bigvoice::dyn::log_cb_t = std::function< void(LogSeverity, const std::string &)>
- using sound4::bigvoice::dyn::PresetChange_Kind = bigvoice_PresetChange_Kind

Variables

- const size_t sound4::bigvoice::dyn::ChunkFrames = 12
- const size_t sound4::bigvoice::dyn::InputChannels = 1 * 1
- const size_t sound4::bigvoice::dyn::OutputChannels = 1 * 1
- const size_t sound4::bigvoice::dyn::InputSampleSize =ChunkFrames*InputChannels
- const size_t sound4::bigvoice::dyn::OutputSampleSize =ChunkFrames*OutputChannels
- const size_t sound4::bigvoice::dyn::SampleRate = 48000

11.15.1 Detailed Description

You need to load the dynamic library, and if it succeed you can use it.

Definition in file sound4.bigvoice.cl_dyn.hpp.

11.15.2 Macro Definition Documentation

11.15.2.1 fs_strdup

```
#define fs_strdup wcsdup
```

Definition at line 42 of file sound4.bigvoice.cl_dyn.hpp.

11.15.2.2 UNIXLIKE

```
#define UNIXLIKE 0
```

Definition at line 23 of file sound4.bigvoice.cl_dyn.hpp.

11.16 sound4.bigvoice.cl_dyn.hpp

Go to the documentation of this file.

```
00001
00007 #pragma once
00008
00009 #include <string>
00010 #include <exception>
00011 #include <thread>
00012 #include <functional>
00013 #include <memory>
00014 #include <array>
00015 #include <assert.h>
00016 #include <stdexcept>
00017 #include <filesystem>
00018 #include <string.h>
00019
00020 #if defined(__unix__) || defined(__APPLE__)
00021     #define UNIXLIKE 1
00022 #else
00023     #define UNIXLIKE 0
00024 #endif
00025
00026 #if UNIXLIKE
00027     #include <dlfcn.h>
00028 #elif defined(_WIN32)
00029     #include <libloaderapi.h>
00030 #else
00031     #error "Unsupported OS"
00032 #endif
00033
00034 // Needed to have function declarations
00035 #include "sound4.bigvoice.cl.h"
00036
00037
00038 #ifdef _WIN32
00039     #include <stringapiset.h>
00040     #include <wchar.h>
00042     #define fs_strdup wcsdup
00043 #else // !_WIN32
00045     #define fs_strdup strdup
00046 #endif // !_WIN32
00047
00048 // bridge C callbacks to CPresetLoader
00049 extern "C" {
00050     static char *bigvoice_custom_reader(const fs_char *filename, void* userdata);
00051     static void bigvoice_custom_reader_free(char *content, void* userdata);
00052     static int bigvoice_custom_writer(const fs_char *filename, const char *content, void* userdata);
00053     static int bigvoice_custom_exists(const fs_char *filename, void* userdata);
00054     static fs_char** bigvoice_custom_getall(void* userdata);
00055     static void bigvoice_custom_getall_free(fs_char** all, void* userdata);
00056     static int bigvoice_custom_remove(const fs_char *filename, void* userdata);
00057     static int bigvoice_custom_rename(const fs_char *from, const fs_char *to, void* userdata);
00058 };
00059
00063 namespace sound4 {
00067 namespace bigvoice {
```

```
00071 namespace dyn {
00072
00076     namespace helper {
00077         // Storage definition, depending on OS
00078         #if UNIXLIKE
00079             using DynLib_t = void *;
00080             using DynFunc_t = void *;
00081         #else
00082             using DynLib_t = HMODULE;
00083             using DynFunc_t = FARPROC;
00084         #endif
00085
00091             template <typename T>
00092         class DynFuncHolder {
00093         private:
00094             DynFunc_t m_ptr = NULL;
00095         public:
00096             DynFuncHolder() = default;
00097             DynFuncHolder(DynFunc_t a_ptr) : m_ptr(a_ptr) {}
00098             operator bool() const { return (m_ptr!=NULL); }
00099             bool IsOk() const { return m_ptr!=NULL; }
00100             operator T* () const { return reinterpret_cast<T*>(m_ptr); }
00101         };
00102
00107         class CDynLoader {
00108             private:
00109                 DynLib_t m_lib {};
00110             public:
00111                 CDynLoader() = default;
00112                 CDynLoader(CDynLoader&&) = default;
00113                 // No copy allowed
00114                 CDynLoader(const CDynLoader&) = delete;
00115                 CDynLoader& operator=(CDynLoader const&) = delete;
00116                 operator bool() const {
00117                     return (m_lib!=NULL);
00118                 }
00119                 bool IsOk() const {
00120                     return (m_lib!=NULL);
00121                 }
00122                 #if UNIXLIKE
00123                     void Close() {
00124                         if (IsOk()) {
00125                             #ifndef __SANITIZE_ADDRESS__ // avoid unloading to keep call stack
00126                             dlclose(m_lib);
00127                             #endif // __SANITIZE_ADDRESS__
00128                             m_lib=NULL;
00129                         }
00130                     }
00131                     bool Load(const std::filesystem::path& dynlib)
00132                     {
00133                         Close();
00134                         m_lib=dlopen(dynlib.c_str(), RTLD_NOW|RTLD_LOCAL);
00135                         return IsOk();
00136                     }
00137                         template <typename T>
00138                     DynFuncHolder<T> GetSymbol(const std::string& name) {
00139                         auto ptr=dlsym(m_lib, name.c_str());
00140                         if (!ptr) {
00141                             throw std::runtime_error("Missing function in library");
00142                         }
00143                         return DynFuncHolder<T>(ptr);
00144                     }
00145                         template <typename T>
00146                     DynFuncHolder<T> GetSymbol_safe(const std::string& name) {
00147                         auto ptr=dlsym(m_lib, name.c_str());
00148                         return DynFuncHolder<T>(ptr);
00149                     }
00150                     static std::filesystem::path GetThisLibraryPath(void) {
00151                         static Dl_info info;
00152                         if (! dladdr((void *) GetThisLibraryPath, & info)) return {};
00153                         auto rp=realpath(info.dli_fname, NULL);
00154                         std::filesystem::path p(rp);
00155                         free(rp);
00156                         return p;
00157                     }
00158                 #else
00159                     bool Load(const std::filesystem::path& dynlib, uint32_t loadflags =
    LOAD_LIBRARY_SEARCH_DLL_LOAD_DIR|LOAD_LIBRARY_SEARCH_APPLICATION_DIR|LOAD_LIBRARY_SEARCH_USER_DIRS|LOAD_LIBRARY_SEARCH_
00160                     {
00161                         Close();
00162                         m_lib=LoadLibraryExW(dynlib.c_str(), NULL, loadflags);
00163                         return IsOk();
00164                     }
00165                     void Close() {
00166                         if (IsOk()) {
00167                             // NOTE: On Windows, FreeLibrary may create a lot of troubles, should use
    FreeLibraryAndExitThread but...
```

```
00168                           // So define SOUND4_CALL_FREELIBRARY before including this file if you
      really want to unload the library dynamically
00169                           #if defined(SOUND4_CALL_FREELIBRARYANDEXITTHREAD)
00170                               FreeLibraryAndExitThread(m_lib, 0);
00171                           #elif defined(SOUND4_CALL_FREELIBRARY)
00172                               FreeLibrary(m_lib);
00173                           #endif // SOUND4_CALL_FREELIBRARY
00174                           m_lib=NULL;
00175                       }
00176                   }
00177                       template <typename T>
00178                   DynFuncHolder<T> GetSymbol(const std::string& name) {
00179                       auto ptr=GetProcAddress(m_lib, name.c_str());
00180                       if (!ptr) {
00181                           throw std::runtime_error("Missing function in library");
00182                       }
00183                       return DynFuncHolder<T>(ptr);
00184                   }
00185                       template <typename T>
00186                   DynFuncHolder<T> GetSymbol_safe(const std::string& name) {
00187                       auto ptr=GetProcAddress(m_lib, name.c_str());
00188                       return DynFuncHolder<T>(ptr);
00189                   }
00190                   static std::filesystem::path GetThisLibraryPath(void) {
00191                       static wchar_t path[MAX_PATH]={};
00192                       HMODULE hm = NULL;
00193                       if (GetModuleHandleExW(GET_MODULE_HANDLE_EX_FLAG_FROM_ADDRESS |
00194                               GET_MODULE_HANDLE_EX_FLAG_UNCHANGED_REFCOUNT,
00195                               (LPCWSTR) &GetThisLibraryPath, &hm))
00196                       {
00197                           if (GetModuleFileNameW(hm, path, MAX_PATH) > 0) {
00198                               return std::filesystem::path(path);
00199                           }
00200                       }
00201                       return {};
00202                   }
00203               #endif
00204               virtual ~CDynLoader() {
00205                   Close();
00206               }
00207       }; // class CDynLoader
00208   }; // namespace helper
00209
00216   class CDynLib {
00217   private:
00218       helper::CDynLoader m_lib;
00219       bool m_bOK = false;
00220   public:
00221       CDynLib() = default;
00222
00223       // All those wrapped functions have the same signature as there C bigvoice_XXX
00224       helper::DynFuncHolder< decltype(bigvoice_GetVersion             ) > GetVersion             ;
00225       helper::DynFuncHolder< decltype(bigvoice_GetChunkSizeInFrames   ) > GetChunkSizeInFrames   ;
00226       helper::DynFuncHolder< decltype(bigvoice_GetChannelCount        ) > GetChannelCount        ;
00227       helper::DynFuncHolder< decltype(bigvoice_GetAudioInputCount     ) > GetAudioInputCount     ;
00228       helper::DynFuncHolder< decltype(bigvoice_GetAudioOutputCount    ) > GetAudioOutputCount    ;
00229       helper::DynFuncHolder< decltype(bigvoice_GetSampleRate          ) > GetSampleRate          ;
00230       helper::DynFuncHolder< decltype(bigvoice_SetLoggerCallback      ) > SetLoggerCallback      ;
00231       helper::DynFuncHolder< decltype(bigvoice_SetLogSeverity         ) > SetLogSeverity         ;
00232       helper::DynFuncHolder< decltype(bigvoice_NewParameters          ) > NewParameters          ;
00233       helper::DynFuncHolder< decltype(bigvoice_FreeParameters         ) > FreeParameters         ;
00234       helper::DynFuncHolder< decltype(bigvoice_SetParameter           ) > SetParameter           ;
00235       helper::DynFuncHolder< decltype(bigvoice_GetParameter           ) > GetParameter           ;
00236       helper::DynFuncHolder< decltype(bigvoice_FreeParameterValue     ) > FreeParameterValue     ;
00237       helper::DynFuncHolder< decltype(bigvoice_InitProcess            ) > InitProcess            ;
00238       helper::DynFuncHolder< decltype(bigvoice_InitProcess2           ) > InitProcess2           ;
00239       helper::DynFuncHolder< decltype(bigvoice_TerminateProcess       ) > TerminateProcess       ;
00240       helper::DynFuncHolder< decltype(bigvoice_ExitProcess            ) > ExitProcess            ;
00241       helper::DynFuncHolder< decltype(bigvoice_StartUpdateThread      ) > StartUpdateThread      ;
00242       helper::DynFuncHolder< decltype(bigvoice_StopUpdateThread       ) > StopUpdateThread       ;
00243       helper::DynFuncHolder< decltype(bigvoice_WaitUpdateThreadReady  ) > WaitUpdateThreadReady  ;
00244       helper::DynFuncHolder< decltype(bigvoice_ProcessAudio           ) > ProcessAudio           ;
00245       helper::DynFuncHolder< decltype(bigvoice_ProcessAudio_Planar    ) > ProcessAudio_Planar    ;
00246       helper::DynFuncHolder< decltype(bigvoice_GetBufferIn            ) > GetBufferIn            ;
00247       helper::DynFuncHolder< decltype(bigvoice_GetBufferOut           ) > GetBufferOut           ;
00248       helper::DynFuncHolder< decltype(bigvoice_GetEstimatedDelay      ) > GetEstimatedDelay      ;
00249       helper::DynFuncHolder< decltype(bigvoice_GetFormatName          ) > GetFormatName          ;
00250       helper::DynFuncHolder< decltype(bigvoice_GetFormatFromName      ) > GetFormatFromName      ;
00251       helper::DynFuncHolder< decltype(bigvoice_GetBytesFromFormat     ) > GetBytesFromFormat     ;
00252       helper::DynFuncHolder< decltype(bigvoice_GetMaxPacketFrame      ) > GetMaxPacketFrame      ;
00253       helper::DynFuncHolder< decltype(bigvoice_AddAudio               ) > AddAudio               ;
00254       helper::DynFuncHolder< decltype(bigvoice_AddPadAudio            ) > AddPadAudio            ;
00255       helper::DynFuncHolder< decltype(bigvoice_GetOutputCount         ) > GetOutputCount         ;
00256       helper::DynFuncHolder< decltype(bigvoice_GetAudio               ) > GetAudio               ;
00257       helper::DynFuncHolder< decltype(bigvoice_AudioConvertFrom       ) > AudioConvertFrom       ;
00258       helper::DynFuncHolder< decltype(bigvoice_AudioConvertTo         ) > AudioConvertTo         ;
00259       helper::DynFuncHolder< decltype(bigvoice_StereoToMono           ) > StereoToMono           ;
```

```
00260         helper::DynFuncHolder< decltype(bigvoice_MonoToStereo           ) > MonoToStereo           ;
00261         helper::DynFuncHolder< decltype(bigvoice_AudioMonoFromLiveStereo) > AudioMonoFromLiveStereo;
00262         helper::DynFuncHolder< decltype(bigvoice_AudioMonoToLiveStereo  ) > AudioMonoToLiveStereo  ;
00263         helper::DynFuncHolder< decltype(bigvoice_NewClient              ) > NewClient              ;
00264         helper::DynFuncHolder< decltype(bigvoice_DeleteClient           ) > DeleteClient           ;
00265         helper::DynFuncHolder< decltype(bigvoice_ProcessJson            ) > ProcessJson            ;
00266         helper::DynFuncHolder< decltype(bigvoice_FreeJsonAnswer         ) > FreeJsonAnswer         ;
00267         helper::DynFuncHolder< decltype(bigvoice_SaveState              ) > SaveState              ;
00268 #if BIGVOICE_HAS_WEBSERVER
00269         helper::DynFuncHolder< decltype(bigvoice_Webserver_tcp          ) > Webserver_tcp          ;
00270         helper::DynFuncHolder< decltype(bigvoice_Webserver_tcp2         ) > Webserver_tcp2         ;
00271         helper::DynFuncHolder< decltype(bigvoice_Webserver              ) > Webserver              ;
00272         helper::DynFuncHolder< decltype(bigvoice_Webserver_Stop         ) > Webserver_Stop         ;
00273         helper::DynFuncHolder< decltype(bigvoice_Webserver_Status       ) > Webserver_Status       ;
00274 #endif // BIGVOICE_HAS_WEBSERVER
00275
00276         // added 2023-03-22
00277         helper::DynFuncHolder< decltype(bigvoice_StereoToMono_Planar    ) > StereoToMono_Planar    ;
00278         helper::DynFuncHolder< decltype(bigvoice_MonoToStereo_Planar    ) > MonoToStereo_Planar    ;
00279
00280         // added 2023-05-22
00281         helper::DynFuncHolder< decltype(bigvoice_SetMetadata            ) > SetMetadata            ;
00282         helper::DynFuncHolder< decltype(bigvoice_GetMetadataInfos       ) > GetMetadataInfos       ;
00283         helper::DynFuncHolder< decltype(bigvoice_FreeMetadataInfos      ) > FreeMetadataInfos      ;
00284
00285         // added 2023-06-20
00286         helper::DynFuncHolder< decltype(bigvoice_SetPresetManager               ) > SetPresetManager
    ;
00287         helper::DynFuncHolder< decltype(bigvoice_PresetManager_InformChange     ) >
    PresetManager_InformChange      ;
00288
00289         // added 2023-09-21
00290         helper::DynFuncHolder< decltype(bigvoice_GetPossibleChunkSizeInFrames   ) >
    GetPossibleChunkSizeInFrames   ;
00291         helper::DynFuncHolder< decltype(bigvoice_GetProcessChunkFrames          ) >
    GetProcessChunkFrames   ;
00292         helper::DynFuncHolder< decltype(bigvoice_InitProcess3                   ) > InitProcess3
    ;
00293
00294 #if BIGVOICE_HAS_CLOUDBUS
00295         // added 2023-06-26
00296         helper::DynFuncHolder< decltype(bigvoice_NewBus                 ) > NewBus      ;
00297         helper::DynFuncHolder< decltype(bigvoice_FreeBus                ) > FreeBus      ;
00298         helper::DynFuncHolder< decltype(bigvoice_SetInstanceBus         ) > SetInstanceBus      ;
00299 #endif // BIGVOICE_HAS_CLOUDBUS
00300
00301         // added 2024-02-19
00302         helper::DynFuncHolder< decltype(bigvoice_SetMetadataMulti       ) > SetMetadataMulti         ;
00303
00304 #if BIGVOICE_HAS_WEBSERVER
00305         // added 2024-05-23
00306         helper::DynFuncHolder< decltype(bigvoice_Webserver_SetAppHealth    ) > Webserver_SetAppHealth
    ;
00307         helper::DynFuncHolder< decltype(bigvoice_Webserver_GetAppHealth    ) > Webserver_GetAppHealth
    ;
00308         helper::DynFuncHolder< decltype(bigvoice_Webserver_FreeString      ) > Webserver_FreeString
    ;
00309 #endif // BIGVOICE_HAS_WEBSERVER
00310
00318         bool Load(const std::filesystem::path& path = {}) {
00319             std::string filename;
00320             #if defined(__APPLE__)
00321                 filename="libsound4.bigvoice.cl.dylib";
00322             #elif defined(__unix__)
00323                 filename="libsound4.bigvoice.cl.so";
00324             #else
00325                 filename="sound4.bigvoice.cl.dll";
00326             #endif
00327             m_lib.Close();
00328             // If a path is given, use it directly and do not try other path
00329             if (!path.empty() && !m_lib.Load(path / filename)) {
00330                 return false;
00331             } else if (!m_lib.IsOk()) {
00332                 auto thisdir=helper::CDynLoader::GetThisLibraryPath().parent_path();
00333                 #ifndef _WIN32
00334                 // Linux: if in a bin directory, try ../lib first
00335                 if (!m_lib.IsOk() && thisdir.filename()=="bin") {
00336                     auto libdir = thisdir.parent_path() / "lib";
00337                     m_lib.Load(libdir / filename);
00338                 }
00339                 #endif // !_WIN32
00340                 if (!m_lib.IsOk()) {
00341                     // Search in the same directory this code is in
00342                     m_lib.Load(thisdir / filename);
00343                 }
00344                 if (!m_lib.IsOk()) {
00345                     // Try current path
```

```
00346                     std::error_code ec;
00347                     auto p = std::filesystem::current_path(ec);
00348                     if (!ec) {
00349                         m_lib.Load(p / filename);
00350                     }
00351                 }
00352                 if (!m_lib.IsOk()) {
00353                     return false;
00354                 }
00355             }
00356             // Load all C functions
00357             try {
00358                 GetVersion              = m_lib.GetSymbol< decltype(bigvoice_GetVersion             )
      >("bigvoice_GetVersion"             );
00359                 GetChunkSizeInFrames    = m_lib.GetSymbol< decltype(bigvoice_GetChunkSizeInFrames   )
      >("bigvoice_GetChunkSizeInFrames"   );
00360                 GetChannelCount         = m_lib.GetSymbol< decltype(bigvoice_GetChannelCount        )
      >("bigvoice_GetChannelCount"        );
00361                 GetAudioInputCount      = m_lib.GetSymbol< decltype(bigvoice_GetAudioInputCount     )
      >("bigvoice_GetAudioInputCount"     );
00362                 GetAudioOutputCount     = m_lib.GetSymbol< decltype(bigvoice_GetAudioOutputCount    )
      >("bigvoice_GetAudioOutputCount"    );
00363                 GetSampleRate           = m_lib.GetSymbol< decltype(bigvoice_GetSampleRate          )
      >("bigvoice_GetSampleRate"          );
00364                 SetLoggerCallback       = m_lib.GetSymbol< decltype(bigvoice_SetLoggerCallback      )
      >("bigvoice_SetLoggerCallback"      );
00365                 SetLogSeverity          = m_lib.GetSymbol< decltype(bigvoice_SetLogSeverity         )
      >("bigvoice_SetLogSeverity"         );
00366                 NewParameters           = m_lib.GetSymbol< decltype(bigvoice_NewParameters          )
      >("bigvoice_NewParameters"          );
00367                 FreeParameters          = m_lib.GetSymbol< decltype(bigvoice_FreeParameters         )
      >("bigvoice_FreeParameters"         );
00368                 SetParameter            = m_lib.GetSymbol< decltype(bigvoice_SetParameter           )
      >("bigvoice_SetParameter"           );
00369                 GetParameter            = m_lib.GetSymbol< decltype(bigvoice_GetParameter           )
      >("bigvoice_GetParameter"           );
00370                 FreeParameterValue      = m_lib.GetSymbol< decltype(bigvoice_FreeParameterValue     )
      >("bigvoice_FreeParameterValue"     );
00371                 InitProcess             = m_lib.GetSymbol< decltype(bigvoice_InitProcess            )
      >("bigvoice_InitProcess"            );
00372                 InitProcess2            = m_lib.GetSymbol< decltype(bigvoice_InitProcess2           )
      >("bigvoice_InitProcess2"           );
00373                 TerminateProcess        = m_lib.GetSymbol< decltype(bigvoice_TerminateProcess       )
      >("bigvoice_TerminateProcess"       );
00374                 ExitProcess             = m_lib.GetSymbol< decltype(bigvoice_ExitProcess            )
      >("bigvoice_ExitProcess"            );
00375                 StartUpdateThread       = m_lib.GetSymbol< decltype(bigvoice_StartUpdateThread      )
      >("bigvoice_StartUpdateThread"      );
00376                 StopUpdateThread        = m_lib.GetSymbol< decltype(bigvoice_StopUpdateThread       )
      >("bigvoice_StopUpdateThread"       );
00377                 WaitUpdateThreadReady   = m_lib.GetSymbol< decltype(bigvoice_WaitUpdateThreadReady  )
      >("bigvoice_WaitUpdateThreadReady"  );
00378                 ProcessAudio            = m_lib.GetSymbol< decltype(bigvoice_ProcessAudio           )
      >("bigvoice_ProcessAudio"           );
00379                 ProcessAudio_Planar     = m_lib.GetSymbol< decltype(bigvoice_ProcessAudio_Planar    )
      >("bigvoice_ProcessAudio_Planar"    );
00380                 GetBufferIn             = m_lib.GetSymbol< decltype(bigvoice_GetBufferIn            )
      >("bigvoice_GetBufferIn"            );
00381                 GetBufferOut            = m_lib.GetSymbol< decltype(bigvoice_GetBufferOut           )
      >("bigvoice_GetBufferOut"           );
00382                 GetEstimatedDelay       = m_lib.GetSymbol< decltype(bigvoice_GetEstimatedDelay      )
      >("bigvoice_GetEstimatedDelay"      );
00383                 GetFormatName           = m_lib.GetSymbol< decltype(bigvoice_GetFormatName          )
      >("bigvoice_GetFormatName"          );
00384                 GetFormatFromName       = m_lib.GetSymbol< decltype(bigvoice_GetFormatFromName      )
      >("bigvoice_GetFormatFromName"      );
00385                 GetBytesFromFormat      = m_lib.GetSymbol< decltype(bigvoice_GetBytesFromFormat     )
      >("bigvoice_GetBytesFromFormat"     );
00386                 GetMaxPacketFrame       = m_lib.GetSymbol< decltype(bigvoice_GetMaxPacketFrame      )
      >("bigvoice_GetMaxPacketFrame"      );
00387                 AddAudio                = m_lib.GetSymbol< decltype(bigvoice_AddAudio               )
      >("bigvoice_AddAudio"               );
00388                 AddPadAudio             = m_lib.GetSymbol< decltype(bigvoice_AddPadAudio            )
      >("bigvoice_AddPadAudio"            );
00389                 GetOutputCount          = m_lib.GetSymbol< decltype(bigvoice_GetOutputCount         )
      >("bigvoice_GetOutputCount"         );
00390                 GetAudio                = m_lib.GetSymbol< decltype(bigvoice_GetAudio               )
      >("bigvoice_GetAudio"               );
00391                 AudioConvertFrom        = m_lib.GetSymbol< decltype(bigvoice_AudioConvertFrom       )
      >("bigvoice_AudioConvertFrom"       );
00392                 AudioConvertTo          = m_lib.GetSymbol< decltype(bigvoice_AudioConvertTo         )
      >("bigvoice_AudioConvertTo"         );
00393                 StereoToMono            = m_lib.GetSymbol< decltype(bigvoice_StereoToMono           )
      >("bigvoice_StereoToMono"           );
00394                 MonoToStereo            = m_lib.GetSymbol< decltype(bigvoice_MonoToStereo           )
      >("bigvoice_MonoToStereo"           );
00395                 AudioMonoFromLiveStereo = m_lib.GetSymbol< decltype(bigvoice_AudioMonoFromLiveStereo)
```

```
>("bigvoice_AudioMonoFromLiveStereo");
00396                 AudioMonoToLiveStereo   = m_lib.GetSymbol< decltype(bigvoice_AudioMonoToLiveStereo  )
>("bigvoice_AudioMonoToLiveStereo"  );
00397                 NewClient               = m_lib.GetSymbol< decltype(bigvoice_NewClient              )
>("bigvoice_NewClient"              );
00398                 DeleteClient            = m_lib.GetSymbol< decltype(bigvoice_DeleteClient           )
>("bigvoice_DeleteClient"           );
00399                 ProcessJson             = m_lib.GetSymbol< decltype(bigvoice_ProcessJson            )
>("bigvoice_ProcessJson"            );
00400                 FreeJsonAnswer          = m_lib.GetSymbol< decltype(bigvoice_FreeJsonAnswer         )
>("bigvoice_FreeJsonAnswer"         );
00401                 SaveState               = m_lib.GetSymbol< decltype(bigvoice_SaveState              )
>("bigvoice_SaveState"              );
00402 #if BIGVOICE_HAS_WEBSERVER
00403                 Webserver_tcp           = m_lib.GetSymbol< decltype(bigvoice_Webserver_tcp          )
>("bigvoice_Webserver_tcp"          );
00404                 Webserver_tcp2          = m_lib.GetSymbol< decltype(bigvoice_Webserver_tcp2         )
>("bigvoice_Webserver_tcp2"         );
00405                 Webserver               = m_lib.GetSymbol< decltype(bigvoice_Webserver              )
>("bigvoice_Webserver"              );
00406                 Webserver_Stop          = m_lib.GetSymbol< decltype(bigvoice_Webserver_Stop         )
>("bigvoice_Webserver_Stop"         );
00407                 Webserver_Status        = m_lib.GetSymbol< decltype(bigvoice_Webserver_Status       )
>("bigvoice_Webserver_Status"       );
00408 #endif // BIGVOICE_HAS_WEBSERVER
00409
00410                 // added 2023-03-22
00411                 StereoToMono_Planar     = m_lib.GetSymbol< decltype(bigvoice_StereoToMono_Planar    )
>("bigvoice_StereoToMono_Planar"    );
00412                 MonoToStereo_Planar     = m_lib.GetSymbol< decltype(bigvoice_MonoToStereo_Planar    )
>("bigvoice_MonoToStereo_Planar"    );
00413
00414             } catch (std::runtime_error& ) {
00415                 return false;
00416             }
00417             // C functions allowed to be missing
00418             try {
00419                 // added 2023-05-22
00420                 SetMetadata             = m_lib.GetSymbol< decltype(bigvoice_SetMetadata             )
>("bigvoice_SetMetadata"            );
00421                 GetMetadataInfos        = m_lib.GetSymbol< decltype(bigvoice_GetMetadataInfos        )
>("bigvoice_GetMetadataInfos"       );
00422                 FreeMetadataInfos       = m_lib.GetSymbol< decltype(bigvoice_FreeMetadataInfos       )
>("bigvoice_FreeMetadataInfos"      );
00423                 // added 2023-06-20
00424                 SetPresetManager            = m_lib.GetSymbol< decltype(bigvoice_SetPresetManager
) >("bigvoice_SetPresetManager"             );
00425                 PresetManager_InformChange  = m_lib.GetSymbol<
decltype(bigvoice_PresetManager_InformChange     ) >("bigvoice_PresetManager_InformChange"    );
00426
00427                 // added 2023-09-21
00428                 GetPossibleChunkSizeInFrames = m_lib.GetSymbol<
decltype(bigvoice_GetPossibleChunkSizeInFrames  ) >("bigvoice_GetPossibleChunkSizeInFrames"  );
00429                 GetProcessChunkFrames        = m_lib.GetSymbol<
decltype(bigvoice_GetProcessChunkFrames         ) >("bigvoice_GetProcessChunkFrames"         );
00430                 InitProcess3                 = m_lib.GetSymbol< decltype(bigvoice_InitProcess3
) >("bigvoice_InitProcess3"                 );
00431
00432 #if BIGVOICE_HAS_CLOUDBUS
00433                 // added 2023-06-26
00434                 NewBus                  = m_lib.GetSymbol< decltype(bigvoice_NewBus                  )
>("bigvoice_NewBus"                 );
00435                 FreeBus                 = m_lib.GetSymbol< decltype(bigvoice_FreeBus                 )
>("bigvoice_FreeBus"                );
00436                 SetInstanceBus          = m_lib.GetSymbol< decltype(bigvoice_SetInstanceBus          )
>("bigvoice_SetInstanceBus"         );
00437 #endif // BIGVOICE_HAS_CLOUDBUS
00438
00439                 // added 2024-02-19
00440                 SetMetadataMulti        = m_lib.GetSymbol< decltype(bigvoice_SetMetadataMulti        )
>("bigvoice_SetMetadataMulti"       );
00441
00442 #if BIGVOICE_HAS_WEBSERVER
00443                 // added 2024-05-23
00444                 Webserver_SetAppHealth      = m_lib.GetSymbol<
decltype(bigvoice_Webserver_SetAppHealth      ) >("Webserver_SetAppHealth"     );
00445                 Webserver_GetAppHealth      = m_lib.GetSymbol<
decltype(bigvoice_Webserver_GetAppHealth      ) >("Webserver_GetAppHealth"     );
00446                 Webserver_FreeString        = m_lib.GetSymbol< decltype(bigvoice_Webserver_FreeString
) >("Webserver_FreeString"       );
00447 #endif // BIGVOICE_HAS_WEBSERVER
00448
00449             }  catch (std::runtime_error& ) {
00450                 // Ignored, handler will take care
00451             }
00452
00453             m_bOK=true;
```

```
00454             return true;
00455         }
00462         bool IsOk() const {
00463             return m_bOK;
00464         }
00465
00466     };
00467
00468
00469
00470     static constexpr const char *process_name = "SOUND4 BIGVOICE.CL";
00471     static constexpr const char *process_shortname = "bigvoice";
00472     using SampleFormat = bigvoice_SampleFormat;
00476     const size_t ChunkFrames=BIGVOICE_AUDIOFRAME_COUNT;
00480     const size_t InputChannels=BIGVOICE_CHANNEL_COUNT*BIGVOICE_AUDIO_INPUT_COUNT;
00484     const size_t OutputChannels=BIGVOICE_CHANNEL_COUNT*BIGVOICE_AUDIO_OUTPUT_COUNT;
00488     const size_t InputSampleSize=ChunkFrames*InputChannels;
00492     const size_t OutputSampleSize=ChunkFrames*OutputChannels;
00496     const size_t SampleRate=BIGVOICE_SAMPLE_RATE;
00497
00501     namespace helper {
00502         #ifdef _WIN32
00508         static inline std::string WStringToUTF8(const std::wstring& wstr) {
00509             if (wstr.empty()) return {};
00510             int size = WideCharToMultiByte(CP_UTF8, 0, &wstr[0], (int)wstr.size(), NULL, 0, NULL,
      NULL);
00511             std::string str(size, 0);
00512             WideCharToMultiByte(CP_UTF8, 0, &wstr[0], (int)wstr.size(), &str[0], size, NULL, NULL);
00513             return str;
00514         }
00520         static inline std::wstring UTF8ToWString(const std::string& str) {
00521             if (str.empty()) return std::wstring();
00522             int size = MultiByteToWideChar(CP_UTF8, 0, &str[0], (int)str.size(), NULL, 0);
00523             std::wstring wstr(size, 0);
00524             MultiByteToWideChar(CP_UTF8, 0, &str[0], (int)str.size(), &wstr[0], size);
00525             return wstr;
00526         }
00527         #endif // _WIN32
00528
00534             template<typename T>
00535         struct SampleFormat {
00536         };
00537             template<>
00538         struct SampleFormat<int16_t> {
00539             const bigvoice_SampleFormat format = S16_NATIVE;
00540         };
00541             template<>
00542         struct SampleFormat<int32_t> {
00543             const bigvoice_SampleFormat format = S32_NATIVE;
00544         };
00545             template<>
00546         struct SampleFormat<float> {
00547             const bigvoice_SampleFormat format = F32_NATIVE;
00548         };
00549
00550
00555         static inline void AudioConvertFrom(CDynLib& dynlib, const uint8_t *payload, float *output,
      size_t nSpl, bigvoice_SampleFormat fmt) {
00556             dynlib.AudioConvertFrom(payload, output, nSpl, fmt);
00557         }
00558         static inline void AudioConvertTo(CDynLib& dynlib, const float *input, uint8_t *payload,
      size_t nSpl, bigvoice_SampleFormat fmt) {
00559             dynlib.AudioConvertTo(input, payload, nSpl, fmt);
00560         }
00561         static inline void StereoToMono(CDynLib& dynlib, const float *input, float *output, size_t
      nSpl) {
00562             dynlib.StereoToMono(input, output, nSpl);
00563         }
00564         static inline void MonoToStereo(CDynLib& dynlib, const float *input, float *output, size_t
      nSpl) {
00565             dynlib.MonoToStereo(input, output, nSpl);
00566         }
00567         static inline void StereoToMono_Planar(CDynLib& dynlib, const float *inputL, const float
      *inputR, float *output, size_t nSpl) {
00568             dynlib.StereoToMono_Planar(inputL, inputR, output, nSpl);
00569         }
00570         static inline void MonoToStereo_Planar(CDynLib& dynlib, const float *input, float *outputL,
      float *outputR, size_t nSpl) {
00571             dynlib.MonoToStereo_Planar(input, outputL, outputR, nSpl);
00572         }
00573         static inline void AudioMonoFromLiveStereo(CDynLib& dynlib, const uint8_t *payload, float
      *output) {
00574             dynlib.AudioMonoFromLiveStereo(payload, output);
00575         }
00576         static inline void AudioMonoToLiveStereo(CDynLib& dynlib, const float *input, uint8_t
      *payload) {
00577             dynlib.AudioMonoToLiveStereo(input, payload);
```

```
00578         }
00579
00580     }; // namespace helper
00581
00587     static inline std::string GetVersion(CDynLib& dynlib)
00588         {
00589             return std::string(dynlib.GetVersion());
00590         }
00591
00602     static inline unsigned int GetChunkSizeInFrames(CDynLib& dynlib)
00603         {
00604             return dynlib.GetChunkSizeInFrames();
00605         }
00606
00615     static inline std::vector<unsigned int> GetPossibleChunkSizeInFrames(CDynLib& dynlib)
00616         {
00617             std::vector<unsigned int> list;
00618             if (dynlib.GetPossibleChunkSizeInFrames) {
00619                 for (unsigned int* src=dynlib.GetPossibleChunkSizeInFrames();*src;src++) {
00620                     list.push_back(*src);
00621                 }
00622             } else {
00623                 list.push_back(12);
00624             }
00625             return list;
00626         }
00627
00635     static inline unsigned int GetChannelCount(CDynLib& dynlib)
00636         {
00637             return dynlib.GetChannelCount();
00638         }
00639
00648     static inline unsigned int GetAudioInputCount(CDynLib& dynlib)
00649         {
00650             return dynlib.GetAudioInputCount();
00651         }
00652
00661     static inline unsigned int GetAudioOutputCount(CDynLib& dynlib)
00662         {
00663             return dynlib.GetAudioOutputCount();
00664         }
00665
00673     static inline unsigned int GetSampleRate(CDynLib& dynlib)
00674         {
00675             return dynlib.GetSampleRate();
00676         }
00677
00683     static inline void SanityCheck(CDynLib& dynlib, bool a_bCheckFrames = true)
00684     {
00685         // Do some sanity checks
00686         if (dynlib.GetSampleRate()!=BIGVOICE_SAMPLE_RATE) {
00687             throw std::runtime_error("Bad library sampling rate");
00688         }
00689         if (a_bCheckFrames && dynlib.GetChunkSizeInFrames()!=ChunkFrames) {
00690             throw std::runtime_error("Bad library frame size");
00691         }
00692         if (dynlib.GetChannelCount()!=BIGVOICE_CHANNEL_COUNT) {
00693             throw std::runtime_error("Bad library channel count");
00694         }
00695         if (dynlib.GetAudioInputCount()!=BIGVOICE_AUDIO_INPUT_COUNT) {
00696             throw std::runtime_error("Bad library input count");
00697         }
00698         if (dynlib.GetAudioOutputCount()!=BIGVOICE_AUDIO_OUTPUT_COUNT) {
00699             throw std::runtime_error("Bad library output count");
00700         }
00701     }
00702
00709     static inline std::string GetFormatName(CDynLib& dynlib, const SampleFormat fmt)
00710         {
00711             return std::string(dynlib.GetFormatName(fmt));
00712         }
00713
00720     static inline SampleFormat GetFormatFromName(CDynLib& dynlib, const std::string& name)
00721         {
00722             return dynlib.GetFormatFromName(name.c_str());
00723         }
00724
00731     static inline unsigned int GetBytesFromFormat(CDynLib& dynlib, const SampleFormat fmt)
00732         {
00733             return dynlib.GetBytesFromFormat(fmt);
00734         }
00735
00739     using LogSeverity = bigvoice_LogSeverity;
00743     using log_cb_t = std::function<void(LogSeverity,const std::string&)>;
00744
00748     static log_cb_t _log_cb;
00749     extern "C" {
```

```
00750         static inline void _log_cb_c(bigvoice_LogSeverity severity, const char *c_msg) {
00751             _log_cb(severity, std::string(c_msg));
00752         }
00753     }
00757     static inline void SetLogSeverity(CDynLib& dynlib, LogSeverity severity)
00758         {
00759             dynlib.SetLogSeverity(severity);
00760         }
00764     static inline void SetLoggerCallback(CDynLib& dynlib, log_cb_t cb)
00765         {
00766             _log_cb=cb;
00767             dynlib.SetLoggerCallback(&_log_cb_c);
00768         }
00769
00770 #if BIGVOICE_HAS_CLOUDBUS
00771     class CBus {
00772         public:
00773             CBus(CDynLib& dynlib)
00774                 : m_dynlib(dynlib)
00775                 {
00776                     if (m_dynlib.NewBus) {
00777                         m_bus=m_dynlib.NewBus();
00778                     }
00779                 }
00780             ~CBus()
00781                 {
00782                     if (m_bus && m_dynlib.FreeBus) {
00783                         m_dynlib.FreeBus(m_bus);
00784                     }
00785                     m_bus = nullptr;
00786                 }
00787             bigvoice_CBus *Get() const { return m_bus; }
00788         private:
00789             CDynLib& m_dynlib;
00790             bigvoice_CBus *m_bus;
00791     };
00792 #endif // BIGVOICE_HAS_CLOUDBUS
00793
00798     class CPresetLoader {
00799     public:
00800         CPresetLoader() = default;
00801         virtual ~CPresetLoader() = default;
00802
00803         virtual bool IsReadOnly() = 0;
00804         virtual bool Exists(const std::filesystem::path &name) = 0;
00805         virtual bool Remove(const std::filesystem::path &name) = 0;
00806         virtual bool Rename(const std::filesystem::path &from, const std::filesystem::path &to) = 0;
00807         virtual std::vector<std::filesystem::path> GetAll() = 0;
00808         virtual std::string Read(const std::filesystem::path &filename) = 0;
00809         virtual bool Write(const std::filesystem::path &filename, const std::string &content) =0;
00810     };
00811     using PresetChange_Kind = bigvoice_PresetChange_Kind;
00812
00816     class CInstance {
00817     public:
00821         CInstance(CDynLib& dynlib)
00822         : m_dynlib(dynlib)
00823             {
00824                 params = m_dynlib.NewParameters();
00825                 if (!params) throw std::bad_alloc();
00826             }
00827         // non-copyable
00828         CInstance( const CInstance& ) = delete;
00829         CInstance& operator=( const CInstance& ) = delete;
00833         virtual ~CInstance()
00834             {
00835                 Stop();
00836                 if (params) m_dynlib.FreeParameters(params);
00837             }
00841        bool IsOk() const { return instance!=nullptr; }
00849         void SetParam(const std::string& name, const std::string& value)
00850             {
00851                 assert(params);
00852                 m_dynlib.SetParameter(params, name.c_str(), value.c_str());
00853             }
00859         std::string GetParam(const std::string& name)
00860             {
00861                 assert(params);
00862                 auto c_value=m_dynlib.GetParameter(params, name.c_str());
00863                 std::string ret(c_value);
00864                 m_dynlib.FreeParameterValue(c_value);
00865                 return ret;
00866             }
00867 #if BIGVOICE_HAS_CLOUDBUS
00871             void SetBus(const CBus& bus) {
00872                 assert(params);
00873                 if (m_dynlib.SetInstanceBus) {
```

```
00874                     m_dynlib.SetInstanceBus(params,bus.Get());
00875                 }
00876             }
00877 #endif // BIGVOICE_HAS_CLOUDBUS
00878
00888         void SetPresetManager(CPresetLoader *preset_manager)
00889             {
00890                 if (!preset_manager) return;
00891                 if (!m_dynlib.SetPresetManager) return;
00892                 assert(params);
00893                 m_dynlib.SetPresetManager(params,
00894                     bigvoice_custom_reader,
00895                     bigvoice_custom_reader_free,
00896                     bigvoice_custom_writer,
00897                     bigvoice_custom_exists,
00898                     bigvoice_custom_getall,
00899                     bigvoice_custom_getall_free,
00900                     bigvoice_custom_remove,
00901                     bigvoice_custom_rename,
00902                     preset_manager->IsReadOnly(),
00903                     preset_manager
00904                 );
00905             }
00906
00921         bool Create(const std::string& LoginKey, const std::string& RadioName, const std::string&
      Access_Key_ID, const std::string& Access_Key_Secret, const std::filesystem::path& save_path, int
      json_port = 0, unsigned int frames_per_chunk=12)
00922             {
00923                 assert(!instance);
00924                 std::string l_save_path;
00925                 if (!save_path.empty()) {
00926                     l_save_path = save_path.u8string();
00927                 }
00928                 if (m_dynlib.InitProcess3) {
00929                     instance = m_dynlib.InitProcess3(LoginKey.c_str(), RadioName.c_str(),
      Access_Key_ID.c_str(), Access_Key_Secret.c_str(), l_save_path.c_str(), params, frames_per_chunk);
00930                 } else if (frames_per_chunk!=12) {
00931                     return false;
00932                 } else {
00933                     instance = m_dynlib.InitProcess2(LoginKey.c_str(), RadioName.c_str(),
      Access_Key_ID.c_str(), Access_Key_Secret.c_str(), l_save_path.c_str(), params);
00934                 }
00935                 if (!instance) return false;
00936                 if (!init_metadata.empty()) {
00937                     for (auto&& [key,value]: init_metadata) {
00938                         m_dynlib.SetMetadata(instance, key.c_str(), value.c_str());
00939                     }
00940                     init_metadata.clear();
00941                 }
00942                 update_thread = std::thread([this, json_port](){
00943                     OnUpdateThreadStart();
00944                     m_dynlib.StartUpdateThread(instance, json_port);
00945                     OnUpdateThreadStop();
00946                 });
00947                 int timeout=5000;
00948                 #ifdef DEBUG
00949                     timeout=60*1000; // For debugger pauses
00950                 #endif
00951                 if (m_dynlib.WaitUpdateThreadReady(instance, timeout)<0) {
00952                     return false;
00953                 }
00954                 return true;
00955             }
00956
00964         unsigned int GetChunkFrames() {
00965             if (instance && m_dynlib.GetProcessChunkFrames) {
00966                 return m_dynlib.GetProcessChunkFrames(instance);
00967             }
00968             return 12;
00969         }
00970
00978         void PresetManager_InformChange(const std::filesystem::path& relative_path, PresetChange_Kind
      change_kind)
00979             {
00980                 if (m_dynlib.PresetManager_InformChange) {
00981                     m_dynlib.PresetManager_InformChange(instance, relative_path.c_str(),
      (bigvoice_PresetChange_Kind) change_kind);
00982                 }
00983             }
00984
00996         void SetMetadata(const std::string& key, const char* value)
00997             {
00998                 if (m_dynlib.SetMetadata) {
00999                     if (instance) {
01000                         m_dynlib.SetMetadata(instance, key.c_str(), value);
01001                     } else {
01002                         init_metadata.push_back( {key, value} );
```

```
01003                     }
01004                 }
01005             }
01006
01007
01017         void SetMetadataMulti(const std::unordered_map<std::string, const char*>& list)
01018             {
01019                 if (instance) {
01020                     if (m_dynlib.SetMetadataMulti) {
01021                         typedef const char* pchar;
01022                         pchar* keyvalue=new pchar[2*list.size()+1];
01023                         size_t n=0;
01024                         for (auto&& [key,value]: list) {
01025                             keyvalue[2*n+0]=key.c_str();
01026                             keyvalue[2*n+1]=value;
01027                             n++;
01028                         }
01029                         keyvalue[2*n+0]=nullptr;
01030
01031                         bigvoice_SetMetadataMulti(instance, keyvalue);
01032
01033                         delete[] keyvalue;
01034                     } else {
01035                         for (auto&& [key,value]: list) {
01036                             SetMetadata(key, value);
01037                         }
01038                     }
01039                 } else {
01040                     for (auto&& [key,value]: list) {
01041                         if (value) {
01042                             init_metadata.push_back( {key, value} );
01043                         }
01044                     }
01045                 }
01046             }
01047
01053         std::vector< std::tuple<std::string,std::string> > GetMetadataInfos()
01054             {
01055                 std::vector< std::tuple<std::string,std::string> > values;
01056                 if (m_dynlib.GetMetadataInfos) {
01057                     const char** c_values = m_dynlib.GetMetadataInfos(instance);
01058                     if (c_values) {
01059                         for (const char** c_value=c_values; *c_value; ) {
01060                             std::string key(*c_value);
01061                             c_value++;
01062                             if (*c_value) {
01063                                 std::string descr(*c_value);
01064                                 c_value++;
01065                                 values.push_back( {key,descr} );
01066                             }
01067                         }
01068                         m_dynlib.FreeMetadataInfos(instance, c_values);
01069                     }
01070                 }
01071                 return values;
01072             }
01073
01074         // Can be overloaded, called at start of UpdateThread in the thread
01075         virtual void OnUpdateThreadStart() {}
01076         // Can be overloaded to call at end of UpdateThread in the thread
01077         virtual void OnUpdateThreadStop() {}
01078
01079 #if BIGVOICE_HAS_WEBSERVER
01085         bool StartWebServer(int http_port, int https_port=0)
01086             {
01087                 assert(instance);
01088                 assert(webserver==SOUND4_INVALID_WEBSERVER_ID);
01089                 webserver = m_dynlib.Webserver(http_port,https_port,instance);
01090                 if (webserver==SOUND4_INVALID_WEBSERVER_ID) {
01091                     return false;
01092                 }
01093                 int web_status=m_dynlib.Webserver_Status(webserver);
01094                 if (http_port && (web_status & SOUND4_WEBSERVER_HTTP_OK) == 0) {
01095                     return false;
01096                 }
01097                 if (https_port && (web_status & SOUND4_WEBSERVER_HTTPS_OK) == 0) {
01098                     return false;
01099                 }
01100                 return true;
01101             }
01107         void StopWebServer()
01108             {
01109                 if (webserver != SOUND4_INVALID_WEBSERVER_ID) {
01110                     m_dynlib.Webserver_Stop(webserver, 1000);
01111                     webserver = SOUND4_INVALID_WEBSERVER_ID;
01112                 }
01113             }
```

```
01114
01118         void SetWebServerAppHealth(int httpcode, const std::string& contenttype, const std::string&
      content)
01119             {
01120                 assert(instance);
01121                 if (!m_dynlib.Webserver_SetAppHealth) return;
01122                 m_dynlib.Webserver_SetAppHealth(instance, httpcode, contenttype.c_str(),
      content.c_str());
01123             }
01127         void GetWebServerAppHealth(int &httpcode, std::string& contenttype, std::string& content)
01128             {
01129                 assert(instance);
01130                 if (!m_dynlib.Webserver_GetAppHealth) return;
01131                 char* c_contenttype=nullptr;
01132                 char* c_content=nullptr;
01133                 m_dynlib.Webserver_GetAppHealth(instance, &httpcode, &c_contenttype, &c_content);
01134                 contenttype=c_contenttype;
01135                 content=c_content;
01136                 m_dynlib.Webserver_FreeString(c_contenttype);
01137                 m_dynlib.Webserver_FreeString(c_content);
01138             }
01139 #endif // BIGVOICE_HAS_WEBSERVER
01140
01147         int Terminate()
01148             {
01149                 if (!instance) return 0;
01150                 return m_dynlib.TerminateProcess(instance);
01151             }
01152
01158         void Stop()
01159             {
01160                 if (!instance) return;
01161 #if BIGVOICE_HAS_WEBSERVER
01162                 StopWebServer();
01163 #endif // BIGVOICE_HAS_WEBSERVER
01164                 if (update_thread.joinable()) {
01165                     m_dynlib.StopUpdateThread(instance); // this will exit the update thread
01166                     update_thread.join();
01167                 }
01168                 m_dynlib.ExitProcess(instance);
01169                 instance=nullptr;
01170             }
01171
01177         unsigned int GetEstimatedDelay()
01178             {
01179                 assert(instance);
01180                 return m_dynlib.GetEstimatedDelay(instance);
01181             }
01182
01193         std::array<float,InputSampleSize>& GetBufferIn()
01194             {
01195                 assert(instance);
01196                 float *buf=m_dynlib.GetBufferIn(instance);
01197                 return reinterpret_cast< std::array<float,InputSampleSize>& >(*buf);
01198
01199             }
01210         std::array<float,OutputSampleSize>& GetBufferOut()
01211             {
01212                 assert(instance);
01213                 float *buf= m_dynlib.GetBufferOut(instance);
01214                 return reinterpret_cast< std::array<float,OutputSampleSize>& >(*buf);
01215             }
01221         void ProcessAudio()
01222             {
01223                 assert(instance);
01224                 m_dynlib.ProcessAudio(instance, m_dynlib.GetBufferIn(instance),
      m_dynlib.GetBufferOut(instance));
01225             }
01239         // void ProcessAudio_Planar(const std::array< const std::array<const float,ChunkFrames>,
      InputChannels>& input, std::array< std::array<float,ChunkFrames>, OutputChannels>& output)
01240         void ProcessAudio_Planar(float const * const *input, float * const *output)
01241             {
01242                 assert(instance);
01243                 m_dynlib.ProcessAudio_Planar(instance, input, output);
01244             }
01245
01257             template<typename T>
01258         std::vector<T> ProcessAnyAudio(const std::vector<T> input)
01259             {
01260                 assert(instance);
01261                 std::vector<T> output;
01262                 unsigned int out_offset=0;
01263                 unsigned int in_offset=0;
01264                 unsigned int todo = input.size();
01265                 while (todo>0) {
01266                     unsigned int left = AddAudio(&input[in_offset], todo);
01267                     unsigned int out_avail = m_dynlib.GetOutputCount(instance);
```

```
01268                     output.resize(out_offset + out_avail);
01269                     GetAudio(&output[out_offset], out_avail);
01270                     out_offset+=out_avail;
01271                     in_offset += todo-left;
01272                     todo=left;
01273                 }
01274                 return output;
01275             }
01276 
01282         class CClient {
01283             public:
01284                 CClient(bigvoice_CInstance* instance, CDynLib& dynlib)
01285                 : m_dynlib(dynlib)
01286                     {
01287                         assert(instance);
01288                         client=m_dynlib.NewClient(instance);
01289                         if (!client) throw std::bad_alloc();
01290                     }
01291                 // non-copyable
01292                 CClient( const CInstance& ) = delete;
01293                 CClient& operator=( const CInstance& ) = delete;
01294                 ~CClient()
01295                     {
01296                         if (client) {
01297                             m_dynlib.DeleteClient(client);
01298                             client=nullptr;
01299                         }
01300                     }
01308                 std::string ProcessJson(const std::string &request, bool *NeedSave = nullptr)
01309                     {
01310                         assert(client);
01311                         int need_save=0;
01312                         const char *canswer = m_dynlib.ProcessJson(client, request.c_str(),
    &need_save);
01313                         if (!canswer) return {};
01314                         std::string answer(canswer);
01315                         m_dynlib.FreeJsonAnswer (canswer);
01316                         if (NeedSave) {
01317                             *NeedSave=(need_save!=0);
01318                         }
01319                         return answer;
01320                     }
01321             private:
01322                 CDynLib& m_dynlib;
01323                 bigvoice_CClientInstance *client = nullptr;
01324         };
01332         std::shared_ptr<CClient> NewClient()
01333             {
01334                 assert(instance);
01335                 return std::make_shared<CClient>(instance, m_dynlib);
01336             }
01337 
01338     protected:
01339             template<typename T>
01340         unsigned int AddAudio(const T* payload, unsigned int nFrame)
01341             {
01342                 assert(instance);
01343                 return m_dynlib.AddAudio(instance, reinterpret_cast<const T*>(payload), nFrame,
    helper::SampleFormat<T>::format);
01344             }
01345             template<typename T>
01346         unsigned int GetAudio(T* payload, unsigned int max_nFrame)
01347             {
01348                 assert(instance);
01349                 return m_dynlib.GetAudio(instance, reinterpret_cast<T*>(payload), max_nFrame,
    helper::SampleFormat<T>::format);
01350             }
01351 
01352     private:
01353         CDynLib& m_dynlib;
01354         bigvoice_CParameters* params = nullptr;
01355         bigvoice_CInstance* instance = nullptr;
01356         std::thread update_thread;
01357         std::vector< std::pair<std::string,std::string> > init_metadata;
01358 #if BIGVOICE_HAS_WEBSERVER
01359         uint64_t webserver = SOUND4_INVALID_WEBSERVER_ID;
01360 #endif // BIGVOICE_HAS_WEBSERVER
01361     };
01362 
01363 }; // namespace dyn
01364 }; // namespace bigvoice
01365 }; // namespace sound4
01366 
01367 
01368 // bridge C callbacks to CPresetLoader
01369 extern "C" {
01370     static char *bigvoice_custom_reader(const fs_char *filename, void* userdata) {
```

```
01371         sound4::bigvoice::dyn::CPresetLoader
      *preset_loader=reinterpret_cast<sound4::bigvoice::dyn::CPresetLoader*>(userdata);
01372         std::string content=preset_loader->Read(filename);
01373         return strdup(content.c_str());
01374     }
01375     static void bigvoice_custom_reader_free(char *content, void* userdata) {
01376         if (!content) return;
01377         free(content);
01378     }
01379     static int bigvoice_custom_writer(const fs_char *filename, const char *content, void* userdata) {
01380         sound4::bigvoice::dyn::CPresetLoader
      *preset_loader=reinterpret_cast<sound4::bigvoice::dyn::CPresetLoader*>(userdata);
01381         auto res=preset_loader->Write(filename,content);
01382         return res?0:-1;
01383     }
01384     static int bigvoice_custom_exists(const fs_char *filename, void* userdata) {
01385         sound4::bigvoice::dyn::CPresetLoader
      *preset_loader=reinterpret_cast<sound4::bigvoice::dyn::CPresetLoader*>(userdata);
01386         auto res=preset_loader->Exists(filename);
01387         return res?0:-1;
01388     }
01389     static fs_char** bigvoice_custom_getall(void* userdata) {
01390         sound4::bigvoice::dyn::CPresetLoader
      *preset_loader=reinterpret_cast<sound4::bigvoice::dyn::CPresetLoader*>(userdata);
01391         auto res=preset_loader->GetAll();
01392         fs_char**all=(fs_char**)malloc((res.size()+1)*sizeof(const fs_char*));
01393         for (size_t n=0;n<res.size();n++) {
01394             all[n]=fs_strdup(res[n].c_str());
01395         }
01396         all[res.size()]=nullptr;
01397         return all;
01398     }
01399     static void bigvoice_custom_getall_free(fs_char** all, void* userdata) {
01400         if (!all) return;
01401         for (fs_char** one=all;*one;one++) {
01402             free(*one);
01403         }
01404         free(all);
01405     }
01406     static int bigvoice_custom_remove(const fs_char *filename, void* userdata) {
01407         sound4::bigvoice::dyn::CPresetLoader
      *preset_loader=reinterpret_cast<sound4::bigvoice::dyn::CPresetLoader*>(userdata);
01408         auto res=preset_loader->Remove(filename);
01409         return res?0:-1;
01410     }
01411     static int bigvoice_custom_rename(const fs_char *from, const fs_char *to, void* userdata) {
01412         sound4::bigvoice::dyn::CPresetLoader
      *preset_loader=reinterpret_cast<sound4::bigvoice::dyn::CPresetLoader*>(userdata);
01413         auto res=preset_loader->Rename(from,to);
01414         return res?0:-1;
01415     }
01416
01417 };
```

11.17 sound4cl.hpp File Reference

C++ wrapper for dynamic loading library.

```
#include <string>
#include <exception>
#include <thread>
#include <functional>
#include <memory>
#include <array>
#include <assert.h>
#include <stdexcept>
#include <filesystem>
#include <string.h>
#include <libloaderapi.h>
#include "sound4cl_cdef.h"
```

Classes

- struct sound4::helper::SampleFormat< T >

  Helper for Sample format types.
- struct sound4::helper::SampleFormat< int16_t >
- struct sound4::helper::SampleFormat< int32_t >
- struct sound4::helper::SampleFormat< float >
- class sound4::helper::DynFuncHolder< T >

  Helper to wrap a function pointer.
- class sound4::helper::CDynLoader

  Helper to load dynamic library.
- class sound4::CProcessor

  Dynamic library interface.
- class sound4::CBus

  Shared Bus.
- class sound4::CPresetLoader

  Custom preset handler helper.
- class sound4::CInstance

  Instance handling class.
- class sound4::CInstance::CClient

Namespaces

- namespace sound4
- namespace sound4::helper

Macros

- #define UNIXLIKE 0

Typedefs

- using sound4::helper::DynLib_t = HMODULE
- using sound4::helper::DynFunc_t = FARPROC
- using sound4::LogSeverity = sound4cl_LogSeverity
- using sound4::log_cb_t = std::function< void(LogSeverity, const std::string &)>
- using sound4::PresetChange_Kind = sound4cl_PresetChange_Kind

Variables

- const LogSeverity sound4::none = (LogSeverity) sound4cl_none
- const LogSeverity sound4::fatal = (LogSeverity) sound4cl_fatal
- const LogSeverity sound4::error = (LogSeverity) sound4cl_error
- const LogSeverity sound4::warning = (LogSeverity) sound4cl_warning
- const LogSeverity sound4::info = (LogSeverity) sound4cl_info
- const LogSeverity sound4::verbose = (LogSeverity) sound4cl_verbose
- const LogSeverity sound4::verbose2 = (LogSeverity) sound4cl_verbose2
- const LogSeverity sound4::verbose3 = (LogSeverity) sound4cl_verbose3
- const LogSeverity sound4::verbose4 = (LogSeverity) sound4cl_verbose4
- const LogSeverity sound4::verbose5 = (LogSeverity) sound4cl_verbose5

11.17.1 Detailed Description

You need to load the dynamic library, and if it succeed you can use it.

Definition in file sound4cl.hpp.

11.17.2 Macro Definition Documentation

11.17.2.1 UNIXLIKE

```
#define UNIXLIKE 0
```

Definition at line 24 of file sound4cl.hpp.

11.18 sound4cl.hpp

Go to the documentation of this file.

```
00001
00008 #pragma once
00009
00010 #include <string>
00011 #include <exception>
00012 #include <thread>
00013 #include <functional>
00014 #include <memory>
00015 #include <array>
00016 #include <assert.h>
00017 #include <stdexcept>
00018 #include <filesystem>
00019 #include <string.h>
00020
00021 #if defined(__unix__) || defined(__APPLE__)
00022     #define UNIXLIKE 1
00023 #else
00024     #define UNIXLIKE 0
00025 #endif
00026
00027 #if UNIXLIKE
00028     #include <dlfcn.h>
00029 #elif defined(_WIN32)
00030     #include <libloaderapi.h>
00031 #else
00032     #error "Unsupported OS"
00033 #endif
00034
00035 #include "sound4cl_cdef.h"
00036
00037     // bridge C callbacks to CPresetLoader
00038     extern "C" {
00039         static char *sound4cl_custom_reader(const fs_char *filename, void* userdata);
00040         static void sound4cl_custom_reader_free(char *content, void* userdata);
00041         static int sound4cl_custom_writer(const fs_char *filename, const char *content, void*
      userdata);
00042         static int sound4cl_custom_exists(const fs_char *filename, void* userdata);
00043         static fs_char** sound4cl_custom_getall(void* userdata);
00044         static void sound4cl_custom_getall_free(fs_char** all, void* userdata);
00045         static int sound4cl_custom_remove(const fs_char *filename, void* userdata);
00046         static int sound4cl_custom_rename(const fs_char *from, const fs_char *to, void* userdata);
00047     };
00048
00052 namespace sound4 {
00053
00057     namespace helper {
00058         #ifdef _WIN32
00064         static inline std::string WStringToUTF8(const std::wstring& wstr) {
00065             if (wstr.empty()) return {};
00066             int size = WideCharToMultiByte(CP_UTF8, 0, &wstr[0], (int)wstr.size(), NULL, 0, NULL,
      NULL);
00067             std::string str(size, 0);
00068             WideCharToMultiByte(CP_UTF8, 0, &wstr[0], (int)wstr.size(), &str[0], size, NULL, NULL);
00069             return str;
```

```
00070         }
00076         static inline std::wstring UTF8ToWString(const std::string& str) {
00077             if (str.empty()) return std::wstring();
00078             int size = MultiByteToWideChar(CP_UTF8, 0, &str[0], (int)str.size(), NULL, 0);
00079             std::wstring wstr(size, 0);
00080             MultiByteToWideChar(CP_UTF8, 0, &str[0], (int)str.size(), &wstr[0], size);
00081             return wstr;
00082         }
00083         #endif // _WIN32
00084
00090             template<typename T>
00091         struct SampleFormat {
00092         };
00093             template<>
00094         struct SampleFormat<int16_t> {
00095             const sound4cl_SampleFormat format = S16_NATIVE;
00096         };
00097             template<>
00098         struct SampleFormat<int32_t> {
00099             const sound4cl_SampleFormat format = S32_NATIVE;
00100         };
00101             template<>
00102         struct SampleFormat<float> {
00103             const sound4cl_SampleFormat format = F32_NATIVE;
00104         };
00105
00106         // Storage definition, depending on OS
00107         #if UNIXLIKE
00108             using DynLib_t = void *;
00109             using DynFunc_t = void *;
00110         #else
00111             using DynLib_t = HMODULE;
00112             using DynFunc_t = FARPROC;
00113         #endif
00114
00120             template <typename T>
00121         class DynFuncHolder {
00122         private:
00123             DynFunc_t m_ptr = NULL;
00124         public:
00125             DynFuncHolder() = default;
00126             DynFuncHolder(DynFunc_t a_ptr) : m_ptr(a_ptr) {}
00127             operator bool() const { return (m_ptr!=NULL); }
00128             bool IsOk() const { return m_ptr!=NULL; }
00129             operator T* () const { return reinterpret_cast<T*>(m_ptr); }
00130         };
00131
00136         class CDynLoader {
00137             private:
00138                 DynLib_t m_lib {};
00139             public:
00140                 CDynLoader() = default;
00141                 CDynLoader(CDynLoader&&) = default;
00142                 // No copy allowed
00143                 CDynLoader(const CDynLoader&) = delete;
00144                 CDynLoader& operator=(CDynLoader const&) = delete;
00145                 operator bool() const {
00146                     return (m_lib!=NULL);
00147                 }
00148                 bool IsOk() const {
00149                     return (m_lib!=NULL);
00150                 }
00151                 #if UNIXLIKE
00152                     void Close() {
00153                         if (IsOk()) {
00154                             #ifndef __SANITIZE_ADDRESS__ // avoid unloading to keep call stack
00155                             dlclose(m_lib);
00156                             #endif // __SANITIZE_ADDRESS__
00157                             m_lib=NULL;
00158                         }
00159                     }
00160                     bool Load(const std::filesystem::path& dynlib)
00161                     {
00162                         Close();
00163                         m_lib=dlopen(dynlib.c_str(), RTLD_NOW|RTLD_LOCAL);
00164                         return IsOk();
00165                     }
00166                         template <typename T>
00167                     DynFuncHolder<T> GetSymbol(const std::string& name) {
00168                         auto ptr=dlsym(m_lib, name.c_str());
00169                         if (!ptr) {
00170                             throw std::runtime_error("Missing function in library");
00171                         }
00172                         return DynFuncHolder<T>(ptr);
00173                     }
00174                         template <typename T>
00175                     DynFuncHolder<T> GetSymbol_safe(const std::string& name) {
```

```
00176                         auto ptr=dlsym(m_lib, name.c_str());
00177                         return DynFuncHolder<T>(ptr);
00178                     }
00179                     static std::filesystem::path GetThisLibraryPath(void) {
00180                         static Dl_info info;
00181                         if (! dladdr((void *) GetThisLibraryPath, & info)) return {};
00182                         auto rp=realpath(info.dli_fname, NULL);
00183                         std::filesystem::path p(rp);
00184                         free(rp);
00185                         return p;
00186                     }
00187                 #else
00188                     bool Load(const std::filesystem::path& dynlib, uint32_t loadflags =
    LOAD_LIBRARY_SEARCH_DLL_LOAD_DIR|LOAD_LIBRARY_SEARCH_APPLICATION_DIR|LOAD_LIBRARY_SEARCH_USER_DIRS|LOAD_LIBRARY_SEARCH_S
00189                     {
00190                         Close();
00191                         m_lib=LoadLibraryExW(dynlib.c_str(), NULL, loadflags);
00192                         return IsOk();
00193                     }
00194                     void Close() {
00195                         if (IsOk()) {
00196                             // NOTE: On Windows, FreeLibrary may create a lot of troubles, should use
    FreeLibraryAndExitThread but...
00197                             // So define SOUND4_CALL_FREELIBRARY before including this file if you
    really want to unload the library dynamically
00198                             #if defined(SOUND4_CALL_FREELIBRARYANDEXITTHREAD)
00199                                 FreeLibraryAndExitThread(m_lib, 0);
00200                             #elif defined(SOUND4_CALL_FREELIBRARY)
00201                                 FreeLibrary(m_lib);
00202                             #endif // SOUND4_CALL_FREELIBRARY
00203                             m_lib=NULL;
00204                         }
00205                     }
00206                         template <typename T>
00207                     DynFuncHolder<T> GetSymbol(const std::string& name) {
00208                         auto ptr=GetProcAddress(m_lib, name.c_str());
00209                         if (!ptr) {
00210                             throw std::runtime_error("Missing function in library");
00211                         }
00212                         return DynFuncHolder<T>(ptr);
00213                     }
00214                         template <typename T>
00215                     DynFuncHolder<T> GetSymbol_safe(const std::string& name) {
00216                         auto ptr=GetProcAddress(m_lib, name.c_str());
00217                         return DynFuncHolder<T>(ptr);
00218                     }
00219                     static std::filesystem::path GetThisLibraryPath(void) {
00220                         static wchar_t path[MAX_PATH]={};
00221                         HMODULE hm = NULL;
00222                         if (GetModuleHandleExW(GET_MODULE_HANDLE_EX_FLAG_FROM_ADDRESS |
00223                                 GET_MODULE_HANDLE_EX_FLAG_UNCHANGED_REFCOUNT,
00224                                 (LPCWSTR) &GetThisLibraryPath, &hm))
00225                         {
00226                             if (GetModuleFileNameW(hm, path, MAX_PATH) > 0) {
00227                                 return std::filesystem::path(path);
00228                             }
00229                         }
00230                         return {};
00231                     }
00232                 #endif
00233                 virtual ~CDynLoader() {
00234                     Close();
00235                 }
00236         }; // class CDynLoader
00237     }; // namespace helper
00238
00239
00243     using LogSeverity = sound4cl_LogSeverity;
00244     const LogSeverity none     = (LogSeverity) sound4cl_none;
00245     const LogSeverity fatal    = (LogSeverity) sound4cl_fatal;
00246     const LogSeverity error    = (LogSeverity) sound4cl_error;
00247     const LogSeverity warning  = (LogSeverity) sound4cl_warning;
00248     const LogSeverity info     = (LogSeverity) sound4cl_info;
00249     const LogSeverity verbose  = (LogSeverity) sound4cl_verbose;
00250     const LogSeverity verbose2 = (LogSeverity) sound4cl_verbose2;
00251     const LogSeverity verbose3 = (LogSeverity) sound4cl_verbose3;
00252     const LogSeverity verbose4 = (LogSeverity) sound4cl_verbose4;
00253     const LogSeverity verbose5 = (LogSeverity) sound4cl_verbose5;
00254
00258     using log_cb_t = std::function<void(LogSeverity,const std::string&)>;
00259
00263     static log_cb_t _log_cb;
00264     extern "C" {
00265         static inline void _log_cb_c(LogSeverity severity, const char *c_msg) {
00266             _log_cb(severity, std::string(c_msg));
00267         }
00268     }
```

```
00269
00276     class CProcessor {
00277     private:
00278         helper::CDynLoader m_lib;
00279         bool m_bOK = false;
00280         std::string m_prefix; // The prefix of functions, depends on the library (cloud_, impact_...)
00281     protected:
00282         friend class CBus;
00283         friend class CInstance;
00284
00285         // All those wrapped functions have the same signature as there C sound4cl_XXX
00286         helper::DynFuncHolder< decltype(sound4cl_GetVersion             ) > fnGetVersion             ;
00287         helper::DynFuncHolder< decltype(sound4cl_GetChunkSizeInFrames   ) > fnGetChunkSizeInFrames   ;
00288         helper::DynFuncHolder< decltype(sound4cl_GetChannelCount        ) > fnGetChannelCount        ;
00289         helper::DynFuncHolder< decltype(sound4cl_GetAudioInputCount     ) > fnGetAudioInputCount     ;
00290         helper::DynFuncHolder< decltype(sound4cl_GetAudioOutputCount    ) > fnGetAudioOutputCount    ;
00291         helper::DynFuncHolder< decltype(sound4cl_GetSampleRate          ) > fnGetSampleRate          ;
00292         helper::DynFuncHolder< decltype(sound4cl_SetLoggerCallback      ) > fnSetLoggerCallback      ;
00293         helper::DynFuncHolder< decltype(sound4cl_SetLogSeverity         ) > fnSetLogSeverity         ;
00294         helper::DynFuncHolder< decltype(sound4cl_NewParameters           ) > fnNewParameters          ;
00295         helper::DynFuncHolder< decltype(sound4cl_FreeParameters          ) > fnFreeParameters         ;
00296         helper::DynFuncHolder< decltype(sound4cl_SetParameter            ) > fnSetParameter           ;
00297         helper::DynFuncHolder< decltype(sound4cl_GetParameter            ) > fnGetParameter           ;
00298         helper::DynFuncHolder< decltype(sound4cl_FreeParameterValue      ) > fnFreeParameterValue     ;
00299         helper::DynFuncHolder< decltype(sound4cl_InitProcess             ) > fnInitProcess            ;
00300         helper::DynFuncHolder< decltype(sound4cl_InitProcess2            ) > fnInitProcess2           ;
00301         helper::DynFuncHolder< decltype(sound4cl_TerminateProcess        ) > fnTerminateProcess       ;
00302         helper::DynFuncHolder< decltype(sound4cl_ExitProcess             ) > fnExitProcess            ;
00303         helper::DynFuncHolder< decltype(sound4cl_StartUpdateThread       ) > fnStartUpdateThread      ;
00304         helper::DynFuncHolder< decltype(sound4cl_StopUpdateThread        ) > fnStopUpdateThread       ;
00305         helper::DynFuncHolder< decltype(sound4cl_WaitUpdateThreadReady   ) > fnWaitUpdateThreadReady  ;
00306         helper::DynFuncHolder< decltype(sound4cl_ProcessAudio            ) > fnProcessAudio           ;
00307         helper::DynFuncHolder< decltype(sound4cl_ProcessAudio_Planar     ) > fnProcessAudio_Planar    ;
00308         helper::DynFuncHolder< decltype(sound4cl_GetBufferIn             ) > fnGetBufferIn            ;
00309         helper::DynFuncHolder< decltype(sound4cl_GetBufferOut            ) > fnGetBufferOut           ;
00310         helper::DynFuncHolder< decltype(sound4cl_GetEstimatedDelay       ) > fnGetEstimatedDelay      ;
00311         helper::DynFuncHolder< decltype(sound4cl_GetFormatName           ) > fnGetFormatName          ;
00312         helper::DynFuncHolder< decltype(sound4cl_GetFormatFromName       ) > fnGetFormatFromName      ;
00313         helper::DynFuncHolder< decltype(sound4cl_GetBytesFromFormat      ) > fnGetBytesFromFormat     ;
00314         helper::DynFuncHolder< decltype(sound4cl_GetMaxPacketFrame       ) > fnGetMaxPacketFrame      ;
00315         helper::DynFuncHolder< decltype(sound4cl_AddAudio                ) > fnAddAudio               ;
00316         helper::DynFuncHolder< decltype(sound4cl_AddPadAudio             ) > fnAddPadAudio            ;
00317         helper::DynFuncHolder< decltype(sound4cl_GetOutputCount          ) > fnGetOutputCount         ;
00318         helper::DynFuncHolder< decltype(sound4cl_GetAudio                ) > fnGetAudio               ;
00319         helper::DynFuncHolder< decltype(sound4cl_AudioConvertFrom        ) > fnAudioConvertFrom       ;
00320         helper::DynFuncHolder< decltype(sound4cl_AudioConvertTo          ) > fnAudioConvertTo         ;
00321         helper::DynFuncHolder< decltype(sound4cl_StereoToMono            ) > fnStereoToMono           ;
00322         helper::DynFuncHolder< decltype(sound4cl_MonoToStereo            ) > fnMonoToStereo           ;
00323         helper::DynFuncHolder< decltype(sound4cl_AudioMonoFromLiveStereo) > fnAudioMonoFromLiveStereo;
00324         helper::DynFuncHolder< decltype(sound4cl_AudioMonoToLiveStereo   ) > fnAudioMonoToLiveStereo  ;
00325         helper::DynFuncHolder< decltype(sound4cl_NewClient               ) > fnNewClient              ;
00326         helper::DynFuncHolder< decltype(sound4cl_DeleteClient            ) > fnDeleteClient           ;
00327         helper::DynFuncHolder< decltype(sound4cl_ProcessJson             ) > fnProcessJson            ;
00328         helper::DynFuncHolder< decltype(sound4cl_FreeJsonAnswer          ) > fnFreeJsonAnswer         ;
00329         helper::DynFuncHolder< decltype(sound4cl_SaveState               ) > fnSaveState              ;
00330
00331         // only for processes supporting webserver
00332         helper::DynFuncHolder< decltype(sound4cl_Webserver_tcp           ) > fnWebserver_tcp          ;
00333         helper::DynFuncHolder< decltype(sound4cl_Webserver_tcp2          ) > fnWebserver_tcp2         ;
00334         helper::DynFuncHolder< decltype(sound4cl_Webserver               ) > fnWebserver              ;
00335         helper::DynFuncHolder< decltype(sound4cl_Webserver_Stop          ) > fnWebserver_Stop         ;
00336         helper::DynFuncHolder< decltype(sound4cl_Webserver_Status        ) > fnWebserver_Status       ;
00337
00338         // added 2023-03-22
00339         helper::DynFuncHolder< decltype(sound4cl_StereoToMono_Planar     ) > fnStereoToMono_Planar    ;
00340         helper::DynFuncHolder< decltype(sound4cl_MonoToStereo_Planar     ) > fnMonoToStereo_Planar    ;
00341
00342         // added 2023-05-22
00343         helper::DynFuncHolder< decltype(sound4cl_SetMetadata             ) > fnSetMetadata            ;
00344         helper::DynFuncHolder< decltype(sound4cl_GetMetadataInfos        ) > fnGetMetadataInfos       ;
00345         helper::DynFuncHolder< decltype(sound4cl_FreeMetadataInfos       ) > fnFreeMetadataInfos      ;
00346
00347         // added 2023-06-20
00348         helper::DynFuncHolder< decltype(sound4cl_SetPresetManager                ) >
      fnSetPresetManager      ;
00349         helper::DynFuncHolder< decltype(sound4cl_PresetManager_InformChange      ) >
      fnPresetManager_InformChange      ;
00350
00351         // added 2023-09-21
00352         helper::DynFuncHolder< decltype(sound4cl_GetPossibleChunkSizeInFrames    ) >
      fnGetPossibleChunkSizeInFrames   ;
00353         helper::DynFuncHolder< decltype(sound4cl_GetProcessChunkFrames           ) >
      fnGetProcessChunkFrames   ;
00354         helper::DynFuncHolder< decltype(sound4cl_InitProcess3                    ) > fnInitProcess3
      ;
00355
00356         // added 2023-06-26, only for processes supporting bus
```

```
00357         helper::DynFuncHolder< decltype(sound4cl_NewBus                ) > fnNewBus      ;
00358         helper::DynFuncHolder< decltype(sound4cl_FreeBus               ) > fnFreeBus      ;
00359         helper::DynFuncHolder< decltype(sound4cl_SetInstanceBus        ) > fnSetInstanceBus      ;
00360 
00361         // added 2024-02-19
00362         helper::DynFuncHolder< decltype(sound4cl_SetMetadataMulti      ) > fnSetMetadataMulti
    ;
00363 
00364         // added 2024-05-23
00365         helper::DynFuncHolder< decltype(sound4cl_Webserver_SetAppHealth    ) >
    fnWebserver_SetAppHealth      ;
00366         helper::DynFuncHolder< decltype(sound4cl_Webserver_GetAppHealth    ) >
    fnWebserver_GetAppHealth      ;
00367         helper::DynFuncHolder< decltype(sound4cl_Webserver_FreeString      ) > fnWebserver_FreeString
    ;
00368 
00369             template <typename T>
00370         helper::DynFuncHolder<T> GetPrefixSymbol(const std::string& name) {
00371             std::string prefix_name = m_prefix+name;
00372             return m_lib.GetSymbol<T>(prefix_name);
00373         }
00374 
00375     public:
00376         CProcessor() = default;
00377 
00385         bool Load(const std::filesystem::path& filepath) {
00386             m_lib.Close();
00387             // If a path is given, use it directly and do not try other path
00388             if (filepath.has_parent_path() && !m_lib.Load(filepath)) {
00389                 return false;
00390             } else if (!m_lib.IsOk()) {
00391                 auto thisdir=helper::CDynLoader::GetThisLibraryPath().parent_path();
00392                 auto filename = filepath.filename();
00393                 #ifndef _WIN32
00394                 // Linux: if in a bin directory, try ../lib first
00395                 if (!m_lib.IsOk() && thisdir.filename()=="bin") {
00396                     auto libdir = thisdir.parent_path() / "lib";
00397                     m_lib.Load(libdir / filename);
00398                 }
00399                 #endif // !_WIN32
00400                 if (!m_lib.IsOk()) {
00401                     // Search in the same directory this code is in
00402                     m_lib.Load(thisdir / filename);
00403                 }
00404                 if (!m_lib.IsOk()) {
00405                     // Try current path
00406                     std::error_code ec;
00407                     auto p = std::filesystem::current_path(ec);
00408                     if (!ec) {
00409                         m_lib.Load(p / filename);
00410                     }
00411                 }
00412                 if (!m_lib.IsOk()) {
00413                     return false;
00414                 }
00415             }
00416             // Get information from the library itself
00417             try {
00418                 auto GetInfo = m_lib.GetSymbol< decltype(sound4cl_SOUND4_GetProcessInfo             )
    >("SOUND4_GetProcessInfo"   );
00419                 auto info = GetInfo();
00420                 m_prefix = info->prefix;
00421             } catch (std::runtime_error& ) {
00422                 // Compatibility: search in known libraries without this function
00423                 #ifdef _WIN32
00424                     std::string filename = helper::WStringToUTF8(filepath.stem());
00425                 #else // !_WIN32
00426                     std::string filename = filepath.stem();
00427                     // remove lib prefix
00428                     filename=filename.substr(3);
00429                 #endif // ! _WIN32
00430                 if (filename=="sound4.x1.cloud") {
00431                     m_prefix="cloudx1_";
00432                 } else if (filename=="sound4.impact.cl") {
00433                     m_prefix="impact_";
00434                 } else if (filename=="sound4.bigvoice.cl") {
00435                     m_prefix="bigvoice_";
00436                 } else {
00437                     return false;
00438                 }
00439             }
00440 
00441             // Load all C functions
00442             try {
00443                 fnGetVersion              = GetPrefixSymbol< decltype(sound4cl_GetVersion
    ) >("GetVersion"            );
00444                 fnGetChunkSizeInFrames    = GetPrefixSymbol< decltype(sound4cl_GetChunkSizeInFrames
```

```
      ) >("GetChunkSizeInFrames"   );
00445                 fnGetChannelCount         = GetPrefixSymbol< decltype(sound4cl_GetChannelCount
      ) >("GetChannelCount"        );
00446                 fnGetAudioInputCount      = GetPrefixSymbol< decltype(sound4cl_GetAudioInputCount
      ) >("GetAudioInputCount"     );
00447                 fnGetAudioOutputCount     = GetPrefixSymbol< decltype(sound4cl_GetAudioOutputCount
      ) >("GetAudioOutputCount"    );
00448                 fnGetSampleRate           = GetPrefixSymbol< decltype(sound4cl_GetSampleRate
      ) >("GetSampleRate"          );
00449                 fnSetLoggerCallback       = GetPrefixSymbol< decltype(sound4cl_SetLoggerCallback
      ) >("SetLoggerCallback"      );
00450                 fnSetLogSeverity          = GetPrefixSymbol< decltype(sound4cl_SetLogSeverity
      ) >("SetLogSeverity"         );
00451                 fnNewParameters           = GetPrefixSymbol< decltype(sound4cl_NewParameters
      ) >("NewParameters"          );
00452                 fnFreeParameters          = GetPrefixSymbol< decltype(sound4cl_FreeParameters
      ) >("FreeParameters"         );
00453                 fnSetParameter            = GetPrefixSymbol< decltype(sound4cl_SetParameter
      ) >("SetParameter"           );
00454                 fnGetParameter            = GetPrefixSymbol< decltype(sound4cl_GetParameter
      ) >("GetParameter"           );
00455                 fnFreeParameterValue      = GetPrefixSymbol< decltype(sound4cl_FreeParameterValue
      ) >("FreeParameterValue"     );
00456                 fnInitProcess             = GetPrefixSymbol< decltype(sound4cl_InitProcess
      ) >("InitProcess"            );
00457                 fnInitProcess2            = GetPrefixSymbol< decltype(sound4cl_InitProcess2
      ) >("InitProcess2"           );
00458                 fnTerminateProcess        = GetPrefixSymbol< decltype(sound4cl_TerminateProcess
      ) >("TerminateProcess"       );
00459                 fnExitProcess             = GetPrefixSymbol< decltype(sound4cl_ExitProcess
      ) >("ExitProcess"            );
00460                 fnStartUpdateThread       = GetPrefixSymbol< decltype(sound4cl_StartUpdateThread
      ) >("StartUpdateThread"      );
00461                 fnStopUpdateThread        = GetPrefixSymbol< decltype(sound4cl_StopUpdateThread
      ) >("StopUpdateThread"       );
00462                 fnWaitUpdateThreadReady   = GetPrefixSymbol< decltype(sound4cl_WaitUpdateThreadReady
      ) >("WaitUpdateThreadReady"  );
00463                 fnProcessAudio            = GetPrefixSymbol< decltype(sound4cl_ProcessAudio
      ) >("ProcessAudio"           );
00464                 fnProcessAudio_Planar     = GetPrefixSymbol< decltype(sound4cl_ProcessAudio_Planar
      ) >("ProcessAudio_Planar"    );
00465                 fnGetBufferIn             = GetPrefixSymbol< decltype(sound4cl_GetBufferIn
      ) >("GetBufferIn"            );
00466                 fnGetBufferOut            = GetPrefixSymbol< decltype(sound4cl_GetBufferOut
      ) >("GetBufferOut"           );
00467                 fnGetEstimatedDelay       = GetPrefixSymbol< decltype(sound4cl_GetEstimatedDelay
      ) >("GetEstimatedDelay"      );
00468                 fnGetFormatName           = GetPrefixSymbol< decltype(sound4cl_GetFormatName
      ) >("GetFormatName"          );
00469                 fnGetFormatFromName       = GetPrefixSymbol< decltype(sound4cl_GetFormatFromName
      ) >("GetFormatFromName"      );
00470                 fnGetBytesFromFormat      = GetPrefixSymbol< decltype(sound4cl_GetBytesFromFormat
      ) >("GetBytesFromFormat"     );
00471                 fnGetMaxPacketFrame       = GetPrefixSymbol< decltype(sound4cl_GetMaxPacketFrame
      ) >("GetMaxPacketFrame"      );
00472                 fnAddAudio                = GetPrefixSymbol< decltype(sound4cl_AddAudio
      ) >("AddAudio"               );
00473                 fnAddPadAudio             = GetPrefixSymbol< decltype(sound4cl_AddPadAudio
      ) >("AddPadAudio"            );
00474                 fnGetOutputCount          = GetPrefixSymbol< decltype(sound4cl_GetOutputCount
      ) >("GetOutputCount"         );
00475                 fnGetAudio                = GetPrefixSymbol< decltype(sound4cl_GetAudio
      ) >("GetAudio"               );
00476                 fnAudioConvertFrom        = GetPrefixSymbol< decltype(sound4cl_AudioConvertFrom
      ) >("AudioConvertFrom"       );
00477                 fnAudioConvertTo          = GetPrefixSymbol< decltype(sound4cl_AudioConvertTo
      ) >("AudioConvertTo"         );
00478                 fnStereoToMono            = GetPrefixSymbol< decltype(sound4cl_StereoToMono
      ) >("StereoToMono"           );
00479                 fnMonoToStereo            = GetPrefixSymbol< decltype(sound4cl_MonoToStereo
      ) >("MonoToStereo"           );
00480                 fnAudioMonoFromLiveStereo = GetPrefixSymbol<
      decltype(sound4cl_AudioMonoFromLiveStereo) >("AudioMonoFromLiveStereo");
00481                 fnAudioMonoToLiveStereo   = GetPrefixSymbol< decltype(sound4cl_AudioMonoToLiveStereo
      ) >("AudioMonoToLiveStereo"  );
00482                 fnNewClient               = GetPrefixSymbol< decltype(sound4cl_NewClient
      ) >("NewClient"              );
00483                 fnDeleteClient            = GetPrefixSymbol< decltype(sound4cl_DeleteClient
      ) >("DeleteClient"           );
00484                 fnProcessJson             = GetPrefixSymbol< decltype(sound4cl_ProcessJson
      ) >("ProcessJson"            );
00485                 fnFreeJsonAnswer          = GetPrefixSymbol< decltype(sound4cl_FreeJsonAnswer
      ) >("FreeJsonAnswer"         );
00486                 fnSaveState               = GetPrefixSymbol< decltype(sound4cl_SaveState
      ) >("SaveState"              );
00487             } catch (std::runtime_error& ) {
00488                 return false;
```

```
00489             }
00490             // Webserver : not for all projects
00491             try {
00492                 fnWebserver_tcp           = GetPrefixSymbol< decltype(sound4cl_Webserver_tcp
      ) >("Webserver_tcp"           );
00493                 fnWebserver_tcp2          = GetPrefixSymbol< decltype(sound4cl_Webserver_tcp2
      ) >("Webserver_tcp2"          );
00494                 fnWebserver               = GetPrefixSymbol< decltype(sound4cl_Webserver
      ) >("Webserver"               );
00495                 fnWebserver_Stop          = GetPrefixSymbol< decltype(sound4cl_Webserver_Stop
      ) >("Webserver_Stop"          );
00496                 fnWebserver_Status        = GetPrefixSymbol< decltype(sound4cl_Webserver_Status
      ) >("Webserver_Status"        );
00497             } catch (std::runtime_error& ) {
00498                 // Ignored, handler will take care
00499             }
00500             // Stereo/Mono conversion : must be present
00501             try {
00502                 // added 2023-03-22
00503                 fnStereoToMono_Planar     = GetPrefixSymbol< decltype(sound4cl_StereoToMono_Planar
      ) >("StereoToMono_Planar"     );
00504                 fnMonoToStereo_Planar     = GetPrefixSymbol< decltype(sound4cl_MonoToStereo_Planar
      ) >("MonoToStereo_Planar"     );
00505
00506             } catch (std::runtime_error& ) {
00507                 return false;
00508             }
00509             // C functions allowed to be missing
00510             try {
00511                 // added 2023-05-22
00512                 fnSetMetadata             = GetPrefixSymbol< decltype(sound4cl_SetMetadata
      ) >("SetMetadata"             );
00513                 fnGetMetadataInfos        = GetPrefixSymbol< decltype(sound4cl_GetMetadataInfos
      ) >("GetMetadataInfos"        );
00514                 fnFreeMetadataInfos       = GetPrefixSymbol< decltype(sound4cl_FreeMetadataInfos
      ) >("FreeMetadataInfos"       );
00515                 // added 2023-06-20
00516                 fnSetPresetManager            = GetPrefixSymbol< decltype(sound4cl_SetPresetManager
      ) >("SetPresetManager"            );
00517                 fnPresetManager_InformChange  = GetPrefixSymbol<
      decltype(sound4cl_PresetManager_InformChange     ) >("PresetManager_InformChange"    );
00518
00519                 // added 2023-09-21
00520                 fnGetPossibleChunkSizeInFrames = GetPrefixSymbol<
      decltype(sound4cl_GetPossibleChunkSizeInFrames  ) >("GetPossibleChunkSizeInFrames"  );
00521                 fnGetProcessChunkFrames        = GetPrefixSymbol<
      decltype(sound4cl_GetProcessChunkFrames         ) >("GetProcessChunkFrames"         );
00522                 fnInitProcess3                 = GetPrefixSymbol< decltype(sound4cl_InitProcess3
      ) >("InitProcess3"                );
00523             }  catch (std::runtime_error& ) {
00524                 // Ignored, handler will take care
00525             }
00526
00527             // bus: not for all projects
00528             try {
00529                 // added 2023-06-26
00530                 fnNewBus                  = GetPrefixSymbol< decltype(sound4cl_NewBus
      ) >("NewBus"                  );
00531                 fnFreeBus                 = GetPrefixSymbol< decltype(sound4cl_FreeBus
      ) >("FreeBus"                 );
00532                 fnSetInstanceBus          = GetPrefixSymbol< decltype(sound4cl_SetInstanceBus
      ) >("SetInstanceBus"          );
00533             }  catch (std::runtime_error& ) {
00534                 // Ignored, handler will take care
00535             }
00536
00537             // Other optional
00538             try {
00539                 // added 2024-02-19
00540                 fnSetMetadataMulti        = GetPrefixSymbol< decltype(sound4cl_SetMetadataMulti
      ) >("SetMetadataMulti"        );
00541
00542             }  catch (std::runtime_error& ) {
00543                 // Ignored, handler will take care
00544             }
00545
00546             // Other optional, web server only
00547             try {
00548                 // added 2024-05-23
00549                 fnWebserver_SetAppHealth    = GetPrefixSymbol<
      decltype(sound4cl_Webserver_SetAppHealth    ) >("Webserver_SetAppHealth"     );
00550                 fnWebserver_GetAppHealth    = GetPrefixSymbol<
      decltype(sound4cl_Webserver_GetAppHealth    ) >("Webserver_GetAppHealth"     );
00551                 fnWebserver_FreeString      = GetPrefixSymbol< decltype(sound4cl_Webserver_FreeString
      ) >("Webserver_FreeString"        );
00552             }  catch (std::runtime_error& ) {
00553                 // Ignored, handler will take care
```

```
00554             }
00555
00556             m_bOK=true;
00557             return true;
00558         }
00565         bool IsOk() const {
00566             return m_bOK;
00567         }
00568
00573         void AudioConvertFrom(const uint8_t *payload, float *output, size_t nSpl,
      sound4cl_SampleFormat fmt)
00574             { fnAudioConvertFrom(payload, output, nSpl, fmt); }
00575         void AudioConvertTo(const float *input, uint8_t *payload, size_t nSpl, sound4cl_SampleFormat
      fmt)
00576             { fnAudioConvertTo(input, payload, nSpl, fmt); }
00577         void StereoToMono(const float *input, float *output, size_t nSpl)
00578             { fnStereoToMono(input, output, nSpl); }
00579         void MonoToStereo(const float *input, float *output, size_t nSpl)
00580             { fnMonoToStereo(input, output, nSpl); }
00581         void StereoToMono_Planar(const float *inputL, const float *inputR, float *output, size_t nSpl)
00582             { fnStereoToMono_Planar(inputL, inputR, output, nSpl); }
00583         void MonoToStereo_Planar(const float *input, float *outputL, float *outputR, size_t nSpl)
00584             { fnMonoToStereo_Planar(input, outputL, outputR, nSpl); }
00585         void AudioMonoFromLiveStereo(const uint8_t *payload, float *output)
00586             { fnAudioMonoFromLiveStereo(payload, output); }
00587         void AudioMonoToLiveStereo(const float *input, uint8_t *payload)
00588             { fnAudioMonoToLiveStereo(input, payload); }
00589
00595         std::string GetVersion()
00596             { return std::string(fnGetVersion()); }
00597
00606         unsigned int GetChunkSizeInFrames()
00607             { return fnGetChunkSizeInFrames(); }
00608
00617         std::vector<unsigned int> GetPossibleChunkSizeInFrames()
00618             {
00619                 std::vector<unsigned int> list;
00620                 if (fnGetPossibleChunkSizeInFrames) {
00621                     for (unsigned int* src=fnGetPossibleChunkSizeInFrames();*src;src++) {
00622                         list.push_back(*src);
00623                     }
00624                 } else {
00625                     list.push_back(fnGetChunkSizeInFrames());
00626                 }
00627                 return list;
00628             }
00629
00635         unsigned int GetChannelCount()
00636             { return fnGetChannelCount(); }
00637
00645         unsigned int GetAudioInputCount()
00646             { return fnGetAudioInputCount(); }
00647
00655         unsigned int GetAudioOutputCount()
00656             { return fnGetAudioOutputCount(); }
00657
00665         unsigned int GetSampleRate()
00666             { return fnGetSampleRate(); }
00667
00674         std::string GetFormatName( const sound4cl_SampleFormat fmt)
00675             { return std::string(fnGetFormatName(fmt)); }
00676
00683         sound4cl_SampleFormat GetFormatFromName( const std::string& name)
00684             { return fnGetFormatFromName(name.c_str()); }
00685
00692         unsigned int GetBytesFromFormat( const sound4cl_SampleFormat fmt)
00693             { return fnGetBytesFromFormat(fmt); }
00694
00698         void SetLogSeverity(LogSeverity severity)
00699             { fnSetLogSeverity(severity); }
00700
00704         void SetLoggerCallback(log_cb_t cb)
00705             { _log_cb=cb; fnSetLoggerCallback(&_log_cb_c); }
00706     };
00707
00713     class CBus {
00714         public:
00715             CBus(CProcessor& dynlib)
00716                 : m_dynlib(dynlib)
00717                 {
00718                     if (m_dynlib.fnNewBus) {
00719                         m_bus=m_dynlib.fnNewBus();
00720                     }
00721                 }
00722             ~CBus()
00723                 {
00724                     if (m_bus && m_dynlib.fnFreeBus) {
```

```
00725                       m_dynlib.fnFreeBus(m_bus);
00726                   }
00727                   m_bus = nullptr;
00728               }
00729           sound4cl_CBus *Get() const { return m_bus; }
00730       private:
00731           CProcessor& m_dynlib;
00732           sound4cl_CBus *m_bus;
00733   };
00734
00740   class CPresetLoader {
00741   public:
00742       CPresetLoader() = default;
00743       virtual ~CPresetLoader() = default;
00744
00745       virtual bool IsReadOnly() = 0;
00746       virtual bool Exists(const std::filesystem::path &name) = 0;
00747       virtual bool Remove(const std::filesystem::path &name) = 0;
00748       virtual bool Rename(const std::filesystem::path &from, const std::filesystem::path &to) = 0;
00749       virtual std::vector<std::filesystem::path> GetAll() = 0;
00750       virtual std::string Read(const std::filesystem::path &filename) = 0;
00751       virtual bool Write(const std::filesystem::path &filename, const std::string &content) =0;
00752   };
00753   using PresetChange_Kind = sound4cl_PresetChange_Kind;
00754
00760   class CInstance {
00761   public:
00765       CInstance(CProcessor& dynlib)
00766       : m_dynlib(dynlib)
00767           {
00768               params = m_dynlib.fnNewParameters();
00769               if (!params) throw std::bad_alloc();
00770           }
00771       // non-copyable
00772       CInstance( const CInstance& ) = delete;
00773       CInstance& operator=( const CInstance& ) = delete;
00777       virtual ~CInstance()
00778           {
00779               Stop();
00780               if (params) m_dynlib.fnFreeParameters(params);
00781           }
00785      bool IsOk() const { return instance!=nullptr; }
00793       void SetParam(const std::string& name, const std::string& value)
00794           {
00795               assert(params);
00796               m_dynlib.fnSetParameter(params, name.c_str(), value.c_str());
00797           }
00803       std::string GetParam(const std::string& name)
00804           {
00805               assert(params);
00806               auto c_value=m_dynlib.fnGetParameter(params, name.c_str());
00807               std::string ret(c_value);
00808               m_dynlib.fnFreeParameterValue(c_value);
00809               return ret;
00810           }
00811
00815       void SetBus(const CBus& bus) {
00816           assert(params);
00817           if (m_dynlib.fnSetInstanceBus) {
00818               m_dynlib.fnSetInstanceBus(params,bus.Get());
00819           }
00820       }
00821
00831       void SetPresetManager(CPresetLoader *preset_manager)
00832           {
00833               if (!preset_manager) return;
00834               if (!m_dynlib.fnSetPresetManager) return;
00835               assert(params);
00836               m_dynlib.fnSetPresetManager(params,
00837                   sound4cl_custom_reader,
00838                   sound4cl_custom_reader_free,
00839                   sound4cl_custom_writer,
00840                   sound4cl_custom_exists,
00841                   sound4cl_custom_getall,
00842                   sound4cl_custom_getall_free,
00843                   sound4cl_custom_remove,
00844                   sound4cl_custom_rename,
00845                   preset_manager->IsReadOnly(),
00846                   preset_manager
00847               );
00848           }
00849
00864       bool Create(const std::string& LoginKey, const std::string& RadioName, const std::string&
      Access_Key_ID, const std::string& Access_Key_Secret, const std::filesystem::path& save_path, int
      json_port = 0, unsigned int frames_per_chunk=0)
00865           {
00866               assert(!instance);
```

```
00867                 std::string l_save_path;
00868                 if (!save_path.empty()) {
00869                     l_save_path = save_path.u8string();
00870                 }
00871                 if (frames_per_chunk==0) {
00872                     frames_per_chunk=m_dynlib.GetChunkSizeInFrames();
00873                 }
00874                 if (m_dynlib.fnInitProcess3) {
00875                     instance = m_dynlib.fnInitProcess3(LoginKey.c_str(), RadioName.c_str(),
      Access_Key_ID.c_str(), Access_Key_Secret.c_str(), l_save_path.c_str(), params, frames_per_chunk);
00876                 } else if (frames_per_chunk!=m_dynlib.GetChunkSizeInFrames()) {
00877                     return false;
00878                 } else {
00879                     instance = m_dynlib.fnInitProcess2(LoginKey.c_str(), RadioName.c_str(),
      Access_Key_ID.c_str(), Access_Key_Secret.c_str(), l_save_path.c_str(), params);
00880                 }
00881                 if (!instance) return false;
00882                 if (!init_metadata.empty()) {
00883                     for (auto&& [key,value]: init_metadata) {
00884                         m_dynlib.fnSetMetadata(instance, key.c_str(), value.c_str());
00885                     }
00886                     init_metadata.clear();
00887                 }
00888                 update_thread = std::thread([this, json_port](){
00889                     OnUpdateThreadStart();
00890                     m_dynlib.fnStartUpdateThread(instance, json_port);
00891                     OnUpdateThreadStop();
00892                 });
00893                 int timeout=5000;
00894                 #ifdef DEBUG
00895                     timeout=60*1000; // For debugger pauses
00896                 #endif
00897                 if (m_dynlib.fnWaitUpdateThreadReady(instance, timeout)<0) {
00898                     return false;
00899                 }
00900                 return true;
00901             }
00902
00910         unsigned int GetChunkFrames() {
00911             if (instance && m_dynlib.fnGetProcessChunkFrames) {
00912                 return m_dynlib.fnGetProcessChunkFrames(instance);
00913             }
00914             return m_dynlib.fnGetChunkSizeInFrames();
00915         }
00916
00924         void PresetManager_InformChange(const std::filesystem::path& relative_path, PresetChange_Kind
      change_kind)
00925             {
00926                 if (m_dynlib.fnPresetManager_InformChange) {
00927                     m_dynlib.fnPresetManager_InformChange(instance, relative_path.c_str(),
      (sound4cl_PresetChange_Kind) change_kind);
00928                 }
00929             }
00930
00942         void SetMetadata(const std::string& key, const char* value)
00943             {
00944                 if (m_dynlib.fnSetMetadata) {
00945                     if (instance) {
00946                         m_dynlib.fnSetMetadata(instance, key.c_str(), value);
00947                     } else {
00948                         init_metadata.push_back( {key, value} );
00949                     }
00950                 }
00951             }
00952
00953
00963         void SetMetadataMulti(const std::unordered_map<std::string, const char*>& list)
00964             {
00965                 if (instance) {
00966                     if (m_dynlib.fnSetMetadataMulti) {
00967                         typedef const char* pchar;
00968                         pchar* keyvalue=new pchar[2*list.size()+1];
00969                         size_t n=0;
00970                         for (auto&& [key,value]: list) {
00971                             keyvalue[2*n+0]=key.c_str();
00972                             keyvalue[2*n+1]=value;
00973                             n++;
00974                         }
00975                         keyvalue[2*n+0]=nullptr;
00976
00977                         sound4cl_SetMetadataMulti(instance, keyvalue);
00978
00979                         delete[] keyvalue;
00980                     } else {
00981                         for (auto&& [key,value]: list) {
00982                             SetMetadata(key, value);
00983                         }
```

```
00984                   }
00985               } else {
00986                   for (auto&& [key,value]: list) {
00987                       if (value) {
00988                           init_metadata.push_back( {key, value} );
00989                       }
00990                   }
00991               }
00992           }
00993
00999       std::vector< std::tuple<std::string,std::string> > GetMetadataInfos()
01000           {
01001               std::vector< std::tuple<std::string,std::string> > values;
01002               if (m_dynlib.fnGetMetadataInfos) {
01003                   const char** c_values = m_dynlib.fnGetMetadataInfos(instance);
01004                   if (c_values) {
01005                       for (const char** c_value=c_values; *c_value; ) {
01006                           std::string key(*c_value);
01007                           c_value++;
01008                           if (*c_value) {
01009                               std::string descr(*c_value);
01010                               c_value++;
01011                               values.push_back( {key,descr} );
01012                           }
01013                       }
01014                       m_dynlib.fnFreeMetadataInfos(instance, c_values);
01015                   }
01016               }
01017               return values;
01018           }
01019
01020       // Can be overloaded, called at start of UpdateThread in the thread
01021       virtual void OnUpdateThreadStart() {}
01022       // Can be overloaded to call at end of UpdateThread in the thread
01023       virtual void OnUpdateThreadStop() {}
01024
01030       bool StartWebServer(int http_port, int https_port=0)
01031           {
01032               assert(instance);
01033               assert(webserver==SOUND4_INVALID_WEBSERVER_ID);
01034               if (!m_dynlib.fnWebserver) {
01035                   return false;
01036               }
01037               webserver = m_dynlib.fnWebserver(http_port,https_port,instance);
01038               if (webserver==SOUND4_INVALID_WEBSERVER_ID) {
01039                   return false;
01040               }
01041               int web_status=m_dynlib.fnWebserver_Status(webserver);
01042               if (http_port && (web_status & SOUND4_WEBSERVER_HTTP_OK) == 0) {
01043                   return false;
01044               }
01045               if (https_port && (web_status & SOUND4_WEBSERVER_HTTPS_OK) == 0) {
01046                   return false;
01047               }
01048               return true;
01049           }
01055       void StopWebServer()
01056           {
01057               if (!m_dynlib.fnWebserver_Stop) return;
01058               if (webserver != SOUND4_INVALID_WEBSERVER_ID) {
01059                   m_dynlib.fnWebserver_Stop(webserver, 1000);
01060                   webserver = SOUND4_INVALID_WEBSERVER_ID;
01061               }
01062           }
01063
01067       void SetWebServerAppHealth(int httpcode, const std::string& contenttype, const std::string&
      content)
01068           {
01069               assert(instance);
01070               if (!m_dynlib.fnWebserver_SetAppHealth) return;
01071               m_dynlib.fnWebserver_SetAppHealth(instance, httpcode, contenttype.c_str(),
      content.c_str());
01072           }
01076       void GetWebServerAppHealth(int &httpcode, std::string& contenttype, std::string& content)
01077           {
01078               assert(instance);
01079               if (!m_dynlib.fnWebserver_GetAppHealth) return;
01080               char* c_contenttype=nullptr;
01081               char* c_content=nullptr;
01082               m_dynlib.fnWebserver_GetAppHealth(instance, &httpcode, &c_contenttype, &c_content);
01083               contenttype=c_contenttype;
01084               content=c_content;
01085               m_dynlib.fnWebserver_FreeString(c_contenttype);
01086               m_dynlib.fnWebserver_FreeString(c_content);
01087           }
01094       int Terminate()
01095           {
```

```
01096                 if (!instance) return 0;
01097                 return m_dynlib.fnTerminateProcess(instance);
01098             }
01099 
01105         void Stop()
01106             {
01107                 if (!instance) return;
01108                 StopWebServer();
01109                 if (update_thread.joinable()) {
01110                     m_dynlib.fnStopUpdateThread(instance); // this will exit the update thread
01111                     update_thread.join();
01112                 }
01113                 m_dynlib.fnExitProcess(instance);
01114                 instance=nullptr;
01115             }
01116 
01122         unsigned int GetEstimatedDelay()
01123             {
01124                 assert(instance);
01125                 return m_dynlib.fnGetEstimatedDelay(instance);
01126             }
01127 
01138         float* GetBufferIn()
01139             {
01140                 assert(instance);
01141                 return m_dynlib.fnGetBufferIn(instance);
01142 
01143             }
01154         float* GetBufferOut()
01155             {
01156                 assert(instance);
01157                 return m_dynlib.fnGetBufferOut(instance);
01158             }
01164         void ProcessAudio()
01165             {
01166                 assert(instance);
01167                 m_dynlib.fnProcessAudio(instance, m_dynlib.fnGetBufferIn(instance),
      m_dynlib.fnGetBufferOut(instance));
01168             }
01182         // void ProcessAudio_Planar(const std::array< const std::array<const float,ChunkFrames>,
      InputChannels>& input, std::array< std::array<float,ChunkFrames>, OutputChannels>& output)
01183         void ProcessAudio_Planar(float const * const *input, float * const *output)
01184             {
01185                 assert(instance);
01186                 m_dynlib.fnProcessAudio_Planar(instance, input, output);
01187             }
01188 
01200             template<typename T>
01201         std::vector<T> ProcessAnyAudio(const std::vector<T> input)
01202             {
01203                 assert(instance);
01204                 std::vector<T> output;
01205                 unsigned int out_offset=0;
01206                 unsigned int in_offset=0;
01207                 unsigned int todo = input.size();
01208                 while (todo>0) {
01209                     unsigned int left = AddAudio(&input[in_offset], todo);
01210                     unsigned int out_avail = m_dynlib.fnGetOutputCount(instance);
01211                     output.resize(out_offset + out_avail);
01212                     GetAudio(&output[out_offset], out_avail);
01213                     out_offset+=out_avail;
01214                     in_offset += todo-left;
01215                     todo=left;
01216                 }
01217                 return output;
01218             }
01219 
01225         class CClient {
01226             public:
01227                 CClient(sound4cl_CInstance* instance, CProcessor& dynlib)
01228                 : m_dynlib(dynlib)
01229                     {
01230                         assert(instance);
01231                         client=m_dynlib.fnNewClient(instance);
01232                         if (!client) throw std::bad_alloc();
01233                     }
01234                 // non-copyable
01235                 CClient( const CInstance& ) = delete;
01236                 CClient& operator=( const CInstance& ) = delete;
01237                 ~CClient()
01238                     {
01239                         if (client) {
01240                             m_dynlib.fnDeleteClient(client);
01241                             client=nullptr;
01242                         }
01243                     }
01251                 std::string ProcessJson(const std::string &request, bool *NeedSave = nullptr)
```

```
01252                     {
01253                         assert(client);
01254                         int need_save=0;
01255                         const char *canswer = m_dynlib.fnProcessJson(client, request.c_str(),
    &need_save);
01256                         if (!canswer) return {};
01257                         std::string answer(canswer);
01258                         m_dynlib.fnFreeJsonAnswer (canswer);
01259                         if (NeedSave) {
01260                             *NeedSave=(need_save!=0);
01261                         }
01262                         return answer;
01263                     }
01264             private:
01265                 CProcessor& m_dynlib;
01266                 sound4cl_CClientInstance *client = nullptr;
01267         };
01275         std::shared_ptr<CClient> NewClient()
01276             {
01277                 assert(instance);
01278                 return std::make_shared<CClient>(instance, m_dynlib);
01279             }
01280
01281     protected:
01282             template<typename T>
01283         unsigned int AddAudio(const T* payload, unsigned int nFrame)
01284             {
01285                 assert(instance);
01286                 return m_dynlib.fnAddAudio(instance, reinterpret_cast<const T*>(payload), nFrame,
    helper::SampleFormat<T>::format);
01287             }
01288             template<typename T>
01289         unsigned int GetAudio(T* payload, unsigned int max_nFrame)
01290             {
01291                 assert(instance);
01292                 return m_dynlib.fnGetAudio(instance, reinterpret_cast<T*>(payload), max_nFrame,
    helper::SampleFormat<T>::format);
01293             }
01294
01295     private:
01296         CProcessor& m_dynlib;
01297         sound4cl_CParameters* params = nullptr;
01298         sound4cl_CInstance* instance = nullptr;
01299         std::thread update_thread;
01300         std::vector< std::pair<std::string,std::string> > init_metadata;
01301         uint64_t webserver = SOUND4_INVALID_WEBSERVER_ID;
01302     };
01303
01304 }; // namespace sound4
01305
01306
01307 // bridge C callbacks to CPresetLoader
01308 extern "C" {
01309     static char *sound4cl_custom_reader(const fs_char *filename, void* userdata) {
01310         sound4::CPresetLoader *preset_loader=reinterpret_cast<sound4::CPresetLoader*>(userdata);
01311         std::string content=preset_loader->Read(filename);
01312         return strdup(content.c_str());
01313     }
01314     static void sound4cl_custom_reader_free(char *content, void* userdata) {
01315         if (!content) return;
01316         free(content);
01317     }
01318     static int sound4cl_custom_writer(const fs_char *filename, const char *content, void* userdata) {
01319         sound4::CPresetLoader *preset_loader=reinterpret_cast<sound4::CPresetLoader*>(userdata);
01320         auto res=preset_loader->Write(filename,content);
01321         return res?0:-1;
01322     }
01323     static int sound4cl_custom_exists(const fs_char *filename, void* userdata) {
01324         sound4::CPresetLoader *preset_loader=reinterpret_cast<sound4::CPresetLoader*>(userdata);
01325         auto res=preset_loader->Exists(filename);
01326         return res?0:-1;
01327     }
01328     static fs_char** sound4cl_custom_getall(void* userdata) {
01329         sound4::CPresetLoader *preset_loader=reinterpret_cast<sound4::CPresetLoader*>(userdata);
01330         auto res=preset_loader->GetAll();
01331         fs_char**all=(fs_char**)malloc((res.size()+1)*sizeof(const fs_char*));
01332         for (size_t n=0;n<res.size();n++) {
01333             all[n]=fs_strdup(res[n].c_str());
01334         }
01335         all[res.size()]=nullptr;
01336         return all;
01337     }
01338     static void sound4cl_custom_getall_free(fs_char** all, void* userdata) {
01339         if (!all) return;
01340         for (fs_char** one=all;*one;one++) {
01341             free(*one);
01342         }
```

```
01343         free(all);
01344     }
01345     static int sound4cl_custom_remove(const fs_char *filename, void* userdata) {
01346         sound4::CPresetLoader *preset_loader=reinterpret_cast<sound4::CPresetLoader*>(userdata);
01347         auto res=preset_loader->Remove(filename);
01348         return res?0:-1;
01349     }
01350     static int sound4cl_custom_rename(const fs_char *from, const fs_char *to, void* userdata) {
01351         sound4::CPresetLoader *preset_loader=reinterpret_cast<sound4::CPresetLoader*>(userdata);
01352         auto res=preset_loader->Rename(from,to);
01353         return res?0:-1;
01354     }
01355
01356 };
```

11.19 sound4cl_cdef.h File Reference

C function signature definition for all sound4.cl libraries.

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

Classes

- struct sound4cl_SOUND4_ProcessInfo

Macros

- #define fs_strdup wcsdup

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

Typedefs

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

Enumerations

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

Functions

- const struct sound4cl_SOUND4_ProcessInfo * sound4cl_SOUND4_GetProcessInfo ()
- const char * sound4cl_GetVersion ()
- unsigned int sound4cl_GetChunkSizeInFrames ()
- unsigned int * sound4cl_GetPossibleChunkSizeInFrames ()
- unsigned int sound4cl_GetChannelCount ()
- unsigned int sound4cl_GetAudioInputCount ()
- unsigned int sound4cl_GetAudioOutputCount ()
- unsigned int sound4cl_GetSampleRate ()
- void sound4cl_SetLoggerCallback (sound4cl_loggerfn logger)
- void sound4cl_SetLogSeverity (enum sound4cl_LogSeverity severity)
- struct sound4cl_CParameters * sound4cl_NewParameters ()
- void sound4cl_FreeParameters (struct sound4cl_CParameters *params)
- void sound4cl_SetParameter (struct sound4cl_CParameters *params, const char *name, const char *value)
- const char * sound4cl_GetParameter (struct sound4cl_CParameters *params, const char *name)
- void sound4cl_FreeParameterValue (const char *value)
- struct sound4cl_CInstance * sound4cl_InitProcess (const char *LoginKey, const char *RadioName, const char *Access_Key_ID, const char *Access_Key_Secret, const char *save_path)
- struct sound4cl_CInstance * sound4cl_InitProcess2 (const char *LoginKey, const char *RadioName, const char *Access_Key_ID, const char *Access_Key_Secret, const char *save_path, const struct sound4cl_↩ CParameters *parameters)
- struct sound4cl_CInstance * sound4cl_InitProcess3 (const char *LoginKey, const char *RadioName, const char *Access_Key_ID, const char *Access_Key_Secret, const char *save_path, const struct sound4cl_↩ CParameters *parameters, unsigned int frames_per_chunk)
- unsigned int sound4cl_GetProcessChunkFrames (struct sound4cl_CInstance *instance)
- int sound4cl_TerminateProcess (struct sound4cl_CInstance *instance)
- void sound4cl_ExitProcess (struct sound4cl_CInstance *instance)
- struct sound4cl_CBus * sound4cl_NewBus ()
- void sound4cl_FreeBus (struct sound4cl_CBus *bus)
- void sound4cl_SetInstanceBus (struct sound4cl_CParameters *params, struct sound4cl_CBus *bus)
- void sound4cl_SetPresetManager (struct sound4cl_CParameters *params, sound4cl_storage_reader reader, sound4cl_storage_reader_free, sound4cl_storage_writer writer, sound4cl_storage_exists exists, sound4cl_storage_getall getall, sound4cl_storage_getall_free getall_free, sound4cl_storage_remove remove, sound4cl_storage_rename rename, int IsReadOnly, void *userdata)
- void sound4cl_PresetManager_InformChange (struct sound4cl_CInstance *instance, const fs_char *relative_path, enum sound4cl_PresetChange_Kind change_kind)
- void sound4cl_SetMetadata (struct sound4cl_CInstance *instance, const char *key, const char *value)
- void sound4cl_SetMetadataMulti (struct sound4cl_CInstance *instance, const char **keyvalue)
- const char ** sound4cl_GetMetadataInfos (struct sound4cl_CInstance *instance)
- void sound4cl_FreeMetadataInfos (struct sound4cl_CInstance *instance, const char **infos)

- void sound4cl_StartUpdateThread (struct sound4cl_CInstance *instance, unsigned int port)
- void sound4cl_StopUpdateThread (struct sound4cl_CInstance *instance)
- int sound4cl_WaitUpdateThreadReady (struct sound4cl_CInstance *instance, int milliseconds)
- void sound4cl_ProcessAudio (struct sound4cl_CInstance *instance, const float *input, float *output)
- void sound4cl_ProcessAudio_Planar (struct sound4cl_CInstance *instance, float const *const *input, float *const *output)
- float * sound4cl_GetBufferIn (struct sound4cl_CInstance *instance)
- float * sound4cl_GetBufferOut (struct sound4cl_CInstance *instance)
- unsigned int sound4cl_GetEstimatedDelay (struct sound4cl_CInstance *instance)
- const char * sound4cl_GetFormatName (const enum sound4cl_SampleFormat fmt)
- enum sound4cl_SampleFormat sound4cl_GetFormatFromName (const char *name)
- unsigned int sound4cl_GetBytesFromFormat (const enum sound4cl_SampleFormat fmt)
- unsigned int sound4cl_GetMaxPacketFrame (struct sound4cl_CInstance *instance)
- unsigned int sound4cl_AddAudio (struct sound4cl_CInstance *instance, const uint8_t *payload, unsigned int nFrame, enum sound4cl_SampleFormat fmt)
- unsigned int sound4cl_AddPadAudio (struct sound4cl_CInstance *instance)
- unsigned int sound4cl_GetOutputCount (struct sound4cl_CInstance *instance)
- unsigned int sound4cl_GetAudio (struct sound4cl_CInstance *instance, uint8_t *payload, unsigned int max↩ _nFrame, enum sound4cl_SampleFormat fmt)
- void sound4cl_AudioConvertFrom (const uint8_t *payload, float *output, size_t nSpl, enum sound4cl_SampleFormat fmt)
- void sound4cl_AudioConvertTo (const float *input, uint8_t *payload, size_t nSpl, enum sound4cl_SampleFormat fmt)
- void sound4cl_StereoToMono (const float *input, float *output, size_t nFrame)
- void sound4cl_MonoToStereo (const float *input, float *output, size_t nFrame)
- void sound4cl_StereoToMono_Planar (const float *inputL, const float *inputR, float *output, size_t nFrame)
- void sound4cl_MonoToStereo_Planar (const float *input, float *outputL, float *outputR, size_t nFrame)
- void sound4cl_AudioMonoFromLiveStereo (const uint8_t *payload, float *output)
- void sound4cl_AudioMonoToLiveStereo (const float *input, uint8_t *payload)
- struct sound4cl_CClientInstance * sound4cl_NewClient (struct sound4cl_CInstance *instance)
- void sound4cl_DeleteClient (struct sound4cl_CClientInstance *client)
- const char * sound4cl_ProcessJson (struct sound4cl_CClientInstance *client, const char *json_str, int *need_save)
- void sound4cl_FreeJsonAnswer (const char *json_str)
- int sound4cl_SaveState (struct sound4cl_CInstance *instance)
- uint64_t sound4cl_Webserver_tcp (unsigned int listenport, unsigned int listenport_secure, const char *socket_ip, unsigned int socket_port)
- uint64_t sound4cl_Webserver_tcp2 (unsigned int listenport, unsigned int listenport_secure, const char *socket_ip, unsigned int socket_port, const struct sound4cl_CParameters *parameters)
- uint64_t sound4cl_Webserver (unsigned int listenport, unsigned int listenport_secure, struct sound4cl_↩ CInstance *instance)
- int sound4cl_Webserver_Stop (uint64_t id, int timeout_ms)
- int sound4cl_Webserver_Status (uint64_t id)
- void sound4cl_Webserver_SetAppHealth (struct sound4cl_CInstance *instance, int httpcode, const char *contenttype, const char *content)
- void sound4cl_Webserver_GetAppHealth (struct sound4cl_CInstance *instance, int *httpcode, char **contenttype, char **content)
- void sound4cl_Webserver_FreeString (char *str)

11.19.1 Macro Definition Documentation

11.19.1.1 fs_strdup

```
#define fs_strdup wcsdup
```

Definition at line 17 of file sound4cl_cdef.h.

11.19.1.2 SOUND4_INVALID_WEBSERVER_ID

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

Definition at line 62 of file sound4cl_cdef.h.

11.19.1.3 SOUND4_WEBSERVER_HTTP_OK

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

Definition at line 63 of file sound4cl_cdef.h.

11.19.1.4 SOUND4_WEBSERVER_HTTPS_OK

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

Definition at line 64 of file sound4cl_cdef.h.

11.19.2 Typedef Documentation

11.19.2.1 fs_char

```
typedef wchar_t fs_char
```

Definition at line 18 of file sound4cl_cdef.h.

11.19.2.2 sound4cl_loggerfn

```
typedef void(* sound4cl_loggerfn) (enum sound4cl_LogSeverity, const char *)
```

Definition at line 102 of file sound4cl_cdef.h.

11.19.2.3 sound4cl_storage_exists

```
typedef int(* sound4cl_storage_exists) (const fs_char *filename, void *userdata)
```

Definition at line 143 of file sound4cl_cdef.h.

11.19.2.4 sound4cl_storage_getall

```
typedef fs_char **(* sound4cl_storage_getall) (void *userdata)
```

Definition at line 144 of file sound4cl_cdef.h.

11.19.2.5 sound4cl_storage_getall_free

```
typedef void(* sound4cl_storage_getall_free) (fs_char **all, void *userdata)
```

Definition at line 145 of file sound4cl_cdef.h.

11.19.2.6 sound4cl_storage_reader

```
typedef char *(* sound4cl_storage_reader) (const fs_char *filename, void *userdata)
```

Definition at line 140 of file sound4cl_cdef.h.

11.19.2.7 sound4cl_storage_reader_free

```
typedef void(* sound4cl_storage_reader_free) (char *content, void *userdata)
```

Definition at line 141 of file sound4cl_cdef.h.

11.19.2.8 sound4cl_storage_remove

```
typedef int(* sound4cl_storage_remove) (const fs_char *filename, void *userdata)
```

Definition at line 146 of file sound4cl_cdef.h.

11.19.2.9 sound4cl_storage_rename

```
typedef int(* sound4cl_storage_rename) (const fs_char *from, const fs_char *to, void *userdata)
```

Definition at line 147 of file sound4cl_cdef.h.

11.19.2.10 sound4cl_storage_writer

```
typedef int(* sound4cl_storage_writer) (const fs_char *filename, const char *content, void
*userdata)
```

Definition at line 142 of file sound4cl_cdef.h.

11.19.3 Enumeration Type Documentation

11.19.3.1 sound4cl_LogSeverity

```
enum sound4cl_LogSeverity
```

Enumerator

sound4cl_none	
sound4cl_fatal	
sound4cl_error	
sound4cl_warning	
sound4cl_info	
sound4cl_verbose	
sound4cl_verbose2	
sound4cl_verbose3	
sound4cl_verbose4	
sound4cl_verbose5	

Definition at line 41 of file sound4cl_cdef.h.

11.19.3.2 sound4cl_PresetChange_Kind

```
enum sound4cl_PresetChange_Kind
```

Enumerator

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

Definition at line 57 of file sound4cl_cdef.h.

11.19.3.3 sound4cl_SampleFormat

```
enum sound4cl_SampleFormat
```

Supported formats for audio I/O

Enumerator

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

Definition at line 69 of file sound4cl_cdef.h.

11.19.4 Function Documentation

11.19.4.1 sound4cl_AddAudio()

```
unsigned int sound4cl_AddAudio (
            struct sound4cl_CInstance * instance,
            const uint8_t * payload,
            unsigned int nFrame,
            enum sound4cl_SampleFormat fmt )
```

11.19.4.2 sound4cl_AddPadAudio()

```
unsigned int sound4cl_AddPadAudio (
            struct sound4cl_CInstance * instance )
```

11.19.4.3 sound4cl_AudioConvertFrom()

```
void sound4cl_AudioConvertFrom (
            const uint8_t * payload,
            float * output,
            size_t nSpl,
            enum sound4cl_SampleFormat fmt )
```

11.19.4.4 sound4cl_AudioConvertTo()

```
void sound4cl_AudioConvertTo (
            const float * input,
            uint8_t * payload,
            size_t nSpl,
            enum sound4cl_SampleFormat fmt )
```

11.19.4.5 sound4cl_AudioMonoFromLiveStereo()

```
void sound4cl_AudioMonoFromLiveStereo (
            const uint8_t * payload,
            float * output )
```

11.19.4.6 sound4cl_AudioMonoToLiveStereo()

```
void sound4cl_AudioMonoToLiveStereo (
            const float * input,
            uint8_t * payload )
```

11.19.4.7 sound4cl_DeleteClient()

```
void sound4cl_DeleteClient (
            struct sound4cl_CClientInstance * client )
```

11.19.4.8 sound4cl_ExitProcess()

```
void sound4cl_ExitProcess (
            struct sound4cl_CInstance * instance )
```

11.19.4.9 sound4cl_FreeBus()

```
void sound4cl_FreeBus (
            struct sound4cl_CBus * bus )
```

11.19.4.10 sound4cl_FreeJsonAnswer()

```
void sound4cl_FreeJsonAnswer (
            const char * json_str )
```

11.19.4.11 sound4cl_FreeMetadataInfos()

```
void sound4cl_FreeMetadataInfos (
            struct sound4cl_CInstance * instance,
            const char ** infos )
```

11.19.4.12 sound4cl_FreeParameters()

```
void sound4cl_FreeParameters (
            struct sound4cl_CParameters * params )
```

11.19.4.13 sound4cl_FreeParameterValue()

```
void sound4cl_FreeParameterValue (
            const char * value )
```

11.19.4.14 sound4cl_GetAudio()

```
unsigned int sound4cl_GetAudio (
            struct sound4cl_CInstance * instance,
            uint8_t * payload,
            unsigned int max_nFrame,
            enum sound4cl_SampleFormat fmt )
```

11.19.4.15 sound4cl_GetAudioInputCount()

```
unsigned int sound4cl_GetAudioInputCount ( )
```

11.19.4.16 sound4cl_GetAudioOutputCount()

```
unsigned int sound4cl_GetAudioOutputCount ( )
```

11.19.4.17 sound4cl_GetBufferIn()

```
float * sound4cl_GetBufferIn (
            struct sound4cl_CInstance * instance )
```

11.19.4.18 sound4cl_GetBufferOut()

```
float * sound4cl_GetBufferOut (
            struct sound4cl_CInstance * instance )
```

11.19.4.19 sound4cl_GetBytesFromFormat()

```
unsigned int sound4cl_GetBytesFromFormat (
            const enum sound4cl_SampleFormat fmt )
```

11.19.4.20 sound4cl_GetChannelCount()

```
unsigned int sound4cl_GetChannelCount ( )
```

11.19.4.21 sound4cl_GetChunkSizeInFrames()

```
unsigned int sound4cl_GetChunkSizeInFrames ( )
```

11.19.4.22 sound4cl_GetEstimatedDelay()

```
unsigned int sound4cl_GetEstimatedDelay (
            struct sound4cl_CInstance * instance )
```

11.19.4.23 sound4cl_GetFormatFromName()

```
enum sound4cl_SampleFormat sound4cl_GetFormatFromName (
            const char * name )
```

11.19.4.24 sound4cl_GetFormatName()

```
const char * sound4cl_GetFormatName (
            const enum sound4cl_SampleFormat fmt )
```

11.19.4.25 sound4cl_GetMaxPacketFrame()

```
unsigned int sound4cl_GetMaxPacketFrame (
            struct sound4cl_CInstance * instance )
```

11.19.4.26 sound4cl_GetMetadataInfos()

```
const char ** sound4cl_GetMetadataInfos (
            struct sound4cl_CInstance * instance )
```

11.19.4.27 sound4cl_GetOutputCount()

```
unsigned int sound4cl_GetOutputCount (
            struct sound4cl_CInstance * instance )
```

11.19.4.28 sound4cl_GetParameter()

```
const char * sound4cl_GetParameter (
            struct sound4cl_CParameters * params,
            const char * name )
```

11.19.4.29 sound4cl_GetPossibleChunkSizeInFrames()

```
unsigned int * sound4cl_GetPossibleChunkSizeInFrames ( )
```

11.19.4.30 sound4cl_GetProcessChunkFrames()

```
unsigned int sound4cl_GetProcessChunkFrames (
            struct sound4cl_CInstance * instance )
```

11.19.4.31 sound4cl_GetSampleRate()

```
unsigned int sound4cl_GetSampleRate ( )
```

11.19.4.32 sound4cl_GetVersion()

```
const char * sound4cl_GetVersion ( )
```

11.19.4.33 sound4cl_InitProcess()

```
struct sound4cl_CInstance * sound4cl_InitProcess (
            const char * LoginKey,
            const char * RadioName,
            const char * Access_Key_ID,
            const char * Access_Key_Secret,
            const char * save_path )
```

11.19.4.34 sound4cl_InitProcess2()

```
struct sound4cl_CInstance * sound4cl_InitProcess2 (
            const char * LoginKey,
            const char * RadioName,
            const char * Access_Key_ID,
            const char * Access_Key_Secret,
            const char * save_path,
            const struct sound4cl_CParameters * parameters )
```

11.19.4.35 sound4cl_InitProcess3()

```
struct sound4cl_CInstance * sound4cl_InitProcess3 (
            const char * LoginKey,
            const char * RadioName,
            const char * Access_Key_ID,
            const char * Access_Key_Secret,
            const char * save_path,
            const struct sound4cl_CParameters * parameters,
            unsigned int frames_per_chunk )
```

11.19.4.36 sound4cl_MonoToStereo()

```
void sound4cl_MonoToStereo (
            const float * input,
            float * output,
            size_t nFrame )
```

11.19.4.37 sound4cl_MonoToStereo_Planar()

```
void sound4cl_MonoToStereo_Planar (
            const float * input,
            float * outputL,
            float * outputR,
            size_t nFrame )
```

11.19.4.38 sound4cl_NewBus()

```
struct sound4cl_CBus * sound4cl_NewBus ( )
```

11.19.4.39 sound4cl_NewClient()

```
struct sound4cl_CClientInstance * sound4cl_NewClient (
            struct sound4cl_CInstance * instance )
```

11.19.4.40 sound4cl_NewParameters()

```
struct sound4cl_CParameters * sound4cl_NewParameters ( )
```

11.19.4.41 sound4cl_PresetManager_InformChange()

```
void sound4cl_PresetManager_InformChange (
            struct sound4cl_CInstance * instance,
            const fs_char * relative_path,
            enum sound4cl_PresetChange_Kind change_kind )
```

11.19.4.42 sound4cl_ProcessAudio()

```
void sound4cl_ProcessAudio (
            struct sound4cl_CInstance * instance,
            const float * input,
            float * output )
```

11.19.4.43 sound4cl_ProcessAudio_Planar()

```
void sound4cl_ProcessAudio_Planar (
            struct sound4cl_CInstance * instance,
            float const *const * input,
            float *const * output )
```

11.19.4.44 sound4cl_ProcessJson()

```
const char * sound4cl_ProcessJson (
            struct sound4cl_CClientInstance * client,
            const char * json_str,
            int * need_save )
```

11.19.4.45 sound4cl_SaveState()

```
int sound4cl_SaveState (
            struct sound4cl_CInstance * instance )
```

11.19.4.46 sound4cl_SetInstanceBus()

```
void sound4cl_SetInstanceBus (
            struct sound4cl_CParameters * params,
            struct sound4cl_CBus * bus )
```

11.19.4.47 sound4cl_SetLoggerCallback()

```
void sound4cl_SetLoggerCallback (
            sound4cl_loggerfn logger )
```

11.19.4.48 sound4cl_SetLogSeverity()

```
void sound4cl_SetLogSeverity (
            enum sound4cl_LogSeverity severity )
```

11.19.4.49 sound4cl_SetMetadata()

```
void sound4cl_SetMetadata (
            struct sound4cl_CInstance * instance,
            const char * key,
            const char * value )
```

11.19.4.50 sound4cl_SetMetadataMulti()

```
void sound4cl_SetMetadataMulti (
            struct sound4cl_CInstance * instance,
            const char ** keyvalue )
```

11.19.4.51 sound4cl_SetParameter()

```
void sound4cl_SetParameter (
            struct sound4cl_CParameters * params,
            const char * name,
            const char * value )
```

11.19.4.52 sound4cl_SetPresetManager()

```
void sound4cl_SetPresetManager (
            struct sound4cl_CParameters * params,
            sound4cl_storage_reader reader,
            sound4cl_storage_reader_free ,
            sound4cl_storage_writer writer,
            sound4cl_storage_exists exists,
            sound4cl_storage_getall getall,
            sound4cl_storage_getall_free getall_free,
            sound4cl_storage_remove remove,
            sound4cl_storage_rename rename,
            int IsReadOnly,
            void * userdata )
```

11.19.4.53 sound4cl_SOUND4_GetProcessInfo()

```
const struct sound4cl_SOUND4_ProcessInfo * sound4cl_SOUND4_GetProcessInfo ( )
```

11.19.4.54 sound4cl_StartUpdateThread()

```
void sound4cl_StartUpdateThread (
            struct sound4cl_CInstance * instance,
            unsigned int port )
```

11.19.4.55 sound4cl_StereoToMono()

```
void sound4cl_StereoToMono (
            const float * input,
            float * output,
            size_t nFrame )
```

11.19.4.56 sound4cl_StereoToMono_Planar()

```
void sound4cl_StereoToMono_Planar (
            const float * inputL,
            const float * inputR,
            float * output,
            size_t nFrame )
```

11.19.4.57 sound4cl_StopUpdateThread()

```
void sound4cl_StopUpdateThread (
            struct sound4cl_CInstance * instance )
```

11.19.4.58 sound4cl_TerminateProcess()

```
int sound4cl_TerminateProcess (
            struct sound4cl_CInstance * instance )
```

11.19.4.59 sound4cl_WaitUpdateThreadReady()

```
int sound4cl_WaitUpdateThreadReady (
            struct sound4cl_CInstance * instance,
            int milliseconds )
```

11.19.4.60 sound4cl_Webserver()

```
uint64_t sound4cl_Webserver (
            unsigned int listenport,
            unsigned int listenport_secure,
            struct sound4cl_CInstance * instance )
```

11.19.4.61 sound4cl_Webserver_FreeString()

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

11.19.4.62 sound4cl_Webserver_GetAppHealth()

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

11.19.4.63 sound4cl_Webserver_SetAppHealth()

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

11.19.4.64 sound4cl_Webserver_Status()

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

11.19.4.65 sound4cl_Webserver_Stop()

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

11.19.4.66 sound4cl_Webserver_tcp()

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

11.19.4.67 sound4cl_Webserver_tcp2()

```
uint64_t sound4cl_Webserver_tcp2 (
            unsigned int listenport,
            unsigned int listenport_secure,
            const char * socket_ip,
            unsigned int socket_port,
            const struct sound4cl_CParameters * parameters )
```

11.20 sound4cl_cdef.h

Go to the documentation of this file.

```
00001
00006 #pragma once
00007
00008 #include <stdint.h>
00009 #ifdef __unix__
00010     #include <sys/types.h>
00011 #endif
00012
00013 #ifdef _WIN32
00014     #include <stringapiset.h>
00015     #include <wchar.h>
00017     #define fs_strdup wcsdup
00018     typedef wchar_t fs_char;
00019     #ifdef _MSC_VER
00020         #define __BYTE_ORDER__  REG_DWORD
00021         #define __ORDER_LITTLE_ENDIAN__ REG_DWORD_LITTLE_ENDIAN
00022         #define __ORDER_BIG_ENDIAN__    REG_DWORD_BIG_ENDIAN
00023     #endif
00024 #else // !_WIN32
00026     #define fs_strdup strdup
00027     typedef char fs_char;
00028 #endif // !_WIN32
00029
00030 #ifdef __cplusplus
00031 extern "C" {
00032 #endif // __cplusplus
00033
00034 struct sound4cl_SOUND4_ProcessInfo {
00035     size_t struct_size;
00036     const char *name;
00037     const char *prefix;
00038     const char *shortname;
00039     const char *version;
00040 };
00041 enum sound4cl_LogSeverity {
00042     sound4cl_none = 0,
00043     sound4cl_fatal = 1,
00044     sound4cl_error = 2,
00045     sound4cl_warning = 3,
00046     sound4cl_info = 4,
00047     sound4cl_verbose = 5,
00048     sound4cl_verbose2 = 6,
00049     sound4cl_verbose3 = 7,
00050     sound4cl_verbose4 = 8,
00051     sound4cl_verbose5 = 9,
00052 };
00053 struct sound4cl_CInstance;
00054 struct sound4cl_CParameters;
00055 struct sound4cl_CBus;
00056 struct sound4cl_CClientInstance;
00057 enum sound4cl_PresetChange_Kind {
00058     change_kind_created = 1,
00059     change_kind_modified   ,
00060     change_kind_deleted    ,
00061 };
00062 #define SOUND4_INVALID_WEBSERVER_ID            ((uint64_t)-1)
00063 #define SOUND4_WEBSERVER_HTTP_OK               (1<<0)
00064 #define SOUND4_WEBSERVER_HTTPS_OK              (1<<1)
00065
00069 enum sound4cl_SampleFormat {
00070     INVALID_FORMAT=0,
00071     S16_LE,
00072     S16_BE,
00073     S24_LE,
00074     S24_BE,
00075     S32_LE,
00076     S32_BE,
00077     F32_LE,
00078     F32_BE,
00079
00080     // Native definitions, only links to the physical ones
00081     #if __BYTE_ORDER__ == __ORDER_LITTLE_ENDIAN__
00082         S16_NATIVE=S16_LE,
00083         S24_NATIVE=S24_LE,
00084         S32_NATIVE=S32_LE,
00085         F32_NATIVE=F32_LE,
00086     #else
00087         S16_NATIVE=S16_BE,
00088         S24_NATIVE=S24_BE,
00089         S32_NATIVE=S32_BE,
00090         F32_NATIVE=F32_BE,
00091     #endif
```

```
00092 };
00093
00094 const struct sound4cl_SOUND4_ProcessInfo* sound4cl_SOUND4_GetProcessInfo();
00095 const char *sound4cl_GetVersion();
00096 unsigned int sound4cl_GetChunkSizeInFrames();
00097 unsigned int* sound4cl_GetPossibleChunkSizeInFrames();
00098 unsigned int sound4cl_GetChannelCount();
00099 unsigned int sound4cl_GetAudioInputCount();
00100 unsigned int sound4cl_GetAudioOutputCount();
00101 unsigned int sound4cl_GetSampleRate();
00102 typedef void (*sound4cl_loggerfn) (enum sound4cl_LogSeverity, const char *);
00103 void sound4cl_SetLoggerCallback( sound4cl_loggerfn logger );
00104 void sound4cl_SetLogSeverity( enum sound4cl_LogSeverity severity );
00105 struct sound4cl_CParameters *sound4cl_NewParameters();
00106 void sound4cl_FreeParameters(struct sound4cl_CParameters *params);
00107 void sound4cl_SetParameter(struct sound4cl_CParameters *params, const char *name, const char *value);
00108 const char *sound4cl_GetParameter(struct sound4cl_CParameters *params, const char *name);
00109 void sound4cl_FreeParameterValue(const char *value);
00110 struct sound4cl_CInstance *sound4cl_InitProcess(
00111         const char *LoginKey,
00112         const char *RadioName,
00113         const char *Access_Key_ID,
00114         const char *Access_Key_Secret,
00115         const char *save_path
00116     );
00117 struct sound4cl_CInstance *sound4cl_InitProcess2(
00118         const char *LoginKey,
00119         const char *RadioName,
00120         const char *Access_Key_ID,
00121         const char *Access_Key_Secret,
00122         const char *save_path,
00123         const struct sound4cl_CParameters *parameters
00124     );
00125 struct sound4cl_CInstance *sound4cl_InitProcess3(
00126         const char *LoginKey,
00127         const char *RadioName,
00128         const char *Access_Key_ID,
00129         const char *Access_Key_Secret,
00130         const char *save_path,
00131         const struct sound4cl_CParameters *parameters,
00132         unsigned int frames_per_chunk
00133     );
00134 unsigned int sound4cl_GetProcessChunkFrames(struct sound4cl_CInstance *instance);
00135 int sound4cl_TerminateProcess(struct sound4cl_CInstance *instance);
00136 void sound4cl_ExitProcess(struct sound4cl_CInstance *instance);
00137 struct sound4cl_CBus* sound4cl_NewBus();
00138 void sound4cl_FreeBus(struct sound4cl_CBus *bus);
00139 void sound4cl_SetInstanceBus(struct sound4cl_CParameters *params, struct sound4cl_CBus *bus);
00140 typedef char * (*sound4cl_storage_reader) (const fs_char *filename, void* userdata);
00141 typedef void (*sound4cl_storage_reader_free) (char *content, void* userdata);
00142 typedef int (*sound4cl_storage_writer) (const fs_char *filename, const char *content, void* userdata);
00143 typedef int (*sound4cl_storage_exists) (const fs_char *filename, void* userdata);
00144 typedef fs_char** (*sound4cl_storage_getall) (void* userdata);
00145 typedef void (*sound4cl_storage_getall_free) (fs_char**all, void* userdata);
00146 typedef int (*sound4cl_storage_remove) (const fs_char *filename, void* userdata);
00147 typedef int (*sound4cl_storage_rename) (const fs_char *from, const fs_char *to, void* userdata);
00148 void sound4cl_SetPresetManager(struct sound4cl_CParameters *params
00149     , sound4cl_storage_reader reader, sound4cl_storage_reader_free
00150     , sound4cl_storage_writer writer
00151     , sound4cl_storage_exists exists
00152     , sound4cl_storage_getall getall, sound4cl_storage_getall_free getall_free
00153     , sound4cl_storage_remove remove
00154     , sound4cl_storage_rename rename
00155     , int IsReadOnly
00156     , void* userdata
00157     );
00158 void sound4cl_PresetManager_InformChange(struct sound4cl_CInstance *instance, const fs_char
      *relative_path, enum sound4cl_PresetChange_Kind change_kind);
00159 void sound4cl_SetMetadata(struct sound4cl_CInstance *instance, const char* key, const char* value);
00160 void sound4cl_SetMetadataMulti(struct sound4cl_CInstance *instance, const char** keyvalue);
00161 const char** sound4cl_GetMetadataInfos(struct sound4cl_CInstance *instance);
00162 void sound4cl_FreeMetadataInfos(struct sound4cl_CInstance *instance, const char** infos);
00163 void sound4cl_StartUpdateThread(struct sound4cl_CInstance *instance, unsigned int port);
00164 void sound4cl_StopUpdateThread(struct sound4cl_CInstance *instance);
00165 int sound4cl_WaitUpdateThreadReady(struct sound4cl_CInstance *instance, int milliseconds);
00166 void sound4cl_ProcessAudio(struct sound4cl_CInstance *instance, const float *input, float *output );
00167 void sound4cl_ProcessAudio_Planar(struct sound4cl_CInstance *instance
00168     , float const * const *input
00169     , float * const *output);
00170 float *sound4cl_GetBufferIn(struct sound4cl_CInstance *instance);
00171 float *sound4cl_GetBufferOut(struct sound4cl_CInstance *instance);
00172 unsigned int sound4cl_GetEstimatedDelay(struct sound4cl_CInstance *instance);
00173 const char *sound4cl_GetFormatName(const enum sound4cl_SampleFormat fmt);
00174 enum sound4cl_SampleFormat sound4cl_GetFormatFromName(const char *name);
00175 unsigned int sound4cl_GetBytesFromFormat(const enum sound4cl_SampleFormat fmt);
00176 unsigned int sound4cl_GetMaxPacketFrame(struct sound4cl_CInstance *instance);
00177 unsigned int sound4cl_AddAudio(struct sound4cl_CInstance *instance, const uint8_t *payload, unsigned
```

```
      int nFrame, enum sound4cl_SampleFormat fmt );
00178 unsigned int sound4cl_AddPadAudio(struct sound4cl_CInstance *instance);
00179 unsigned int sound4cl_GetOutputCount(struct sound4cl_CInstance *instance);
00180 unsigned int sound4cl_GetAudio(struct sound4cl_CInstance *instance, uint8_t *payload, unsigned int
      max_nFrame, enum sound4cl_SampleFormat fmt );
00181 void sound4cl_AudioConvertFrom(const uint8_t *payload, float *output, size_t nSpl, enum
      sound4cl_SampleFormat fmt);
00182 void sound4cl_AudioConvertTo(const float *input, uint8_t *payload, size_t nSpl, enum
      sound4cl_SampleFormat fmt);
00183 void sound4cl_StereoToMono(const float *input, float *output, size_t nFrame);
00184 void sound4cl_MonoToStereo(const float *input, float *output, size_t nFrame);
00185 void sound4cl_StereoToMono_Planar(const float *inputL, const float *inputR, float *output, size_t
      nFrame);
00186 void sound4cl_MonoToStereo_Planar(const float *input, float *outputL, float *outputR, size_t nFrame);
00187 void sound4cl_AudioMonoFromLiveStereo(const uint8_t *payload, float *output);
00188 void sound4cl_AudioMonoToLiveStereo(const float *input, uint8_t *payload);
00189 struct sound4cl_CClientInstance *sound4cl_NewClient(struct sound4cl_CInstance *instance);
00190 void sound4cl_DeleteClient(struct sound4cl_CClientInstance *client);
00191 const char *sound4cl_ProcessJson(struct sound4cl_CClientInstance *client, const char *json_str, int
      *need_save);
00192 void sound4cl_FreeJsonAnswer(const char *json_str);
00193 int sound4cl_SaveState(struct sound4cl_CInstance *instance);
00194 uint64_t sound4cl_Webserver_tcp(unsigned int listenport, unsigned int listenport_secure, const char
      *socket_ip, unsigned int socket_port);
00195 uint64_t sound4cl_Webserver_tcp2(unsigned int listenport, unsigned int listenport_secure, const char
      *socket_ip, unsigned int socket_port, const struct sound4cl_CParameters *parameters);
00196 uint64_t sound4cl_Webserver(unsigned int listenport, unsigned int listenport_secure, struct
      sound4cl_CInstance *instance);
00197 int sound4cl_Webserver_Stop(uint64_t id, int timeout_ms);
00198 int sound4cl_Webserver_Status(uint64_t id);
00199 void sound4cl_Webserver_SetAppHealth(struct sound4cl_CInstance *instance, int httpcode, const char*
      contenttype, const char* content);
00200 void sound4cl_Webserver_GetAppHealth(struct sound4cl_CInstance *instance, int* httpcode, char**
      contenttype, char** content);
00201 void sound4cl_Webserver_FreeString(char* str);
00202
00203 #ifdef __cplusplus
00204 } // extern "C"
00205 #endif // __cplusplus
```

Index

~CBus
sound4::bigvoice::CBus, 84
sound4::bigvoice::dyn::CBus, 85
sound4::CBus, 86
~CClient
sound4::bigvoice::CInstance::CClient, 88
sound4::bigvoice::dyn::CInstance::CClient, 89
sound4::CInstance::CClient, 91
~CDynLoader
sound4::bigvoice::dyn::helper::CDynLoader, 105
sound4::helper::CDynLoader, 107
~CInstance
sound4::bigvoice::CInstance, 110
sound4::bigvoice::dyn::CInstance, 120
sound4::CInstance, 130
~CPresetLoader
sound4::bigvoice::CPresetLoader, 139
sound4::bigvoice::dyn::CPresetLoader, 140
sound4::CPresetLoader, 142
~MyPresetManager
MyPresetManager, 166

AddAudio
sound4::bigvoice::CInstance, 110
sound4::bigvoice::dyn::CDynLib, 94
sound4::bigvoice::dyn::CInstance, 120
sound4::CInstance, 130
AddPadAudio
sound4::bigvoice::dyn::CDynLib, 94
Audio convert helpers, 48
bigvoice_AudioConvertFrom, 49
bigvoice_AudioConvertTo, 49
bigvoice_AudioMonoFromLiveStereo, 49
bigvoice_AudioMonoToLiveStereo, 50
bigvoice_MonoToStereo, 50
bigvoice_MonoToStereo_Planar, 50
bigvoice_StereoToMono, 51
bigvoice_StereoToMono_Planar, 51
Audio managing (any audio), 44
bigvoice_AddAudio, 45
bigvoice_AddPadAudio, 46
bigvoice_GetAudio, 46
bigvoice_GetBytesFromFormat, 47
bigvoice_GetFormatFromName, 47
bigvoice_GetFormatName, 47
bigvoice_GetMaxPacketFrame, 48
bigvoice_GetOutputCount, 48
bigvoice_SampleFormat, 45
F32_BE, 45
F32_LE, 45
F32_NATIVE, 45
INVALID_FORMAT, 45
S16_BE, 45
S16_LE, 45
S16_NATIVE, 45
S24_BE, 45
S24_LE, 45
S24_NATIVE, 45
S32_BE, 45
S32_LE, 45
S32_NATIVE, 45
AudioConvertFrom
sound4::bigvoice::dyn::CDynLib, 94
sound4::CProcessor, 146
AudioConvertTo
sound4::bigvoice::dyn::CDynLib, 95
sound4::CProcessor, 146
AudioMonoFromLiveStereo
sound4::bigvoice::dyn::CDynLib, 95
sound4::CProcessor, 147
AudioMonoToLiveStereo
sound4::bigvoice::dyn::CDynLib, 95
sound4::CProcessor, 147

bigvoice_AddAudio
Audio managing (any audio), 45
bigvoice_AddPadAudio
Audio managing (any audio), 46
BIGVOICE_AUDIO_INPUT_COUNT
Global setup, 21
BIGVOICE_AUDIO_OUTPUT_COUNT
Global setup, 21
bigvoice_AudioConvertFrom
Audio convert helpers, 49
bigvoice_AudioConvertTo
Audio convert helpers, 49
BIGVOICE_AUDIOFRAME_COUNT
Global setup, 21
bigvoice_AudioMonoFromLiveStereo
Audio convert helpers, 49
bigvoice_AudioMonoToLiveStereo
Audio convert helpers, 50
bigvoice_CBus, 83
bigvoice_CClientInstance, 83
BIGVOICE_CHANNEL_COUNT
Global setup, 22
bigvoice_CInstance, 83
bigvoice_CParameters, 84
BIGVOICE_DECL
sound4.bigvoice.cl.h, 202

bigvoice_DeleteClient
Client's JSon config interfacing, 52
bigvoice_ExitProcess
Process creation/destruction, 27
bigvoice_FreeBus
Instance's bus, 32
bigvoice_FreeJsonAnswer
Client's JSon config interfacing, 52
bigvoice_FreeMetadataInfos
Metadata Management, 39
bigvoice_FreeParameters
Process creation/destruction, 28
bigvoice_FreeParameterValue
Process creation/destruction, 28
bigvoice_GetAudio
Audio managing (any audio), 46
bigvoice_GetAudioInputCount
Global setup, 23
bigvoice_GetAudioOutputCount
Global setup, 23
bigvoice_GetBufferIn
Direct processing, 42
bigvoice_GetBufferOut
Direct processing, 42
bigvoice_GetBytesFromFormat
Audio managing (any audio), 47
bigvoice_GetChannelCount
Global setup, 23
bigvoice_GetChunkSizeInFrames
Global setup, 24
bigvoice_GetEstimatedDelay
Direct processing, 43
bigvoice_GetFormatFromName
Audio managing (any audio), 47
bigvoice_GetFormatName
Audio managing (any audio), 47
bigvoice_GetMaxPacketFrame
Audio managing (any audio), 48
bigvoice_GetMetadataInfos
Metadata Management, 39
bigvoice_GetOutputCount
Audio managing (any audio), 48
bigvoice_GetParameter
Process creation/destruction, 28
bigvoice_GetPossibleChunkSizeInFrames
Global setup, 24
bigvoice_GetProcessChunkFrames
Process creation/destruction, 28
bigvoice_GetSampleRate
Global setup, 24
bigvoice_GetVersion
Global setup, 25
BIGVOICE_HAS_CLOUDBUS
sound4.bigvoice.cl.h, 202
BIGVOICE_HAS_WEBSERVER
sound4.bigvoice.cl.h, 203
bigvoice_InitProcess
Process creation/destruction, 29
bigvoice_InitProcess2
Process creation/destruction, 29
bigvoice_InitProcess3
Process creation/destruction, 29
bigvoice_loggerfn
Global setup, 22
bigvoice_LogSeverity
Global setup, 22
bigvoice_MonoToStereo
Audio convert helpers, 50
bigvoice_MonoToStereo_Planar
Audio convert helpers, 50
bigvoice_NewBus
Instance's bus, 32
bigvoice_NewClient
Client's JSon config interfacing, 53
bigvoice_NewParameters
Process creation/destruction, 31
bigvoice_PresetChange_Kind
Custom preset managagement, 37
bigvoice_PresetManager_InformChange
Custom preset managagement, 37
bigvoice_ProcessAudio
Direct processing, 43
bigvoice_ProcessAudio_Planar
Direct processing, 44
bigvoice_ProcessJson
Client's JSon config interfacing, 53
BIGVOICE_SAMPLE_RATE
Global setup, 22
bigvoice_SampleFormat
Audio managing (any audio), 45
bigvoice_SaveState
Client's JSon config interfacing, 54
bigvoice_SetInstanceBus
Instance's bus, 33
bigvoice_SetLoggerCallback
Global setup, 25
bigvoice_SetLogSeverity
Global setup, 25
bigvoice_SetMetadata
Metadata Management, 39
bigvoice_SetMetadataMulti
Metadata Management, 40
bigvoice_SetParameter
Process creation/destruction, 31
bigvoice_SetPresetManager
Custom preset managagement, 38
bigvoice_StartUpdateThread
Update Management, 41
bigvoice_StereoToMono
Audio convert helpers, 51
bigvoice_StereoToMono_Planar
Audio convert helpers, 51
bigvoice_StopUpdateThread
Update Management, 41
bigvoice_storage_exists
Custom preset managagement, 34

bigvoice_storage_getall
Custom preset managagement, 34
bigvoice_storage_getall_free
Custom preset managagement, 35
bigvoice_storage_reader
Custom preset managagement, 35
bigvoice_storage_reader_free
Custom preset managagement, 35
bigvoice_storage_remove
Custom preset managagement, 36
bigvoice_storage_rename
Custom preset managagement, 36
bigvoice_storage_writer
Custom preset managagement, 36
bigvoice_TerminateProcess
Process creation/destruction, 31
bigvoice_WaitUpdateThreadReady
Update Management, 41
bigvoice_Webserver
Web server, 62
bigvoice_Webserver_FreeString
Web server, 62
bigvoice_Webserver_GetAppHealth
Web server, 63
bigvoice_Webserver_SetAppHealth
Web server, 63
bigvoice_Webserver_Status
Web server, 63
bigvoice_Webserver_Stop
Web server, 64
bigvoice_Webserver_tcp
Web server, 64
bigvoice_Webserver_tcp2
Web server, 65

CBus
sound4::bigvoice::CBus, 84
sound4::bigvoice::dyn::CBus, 85
sound4::CBus, 86
sound4::CProcessor, 152
CClient
sound4::bigvoice::CInstance::CClient, 87
sound4::bigvoice::dyn::CInstance::CClient, 89
sound4::CInstance::CClient, 91
CDynLib
sound4::bigvoice::dyn::CDynLib, 93
CDynLoader
sound4::bigvoice::dyn::helper::CDynLoader, 104, 105
sound4::helper::CDynLoader, 107
change_kind_created
Custom preset managagement, 37
sound4cl_cdef.h, 251
change_kind_deleted
Custom preset managagement, 37
sound4cl_cdef.h, 251
change_kind_modified
Custom preset managagement, 37
sound4cl_cdef.h, 251
CHANNEL_COUNT
sound4.bigvoice.cl-pipe.cpp, 190
CHUNK_SIZE
sound4.bigvoice.cl-pipe.cpp, 190
ChunkFrames
sound4::bigvoice, 77
sound4::bigvoice::dyn, 79
CInstance
sound4::bigvoice::CInstance, 110
sound4::bigvoice::dyn::CInstance, 120
sound4::CInstance, 130
sound4::CProcessor, 152
Client's JSon config interfacing, 52
bigvoice_DeleteClient, 52
bigvoice_FreeJsonAnswer, 52
bigvoice_NewClient, 53
bigvoice_ProcessJson, 53
bigvoice_SaveState, 54
Close
sound4::bigvoice::dyn::helper::CDynLoader, 105
sound4::helper::CDynLoader, 107
CPresetLoader
sound4::bigvoice::CPresetLoader, 139
sound4::bigvoice::dyn::CPresetLoader, 140
sound4::CPresetLoader, 142
CProcessor
sound4::CProcessor, 146
Create
sound4::bigvoice::CInstance, 110
sound4::bigvoice::dyn::CInstance, 120
sound4::CInstance, 130
Custom preset managagement, 33
bigvoice_PresetChange_Kind, 37
bigvoice_PresetManager_InformChange, 37
bigvoice_SetPresetManager, 38
bigvoice_storage_exists, 34
bigvoice_storage_getall, 34
bigvoice_storage_getall_free, 35
bigvoice_storage_reader, 35
bigvoice_storage_reader_free, 35
bigvoice_storage_remove, 36
bigvoice_storage_rename, 36
bigvoice_storage_writer, 36
change_kind_created, 37
change_kind_deleted, 37
change_kind_modified, 37
fs_char, 37

DeleteClient
sound4::bigvoice::dyn::CDynLib, 95
Deprecated List, 3
Direct processing, 42
bigvoice_GetBufferIn, 42
bigvoice_GetBufferOut, 42
bigvoice_GetEstimatedDelay, 43
bigvoice_ProcessAudio, 43
bigvoice_ProcessAudio_Planar, 44
documentation.h, 181
dyn, 73

DynFunc_t
sound4::bigvoice::dyn::helper, 81
sound4::helper, 82
DynFuncHolder
sound4::bigvoice::dyn::helper::DynFuncHolder< T >, 162
sound4::helper::DynFuncHolder< T >, 164
DynLib_t
sound4::bigvoice::dyn::helper, 81
sound4::helper, 82

Environment variables and parameters, 19
error
Global setup, 23
sound4, 74
Exists
MyPresetManager, 166
sound4::bigvoice::CPresetLoader, 139
sound4::bigvoice::dyn::CPresetLoader, 141
sound4::CPresetLoader, 143
ExitProcess
sound4::bigvoice::dyn::CDynLib, 95

F32_BE
Audio managing (any audio), 45
sound4cl_cdef.h, 251
F32_LE
Audio managing (any audio), 45
sound4cl_cdef.h, 251
F32_NATIVE
Audio managing (any audio), 45
sound4cl_cdef.h, 251
fatal
Global setup, 23
sound4, 74
fnAddAudio
sound4::CProcessor, 152
fnAddPadAudio
sound4::CProcessor, 152
fnAudioConvertFrom
sound4::CProcessor, 152
fnAudioConvertTo
sound4::CProcessor, 153
fnAudioMonoFromLiveStereo
sound4::CProcessor, 153
fnAudioMonoToLiveStereo
sound4::CProcessor, 153
fnDeleteClient
sound4::CProcessor, 153
fnExitProcess
sound4::CProcessor, 153
fnFreeBus
sound4::CProcessor, 153
fnFreeJsonAnswer
sound4::CProcessor, 153
fnFreeMetadataInfos
sound4::CProcessor, 154
fnFreeParameters
sound4::CProcessor, 154
fnFreeParameterValue
sound4::CProcessor, 154
fnGetAudio
sound4::CProcessor, 154
fnGetAudioInputCount
sound4::CProcessor, 154
fnGetAudioOutputCount
sound4::CProcessor, 154
fnGetBufferIn
sound4::CProcessor, 154
fnGetBufferOut
sound4::CProcessor, 155
fnGetBytesFromFormat
sound4::CProcessor, 155
fnGetChannelCount
sound4::CProcessor, 155
fnGetChunkSizeInFrames
sound4::CProcessor, 155
fnGetEstimatedDelay
sound4::CProcessor, 155
fnGetFormatFromName
sound4::CProcessor, 155
fnGetFormatName
sound4::CProcessor, 155
fnGetMaxPacketFrame
sound4::CProcessor, 156
fnGetMetadataInfos
sound4::CProcessor, 156
fnGetOutputCount
sound4::CProcessor, 156
fnGetParameter
sound4::CProcessor, 156
fnGetPossibleChunkSizeInFrames
sound4::CProcessor, 156
fnGetProcessChunkFrames
sound4::CProcessor, 156
fnGetSampleRate
sound4::CProcessor, 156
fnGetVersion
sound4::CProcessor, 157
fnInitProcess
sound4::CProcessor, 157
fnInitProcess2
sound4::CProcessor, 157
fnInitProcess3
sound4::CProcessor, 157
fnMonoToStereo
sound4::CProcessor, 157
fnMonoToStereo_Planar
sound4::CProcessor, 157
fnNewBus
sound4::CProcessor, 157
fnNewClient
sound4::CProcessor, 158
fnNewParameters
sound4::CProcessor, 158
fnPresetManager_InformChange
sound4::CProcessor, 158

fnProcessAudio
sound4::CProcessor, 158
fnProcessAudio_Planar
sound4::CProcessor, 158
fnProcessJson
sound4::CProcessor, 158
fnSaveState
sound4::CProcessor, 158
fnSetInstanceBus
sound4::CProcessor, 159
fnSetLoggerCallback
sound4::CProcessor, 159
fnSetLogSeverity
sound4::CProcessor, 159
fnSetMetadata
sound4::CProcessor, 159
fnSetMetadataMulti
sound4::CProcessor, 159
fnSetParameter
sound4::CProcessor, 159
fnSetPresetManager
sound4::CProcessor, 159
fnStartUpdateThread
sound4::CProcessor, 160
fnStereoToMono
sound4::CProcessor, 160
fnStereoToMono_Planar
sound4::CProcessor, 160
fnStopUpdateThread
sound4::CProcessor, 160
fnTerminateProcess
sound4::CProcessor, 160
fnWaitUpdateThreadReady
sound4::CProcessor, 160
fnWebserver
sound4::CProcessor, 160
fnWebserver_FreeString
sound4::CProcessor, 161
fnWebserver_GetAppHealth
sound4::CProcessor, 161
fnWebserver_SetAppHealth
sound4::CProcessor, 161
fnWebserver_Status
sound4::CProcessor, 161
fnWebserver_Stop
sound4::CProcessor, 161
fnWebserver_tcp
sound4::CProcessor, 161
fnWebserver_tcp2
sound4::CProcessor, 161
format
sound4::bigvoice::dyn::helper::SampleFormat< float >, 172
sound4::bigvoice::dyn::helper::SampleFormat< int16_t >, 173
sound4::bigvoice::dyn::helper::SampleFormat< int32_t >, 175
sound4::bigvoice::helper::SampleFormat< float >, 172
sound4::bigvoice::helper::SampleFormat< int16_t >, 174
sound4::bigvoice::helper::SampleFormat< int32_t >, 176
sound4::helper::SampleFormat< float >, 173
sound4::helper::SampleFormat< int16_t >, 174
sound4::helper::SampleFormat< int32_t >, 176
FreeBus
sound4::bigvoice::dyn::CDynLib, 95
FreeJsonAnswer
sound4::bigvoice::dyn::CDynLib, 95
FreeMetadataInfos
sound4::bigvoice::dyn::CDynLib, 96
FreeParameters
sound4::bigvoice::dyn::CDynLib, 96
FreeParameterValue
sound4::bigvoice::dyn::CDynLib, 96
fs_char
Custom preset managagement, 37
sound4cl_cdef.h, 249
fs_strdup
sound4.bigvoice.cl.hpp, 208
sound4.bigvoice.cl_dyn.hpp, 218
sound4cl_cdef.h, 248

Get
sound4::bigvoice::CBus, 85
sound4::bigvoice::dyn::CBus, 86
sound4::CBus, 87
GetAll
MyPresetManager, 166
sound4::bigvoice::CPresetLoader, 139
sound4::bigvoice::dyn::CPresetLoader, 141
sound4::CPresetLoader, 143
GetAudio
sound4::bigvoice::CInstance, 111
sound4::bigvoice::dyn::CDynLib, 96
sound4::bigvoice::dyn::CInstance, 121
sound4::CInstance, 131
GetAudioInputCount
sound4::bigvoice::dyn::CDynLib, 96
sound4::CProcessor, 147
GetAudioOutputCount
sound4::bigvoice::dyn::CDynLib, 96
sound4::CProcessor, 147
GetBufferIn
sound4::bigvoice::CInstance, 111
sound4::bigvoice::dyn::CDynLib, 96
sound4::bigvoice::dyn::CInstance, 121
sound4::CInstance, 131
GetBufferOut
sound4::bigvoice::CInstance, 111
sound4::bigvoice::dyn::CDynLib, 97
sound4::bigvoice::dyn::CInstance, 121
sound4::CInstance, 131
GetBytesFromFormat
sound4::bigvoice::dyn::CDynLib, 97

sound4::CProcessor, 147
GetChannelCount
sound4::bigvoice::dyn::CDynLib, 97
sound4::CProcessor, 148
GetChunkFrames
sound4::bigvoice::CInstance, 111
sound4::bigvoice::dyn::CInstance, 121
sound4::CInstance, 131
GetChunkSizeInFrames
sound4::bigvoice::dyn::CDynLib, 97
sound4::CProcessor, 148
GetEnv
sound4.bigvoice.cl-pipe-cxx.cpp, 182
sound4.bigvoice.cl-pipe-dyn.cpp, 185
sound4cl-pipe.cpp, 195
GetEstimatedDelay
sound4::bigvoice::CInstance, 112
sound4::bigvoice::dyn::CDynLib, 97
sound4::bigvoice::dyn::CInstance, 122
sound4::CInstance, 132
GetFormatFromName
sound4::bigvoice::dyn::CDynLib, 97
sound4::CProcessor, 148
GetFormatName
sound4::bigvoice::dyn::CDynLib, 97
sound4::CProcessor, 149
GetMaxPacketFrame
sound4::bigvoice::dyn::CDynLib, 98
GetMetadataInfos
sound4::bigvoice::CInstance, 112
sound4::bigvoice::dyn::CDynLib, 98
sound4::bigvoice::dyn::CInstance, 122
sound4::CInstance, 132
GetOutputCount
sound4::bigvoice::dyn::CDynLib, 98
GetParam
sound4::bigvoice::CInstance, 112
sound4::bigvoice::dyn::CInstance, 122
sound4::CInstance, 132
GetParameter
sound4::bigvoice::dyn::CDynLib, 98
GetPossibleChunkSizeInFrames
sound4::bigvoice::dyn::CDynLib, 98
sound4::CProcessor, 149
GetPrefixSymbol
sound4::CProcessor, 149
GetProcessChunkFrames
sound4::bigvoice::dyn::CDynLib, 98
GetSampleRate
sound4::bigvoice::dyn::CDynLib, 98
sound4::CProcessor, 150
GetSymbol
sound4::bigvoice::dyn::helper::CDynLoader, 105
sound4::helper::CDynLoader, 107
GetSymbol_safe
sound4::bigvoice::dyn::helper::CDynLoader, 105
sound4::helper::CDynLoader, 107
GetThisLibraryPath
sound4::bigvoice::dyn::helper::CDynLoader, 105
sound4::helper::CDynLoader, 108
GetVersion
sound4::bigvoice::dyn::CDynLib, 99
sound4::CProcessor, 150
GetWebServerAppHealth
sound4::bigvoice::CInstance, 112
sound4::bigvoice::dyn::CInstance, 122
sound4::CInstance, 132
Global setup, 20
BIGVOICE_AUDIO_INPUT_COUNT, 21
BIGVOICE_AUDIO_OUTPUT_COUNT, 21
BIGVOICE_AUDIOFRAME_COUNT, 21
BIGVOICE_CHANNEL_COUNT, 22
bigvoice_GetAudioInputCount, 23
bigvoice_GetAudioOutputCount, 23
bigvoice_GetChannelCount, 23
bigvoice_GetChunkSizeInFrames, 24
bigvoice_GetPossibleChunkSizeInFrames, 24
bigvoice_GetSampleRate, 24
bigvoice_GetVersion, 25
bigvoice_loggerfn, 22
bigvoice_LogSeverity, 22
BIGVOICE_SAMPLE_RATE, 22
bigvoice_SetLoggerCallback, 25
bigvoice_SetLogSeverity, 25
error, 23
fatal, 23
info, 23
none, 23
SOUND4_GetProcessInfo, 27
verbose, 23
verbose2, 23
verbose3, 23
verbose4, 23
verbose5, 23
warning, 23

info
Global setup, 23
sound4, 75
InitProcess
sound4::bigvoice::dyn::CDynLib, 99
InitProcess2
sound4::bigvoice::dyn::CDynLib, 99
InitProcess3
sound4::bigvoice::dyn::CDynLib, 99
InputChannels
sound4::bigvoice, 77
sound4::bigvoice::dyn, 79
InputSampleSize
sound4::bigvoice, 77
sound4::bigvoice::dyn, 80
Instance's bus, 32
bigvoice_FreeBus, 32
bigvoice_NewBus, 32
bigvoice_SetInstanceBus, 33
INVALID_FORMAT
Audio managing (any audio), 45

sound4cl_cdef.h, 251
IsOk
sound4::bigvoice::CInstance, 113
sound4::bigvoice::dyn::CDynLib, 94
sound4::bigvoice::dyn::CInstance, 123
sound4::bigvoice::dyn::helper::CDynLoader, 105
sound4::bigvoice::dyn::helper::DynFuncHolder< T >, 163
sound4::CInstance, 133
sound4::CProcessor, 150
sound4::helper::CDynLoader, 108
sound4::helper::DynFuncHolder< T >, 164
IsReadOnly
MyPresetManager, 167
sound4::bigvoice::CPresetLoader, 139
sound4::bigvoice::dyn::CPresetLoader, 141
sound4::CPresetLoader, 143

JSON syntax, 54
JSON syntax for internal setup, 57

Load
sound4::bigvoice::dyn::CDynLib, 94
sound4::bigvoice::dyn::helper::CDynLoader, 106
sound4::CProcessor, 150
sound4::helper::CDynLoader, 108
log_cb_t
sound4, 74
sound4::bigvoice, 76
sound4::bigvoice::dyn, 79
LogSeverity
sound4, 74
sound4::bigvoice, 76
sound4::bigvoice::dyn, 79

main
sound4.bigvoice.cl-pipe-cxx.cpp, 182
sound4.bigvoice.cl-pipe-dyn.cpp, 185
sound4.bigvoice.cl-pipe.cpp, 191
sound4cl-pipe.cpp, 195
Metadata Management, 38
bigvoice_FreeMetadataInfos, 39
bigvoice_GetMetadataInfos, 39
bigvoice_SetMetadata, 39
bigvoice_SetMetadataMulti, 40
MonoToStereo
sound4::bigvoice::dyn::CDynLib, 99
sound4::CProcessor, 151
MonoToStereo_Planar
sound4::bigvoice::dyn::CDynLib, 99
sound4::CProcessor, 151
MyLogger
sound4.bigvoice.cl-pipe-cxx.cpp, 182
sound4.bigvoice.cl-pipe-dyn.cpp, 186
sound4.bigvoice.cl-pipe.cpp, 191
MyPresetManager, 165
~MyPresetManager, 166
Exists, 166
GetAll, 166
IsReadOnly, 167
MyPresetManager, 166
Read, 167
Remove, 167
Rename, 168
SetupPath, 168
Write, 168

name
SOUND4_ProcessInfo, 177
sound4cl_SOUND4_ProcessInfo, 178
NewBus
sound4::bigvoice::dyn::CDynLib, 99
NewClient
sound4::bigvoice::CInstance, 113
sound4::bigvoice::dyn::CDynLib, 100
sound4::bigvoice::dyn::CInstance, 123
sound4::CInstance, 133
NewParameters
sound4::bigvoice::dyn::CDynLib, 100
none
Global setup, 23
sound4, 75

OnUpdateThreadStart
sound4::bigvoice::CInstance, 113
sound4::bigvoice::dyn::CInstance, 123
sound4::CInstance, 133
OnUpdateThreadStop
sound4::bigvoice::CInstance, 113
sound4::bigvoice::dyn::CInstance, 123
sound4::CInstance, 133
operator bool
sound4::bigvoice::dyn::helper::CDynLoader, 106
sound4::bigvoice::dyn::helper::DynFuncHolder< T >, 163
sound4::helper::CDynLoader, 108
sound4::helper::DynFuncHolder< T >, 164
operator T*
sound4::bigvoice::dyn::helper::DynFuncHolder< T >, 163
sound4::helper::DynFuncHolder< T >, 164
operator=
sound4::bigvoice::CInstance, 113
sound4::bigvoice::CInstance::CClient, 88
sound4::bigvoice::dyn::CInstance, 123
sound4::bigvoice::dyn::CInstance::CClient, 89
sound4::bigvoice::dyn::helper::CDynLoader, 106
sound4::CInstance, 133
sound4::CInstance::CClient, 91
sound4::helper::CDynLoader, 108
OutputChannels
sound4::bigvoice, 77
sound4::bigvoice::dyn, 80
OutputSampleSize
sound4::bigvoice, 78
sound4::bigvoice::dyn, 80

prefix

SOUND4_ProcessInfo, 177
sound4cl_SOUND4_ProcessInfo, 178
PresetChange_Kind
sound4, 74
sound4::bigvoice, 77
sound4::bigvoice::dyn, 79
PresetManager_InformChange
sound4::bigvoice::CInstance, 113
sound4::bigvoice::dyn::CDynLib, 100
sound4::bigvoice::dyn::CInstance, 123
sound4::CInstance, 133
Process creation/destruction, 27
bigvoice_ExitProcess, 27
bigvoice_FreeParameters, 28
bigvoice_FreeParameterValue, 28
bigvoice_GetParameter, 28
bigvoice_GetProcessChunkFrames, 28
bigvoice_InitProcess, 29
bigvoice_InitProcess2, 29
bigvoice_InitProcess3, 29
bigvoice_NewParameters, 31
bigvoice_SetParameter, 31
bigvoice_TerminateProcess, 31
ProcessAnyAudio
sound4::bigvoice::CInstance, 114
sound4::bigvoice::dyn::CInstance, 124
sound4::CInstance, 134
ProcessAudio
sound4::bigvoice::CInstance, 114
sound4::bigvoice::dyn::CDynLib, 100
sound4::bigvoice::dyn::CInstance, 124
sound4::CInstance, 134
ProcessAudio_Planar
sound4::bigvoice::CInstance, 114
sound4::bigvoice::dyn::CDynLib, 100
sound4::bigvoice::dyn::CInstance, 124
sound4::CInstance, 134
ProcessJson
sound4::bigvoice::CInstance::CClient, 88
sound4::bigvoice::dyn::CDynLib, 100
sound4::bigvoice::dyn::CInstance::CClient, 89
sound4::CInstance::CClient, 91
ProcessLoop
sound4.bigvoice.cl-pipe-cxx.cpp, 182
sound4.bigvoice.cl-pipe-dyn.cpp, 186
sound4.bigvoice.cl-pipe.cpp, 191
sound4cl-pipe.cpp, 196

Read
MyPresetManager, 167
sound4::bigvoice::CPresetLoader, 139
sound4::bigvoice::dyn::CPresetLoader, 141
sound4::CPresetLoader, 143
Remove
MyPresetManager, 167
sound4::bigvoice::CPresetLoader, 139
sound4::bigvoice::dyn::CPresetLoader, 141
sound4::CPresetLoader, 143
Rename
MyPresetManager, 168
sound4::bigvoice::CPresetLoader, 139
sound4::bigvoice::dyn::CPresetLoader, 141
sound4::CPresetLoader, 143

S16_BE
Audio managing (any audio), 45
sound4cl_cdef.h, 251
S16_LE
Audio managing (any audio), 45
sound4cl_cdef.h, 251
S16_NATIVE
Audio managing (any audio), 45
sound4cl_cdef.h, 251
S24_BE
Audio managing (any audio), 45
sound4cl_cdef.h, 251
S24_LE
Audio managing (any audio), 45
sound4cl_cdef.h, 251
S24_NATIVE
Audio managing (any audio), 45
sound4cl_cdef.h, 251
S32_BE
Audio managing (any audio), 45
sound4cl_cdef.h, 251
S32_LE
Audio managing (any audio), 45
sound4cl_cdef.h, 251
S32_NATIVE
Audio managing (any audio), 45
sound4cl_cdef.h, 251
Sample using C interface, 17
Sample using C++ interface wrapper, 66
Sample using C++ interface wrapper with dynamic library loading, 67
Sample using Generic C++ interface (multi-process compatible), 69
SAMPLE_RATE
sound4.bigvoice.cl-pipe.cpp, 191
SampleFormat
sound4::bigvoice, 77
sound4::bigvoice::dyn, 79
SampleRate
sound4::bigvoice, 78
sound4::bigvoice::dyn, 80
SaveState
sound4::bigvoice::dyn::CDynLib, 100
SetBus
sound4::bigvoice::CInstance, 115
sound4::bigvoice::dyn::CInstance, 125
sound4::CInstance, 135
SetInstanceBus
sound4::bigvoice::dyn::CDynLib, 101
SetLoggerCallback
sound4::bigvoice::dyn::CDynLib, 101
sound4::CProcessor, 151
SetLogSeverity
sound4::bigvoice::dyn::CDynLib, 101

sound4::CProcessor, 151
SetMetadata
sound4::bigvoice::CInstance, 115
sound4::bigvoice::dyn::CDynLib, 101
sound4::bigvoice::dyn::CInstance, 125
sound4::CInstance, 135
SetMetadataMulti
sound4::bigvoice::CInstance, 116
sound4::bigvoice::dyn::CDynLib, 101
sound4::bigvoice::dyn::CInstance, 126
sound4::CInstance, 136
SetParam
sound4::bigvoice::CInstance, 116
sound4::bigvoice::dyn::CInstance, 126
sound4::CInstance, 136
SetParameter
sound4::bigvoice::dyn::CDynLib, 101
SetPresetManager
sound4::bigvoice::CInstance, 116
sound4::bigvoice::dyn::CDynLib, 101
sound4::bigvoice::dyn::CInstance, 126
sound4::CInstance, 136
SetupPath
MyPresetManager, 168
SetWebServerAppHealth
sound4::bigvoice::CInstance, 117
sound4::bigvoice::dyn::CInstance, 127
sound4::CInstance, 137
shortname
SOUND4_ProcessInfo, 177
sound4cl_SOUND4_ProcessInfo, 178
sound4, 73
error, 74
fatal, 74
info, 75
log_cb_t, 74
LogSeverity, 74
none, 75
PresetChange_Kind, 74
verbose, 75
verbose2, 75
verbose3, 75
verbose4, 75
verbose5, 75
warning, 75
SOUND4 BIGVOICE.CL Library, 1
sound4.bigvoice.cl-pipe-cxx.cpp, 181, 183
GetEnv, 182
main, 182
MyLogger, 182
ProcessLoop, 182
Usage, 182
sound4.bigvoice.cl-pipe-dyn.cpp, 185, 187
GetEnv, 185
main, 185
MyLogger, 186
ProcessLoop, 186
Usage, 186
sound4.bigvoice.cl-pipe.cpp, 190, 192
CHANNEL_COUNT, 190
CHUNK_SIZE, 190
main, 191
MyLogger, 191
ProcessLoop, 191
SAMPLE_RATE, 191
Usage, 191
sound4.bigvoice.cl.h, 200, 203
BIGVOICE_DECL, 202
BIGVOICE_HAS_CLOUDBUS, 202
BIGVOICE_HAS_WEBSERVER, 203
sound4.bigvoice.cl.hpp, 207, 209
fs_strdup, 208
sound4.bigvoice.cl_dyn.hpp, 216, 218
fs_strdup, 218
UNIXLIKE, 218
sound4::bigvoice, 76
ChunkFrames, 77
InputChannels, 77
InputSampleSize, 77
log_cb_t, 76
LogSeverity, 76
OutputChannels, 77
OutputSampleSize, 78
PresetChange_Kind, 77
SampleFormat, 77
SampleRate, 78
sound4::bigvoice::CBus, 84
~CBus, 84
CBus, 84
Get, 85
sound4::bigvoice::CInstance, 109
~CInstance, 110
AddAudio, 110
CInstance, 110
Create, 110
GetAudio, 111
GetBufferIn, 111
GetBufferOut, 111
GetChunkFrames, 111
GetEstimatedDelay, 112
GetMetadataInfos, 112
GetParam, 112
GetWebServerAppHealth, 112
IsOk, 113
NewClient, 113
OnUpdateThreadStart, 113
OnUpdateThreadStop, 113
operator=, 113
PresetManager_InformChange, 113
ProcessAnyAudio, 114
ProcessAudio, 114
ProcessAudio_Planar, 114
SetBus, 115
SetMetadata, 115
SetMetadataMulti, 116
SetParam, 116

SetPresetManager, 116
SetWebServerAppHealth, 117
StartWebServer, 117
Stop, 117
StopWebServer, 117
Terminate, 118
sound4::bigvoice::CInstance::CClient, 87
~CClient, 88
CClient, 87
operator=, 88
ProcessJson, 88
sound4::bigvoice::CPresetLoader, 138
~CPresetLoader, 139
CPresetLoader, 139
Exists, 139
GetAll, 139
IsReadOnly, 139
Read, 139
Remove, 139
Rename, 139
Write, 139
sound4::bigvoice::dyn, 78
ChunkFrames, 79
InputChannels, 79
InputSampleSize, 80
log_cb_t, 79
LogSeverity, 79
OutputChannels, 80
OutputSampleSize, 80
PresetChange_Kind, 79
SampleFormat, 79
SampleRate, 80
sound4::bigvoice::dyn::CBus, 85
~CBus, 85
CBus, 85
Get, 86
sound4::bigvoice::dyn::CDynLib, 92
AddAudio, 94
AddPadAudio, 94
AudioConvertFrom, 94
AudioConvertTo, 95
AudioMonoFromLiveStereo, 95
AudioMonoToLiveStereo, 95
CDynLib, 93
DeleteClient, 95
ExitProcess, 95
FreeBus, 95
FreeJsonAnswer, 95
FreeMetadataInfos, 96
FreeParameters, 96
FreeParameterValue, 96
GetAudio, 96
GetAudioInputCount, 96
GetAudioOutputCount, 96
GetBufferIn, 96
GetBufferOut, 97
GetBytesFromFormat, 97
GetChannelCount, 97
GetChunkSizeInFrames, 97
GetEstimatedDelay, 97
GetFormatFromName, 97
GetFormatName, 97
GetMaxPacketFrame, 98
GetMetadataInfos, 98
GetOutputCount, 98
GetParameter, 98
GetPossibleChunkSizeInFrames, 98
GetProcessChunkFrames, 98
GetSampleRate, 98
GetVersion, 99
InitProcess, 99
InitProcess2, 99
InitProcess3, 99
IsOk, 94
Load, 94
MonoToStereo, 99
MonoToStereo_Planar, 99
NewBus, 99
NewClient, 100
NewParameters, 100
PresetManager_InformChange, 100
ProcessAudio, 100
ProcessAudio_Planar, 100
ProcessJson, 100
SaveState, 100
SetInstanceBus, 101
SetLoggerCallback, 101
SetLogSeverity, 101
SetMetadata, 101
SetMetadataMulti, 101
SetParameter, 101
SetPresetManager, 101
StartUpdateThread, 102
StereoToMono, 102
StereoToMono_Planar, 102
StopUpdateThread, 102
TerminateProcess, 102
WaitUpdateThreadReady, 102
Webserver, 102
Webserver_FreeString, 103
Webserver_GetAppHealth, 103
Webserver_SetAppHealth, 103
Webserver_Status, 103
Webserver_Stop, 103
Webserver_tcp, 103
Webserver_tcp2, 103
sound4::bigvoice::dyn::CInstance, 118
~CInstance, 120
AddAudio, 120
CInstance, 120
Create, 120
GetAudio, 121
GetBufferIn, 121
GetBufferOut, 121
GetChunkFrames, 121
GetEstimatedDelay, 122

GetMetadataInfos, 122
GetParam, 122
GetWebServerAppHealth, 122
IsOk, 123
NewClient, 123
OnUpdateThreadStart, 123
OnUpdateThreadStop, 123
operator=, 123
PresetManager_InformChange, 123
ProcessAnyAudio, 124
ProcessAudio, 124
ProcessAudio_Planar, 124
SetBus, 125
SetMetadata, 125
SetMetadataMulti, 126
SetParam, 126
SetPresetManager, 126
SetWebServerAppHealth, 127
StartWebServer, 127
Stop, 127
StopWebServer, 127
Terminate, 128
sound4::bigvoice::dyn::CInstance::CClient, 89
~CClient, 89
CClient, 89
operator=, 89
ProcessJson, 89
sound4::bigvoice::dyn::CPresetLoader, 140
~CPresetLoader, 140
CPresetLoader, 140
Exists, 141
GetAll, 141
IsReadOnly, 141
Read, 141
Remove, 141
Rename, 141
Write, 141
sound4::bigvoice::dyn::helper, 80
DynFunc_t, 81
DynLib_t, 81
sound4::bigvoice::dyn::helper::CDynLoader, 104
~CDynLoader, 105
CDynLoader, 104, 105
Close, 105
GetSymbol, 105
GetSymbol_safe, 105
GetThisLibraryPath, 105
IsOk, 105
Load, 106
operator bool, 106
operator=, 106
sound4::bigvoice::dyn::helper::DynFuncHolder< T >, 162
DynFuncHolder, 162
IsOk, 163
operator bool, 163
operator T*, 163
sound4::bigvoice::dyn::helper::SampleFormat< float >, 171
format, 172
sound4::bigvoice::dyn::helper::SampleFormat< int16_t >, 173
format, 173
sound4::bigvoice::dyn::helper::SampleFormat< int32_t >, 175
format, 175
sound4::bigvoice::dyn::helper::SampleFormat< T >, 169
sound4::bigvoice::helper, 81
sound4::bigvoice::helper::SampleFormat< float >, 172
format, 172
sound4::bigvoice::helper::SampleFormat< int16_t >, 174
format, 174
sound4::bigvoice::helper::SampleFormat< int32_t >, 175
format, 176
sound4::bigvoice::helper::SampleFormat< T >, 169
sound4::CBus, 86
~CBus, 86
CBus, 86
Get, 87
sound4::CInstance, 128
~CInstance, 130
AddAudio, 130
CInstance, 130
Create, 130
GetAudio, 131
GetBufferIn, 131
GetBufferOut, 131
GetChunkFrames, 131
GetEstimatedDelay, 132
GetMetadataInfos, 132
GetParam, 132
GetWebServerAppHealth, 132
IsOk, 133
NewClient, 133
OnUpdateThreadStart, 133
OnUpdateThreadStop, 133
operator=, 133
PresetManager_InformChange, 133
ProcessAnyAudio, 134
ProcessAudio, 134
ProcessAudio_Planar, 134
SetBus, 135
SetMetadata, 135
SetMetadataMulti, 136
SetParam, 136
SetPresetManager, 136
SetWebServerAppHealth, 137
StartWebServer, 137
Stop, 137
StopWebServer, 137
Terminate, 138
sound4::CInstance::CClient, 90

~CClient, 91
CClient, 91
operator=, 91
ProcessJson, 91
sound4::CPresetLoader, 142
~CPresetLoader, 142
CPresetLoader, 142
Exists, 143
GetAll, 143
IsReadOnly, 143
Read, 143
Remove, 143
Rename, 143
Write, 143
sound4::CProcessor, 144
AudioConvertFrom, 146
AudioConvertTo, 146
AudioMonoFromLiveStereo, 147
AudioMonoToLiveStereo, 147
CBus, 152
CInstance, 152
CProcessor, 146
fnAddAudio, 152
fnAddPadAudio, 152
fnAudioConvertFrom, 152
fnAudioConvertTo, 153
fnAudioMonoFromLiveStereo, 153
fnAudioMonoToLiveStereo, 153
fnDeleteClient, 153
fnExitProcess, 153
fnFreeBus, 153
fnFreeJsonAnswer, 153
fnFreeMetadataInfos, 154
fnFreeParameters, 154
fnFreeParameterValue, 154
fnGetAudio, 154
fnGetAudioInputCount, 154
fnGetAudioOutputCount, 154
fnGetBufferIn, 154
fnGetBufferOut, 155
fnGetBytesFromFormat, 155
fnGetChannelCount, 155
fnGetChunkSizeInFrames, 155
fnGetEstimatedDelay, 155
fnGetFormatFromName, 155
fnGetFormatName, 155
fnGetMaxPacketFrame, 156
fnGetMetadataInfos, 156
fnGetOutputCount, 156
fnGetParameter, 156
fnGetPossibleChunkSizeInFrames, 156
fnGetProcessChunkFrames, 156
fnGetSampleRate, 156
fnGetVersion, 157
fnInitProcess, 157
fnInitProcess2, 157
fnInitProcess3, 157
fnMonoToStereo, 157
fnMonoToStereo_Planar, 157
fnNewBus, 157
fnNewClient, 158
fnNewParameters, 158
fnPresetManager_InformChange, 158
fnProcessAudio, 158
fnProcessAudio_Planar, 158
fnProcessJson, 158
fnSaveState, 158
fnSetInstanceBus, 159
fnSetLoggerCallback, 159
fnSetLogSeverity, 159
fnSetMetadata, 159
fnSetMetadataMulti, 159
fnSetParameter, 159
fnSetPresetManager, 159
fnStartUpdateThread, 160
fnStereoToMono, 160
fnStereoToMono_Planar, 160
fnStopUpdateThread, 160
fnTerminateProcess, 160
fnWaitUpdateThreadReady, 160
fnWebserver, 160
fnWebserver_FreeString, 161
fnWebserver_GetAppHealth, 161
fnWebserver_SetAppHealth, 161
fnWebserver_Status, 161
fnWebserver_Stop, 161
fnWebserver_tcp, 161
fnWebserver_tcp2, 161
GetAudioInputCount, 147
GetAudioOutputCount, 147
GetBytesFromFormat, 147
GetChannelCount, 148
GetChunkSizeInFrames, 148
GetFormatFromName, 148
GetFormatName, 149
GetPossibleChunkSizeInFrames, 149
GetPrefixSymbol, 149
GetSampleRate, 150
GetVersion, 150
IsOk, 150
Load, 150
MonoToStereo, 151
MonoToStereo_Planar, 151
SetLoggerCallback, 151
SetLogSeverity, 151
StereoToMono, 151
StereoToMono_Planar, 152
sound4::helper, 81
DynFunc_t, 82
DynLib_t, 82
sound4::helper::CDynLoader, 106
~CDynLoader, 107
CDynLoader, 107
Close, 107
GetSymbol, 107
GetSymbol_safe, 107

GetThisLibraryPath, 108
IsOk, 108
Load, 108
operator bool, 108
operator=, 108
sound4::helper::DynFuncHolder< T >, 163
DynFuncHolder, 164
IsOk, 164
operator bool, 164
operator T*, 164
sound4::helper::SampleFormat< float >, 172
format, 173
sound4::helper::SampleFormat< int16_t >, 174
format, 174
sound4::helper::SampleFormat< int32_t >, 176
format, 176
sound4::helper::SampleFormat< T >, 171
SOUND4_GetProcessInfo
Global setup, 27
SOUND4_INVALID_WEBSERVER_ID
sound4cl_cdef.h, 248
Web server, 61
SOUND4_ProcessInfo, 176
name, 177
prefix, 177
shortname, 177
struct_size, 177
version, 177
SOUND4_WEBSERVER_HTTP_OK
sound4cl_cdef.h, 249
Web server, 61
SOUND4_WEBSERVER_HTTPS_OK
sound4cl_cdef.h, 249
Web server, 62
sound4cl-pipe.cpp, 195, 197
GetEnv, 195
main, 195
ProcessLoop, 196
sound4cl.hpp, 231, 233
UNIXLIKE, 233
sound4cl_AddAudio
sound4cl_cdef.h, 252
sound4cl_AddPadAudio
sound4cl_cdef.h, 252
sound4cl_AudioConvertFrom
sound4cl_cdef.h, 252
sound4cl_AudioConvertTo
sound4cl_cdef.h, 252
sound4cl_AudioMonoFromLiveStereo
sound4cl_cdef.h, 252
sound4cl_AudioMonoToLiveStereo
sound4cl_cdef.h, 252
sound4cl_cdef.h, 246, 261
change_kind_created, 251
change_kind_deleted, 251
change_kind_modified, 251
F32_BE, 251
F32_LE, 251
F32_NATIVE, 251
fs_char, 249
fs_strdup, 248
INVALID_FORMAT, 251
S16_BE, 251
S16_LE, 251
S16_NATIVE, 251
S24_BE, 251
S24_LE, 251
S24_NATIVE, 251
S32_BE, 251
S32_LE, 251
S32_NATIVE, 251
SOUND4_INVALID_WEBSERVER_ID, 248
SOUND4_WEBSERVER_HTTP_OK, 249
SOUND4_WEBSERVER_HTTPS_OK, 249
sound4cl_AddAudio, 252
sound4cl_AddPadAudio, 252
sound4cl_AudioConvertFrom, 252
sound4cl_AudioConvertTo, 252
sound4cl_AudioMonoFromLiveStereo, 252
sound4cl_AudioMonoToLiveStereo, 252
sound4cl_DeleteClient, 252
sound4cl_error, 251
sound4cl_ExitProcess, 253
sound4cl_fatal, 251
sound4cl_FreeBus, 253
sound4cl_FreeJsonAnswer, 253
sound4cl_FreeMetadataInfos, 253
sound4cl_FreeParameters, 253
sound4cl_FreeParameterValue, 253
sound4cl_GetAudio, 253
sound4cl_GetAudioInputCount, 253
sound4cl_GetAudioOutputCount, 254
sound4cl_GetBufferIn, 254
sound4cl_GetBufferOut, 254
sound4cl_GetBytesFromFormat, 254
sound4cl_GetChannelCount, 254
sound4cl_GetChunkSizeInFrames, 254
sound4cl_GetEstimatedDelay, 254
sound4cl_GetFormatFromName, 254
sound4cl_GetFormatName, 254
sound4cl_GetMaxPacketFrame, 255
sound4cl_GetMetadataInfos, 255
sound4cl_GetOutputCount, 255
sound4cl_GetParameter, 255
sound4cl_GetPossibleChunkSizeInFrames, 255
sound4cl_GetProcessChunkFrames, 255
sound4cl_GetSampleRate, 255
sound4cl_GetVersion, 255
sound4cl_info, 251
sound4cl_InitProcess, 255
sound4cl_InitProcess2, 256
sound4cl_InitProcess3, 256
sound4cl_loggerfn, 249
sound4cl_LogSeverity, 250
sound4cl_MonoToStereo, 256
sound4cl_MonoToStereo_Planar, 256

sound4cl_NewBus, 256
sound4cl_NewClient, 256
sound4cl_NewParameters, 257
sound4cl_none, 251
sound4cl_PresetChange_Kind, 251
sound4cl_PresetManager_InformChange, 257
sound4cl_ProcessAudio, 257
sound4cl_ProcessAudio_Planar, 257
sound4cl_ProcessJson, 257
sound4cl_SampleFormat, 251
sound4cl_SaveState, 257
sound4cl_SetInstanceBus, 257
sound4cl_SetLoggerCallback, 257
sound4cl_SetLogSeverity, 258
sound4cl_SetMetadata, 258
sound4cl_SetMetadataMulti, 258
sound4cl_SetParameter, 258
sound4cl_SetPresetManager, 258
sound4cl_SOUND4_GetProcessInfo, 258
sound4cl_StartUpdateThread, 258
sound4cl_StereoToMono, 259
sound4cl_StereoToMono_Planar, 259
sound4cl_StopUpdateThread, 259
sound4cl_storage_exists, 249
sound4cl_storage_getall, 249
sound4cl_storage_getall_free, 249
sound4cl_storage_reader, 250
sound4cl_storage_reader_free, 250
sound4cl_storage_remove, 250
sound4cl_storage_rename, 250
sound4cl_storage_writer, 250
sound4cl_TerminateProcess, 259
sound4cl_verbose, 251
sound4cl_verbose2, 251
sound4cl_verbose3, 251
sound4cl_verbose4, 251
sound4cl_verbose5, 251
sound4cl_WaitUpdateThreadReady, 259
sound4cl_warning, 251
sound4cl_Webserver, 259
sound4cl_Webserver_FreeString, 259
sound4cl_Webserver_GetAppHealth, 260
sound4cl_Webserver_SetAppHealth, 260
sound4cl_Webserver_Status, 260
sound4cl_Webserver_Stop, 260
sound4cl_Webserver_tcp, 260
sound4cl_Webserver_tcp2, 260
sound4cl_DeleteClient
sound4cl_cdef.h, 252
sound4cl_error
sound4cl_cdef.h, 251
sound4cl_ExitProcess
sound4cl_cdef.h, 253
sound4cl_fatal
sound4cl_cdef.h, 251
sound4cl_FreeBus
sound4cl_cdef.h, 253
sound4cl_FreeJsonAnswer
sound4cl_cdef.h, 253
sound4cl_FreeMetadataInfos
sound4cl_cdef.h, 253
sound4cl_FreeParameters
sound4cl_cdef.h, 253
sound4cl_FreeParameterValue
sound4cl_cdef.h, 253
sound4cl_GetAudio
sound4cl_cdef.h, 253
sound4cl_GetAudioInputCount
sound4cl_cdef.h, 253
sound4cl_GetAudioOutputCount
sound4cl_cdef.h, 254
sound4cl_GetBufferIn
sound4cl_cdef.h, 254
sound4cl_GetBufferOut
sound4cl_cdef.h, 254
sound4cl_GetBytesFromFormat
sound4cl_cdef.h, 254
sound4cl_GetChannelCount
sound4cl_cdef.h, 254
sound4cl_GetChunkSizeInFrames
sound4cl_cdef.h, 254
sound4cl_GetEstimatedDelay
sound4cl_cdef.h, 254
sound4cl_GetFormatFromName
sound4cl_cdef.h, 254
sound4cl_GetFormatName
sound4cl_cdef.h, 254
sound4cl_GetMaxPacketFrame
sound4cl_cdef.h, 255
sound4cl_GetMetadataInfos
sound4cl_cdef.h, 255
sound4cl_GetOutputCount
sound4cl_cdef.h, 255
sound4cl_GetParameter
sound4cl_cdef.h, 255
sound4cl_GetPossibleChunkSizeInFrames
sound4cl_cdef.h, 255
sound4cl_GetProcessChunkFrames
sound4cl_cdef.h, 255
sound4cl_GetSampleRate
sound4cl_cdef.h, 255
sound4cl_GetVersion
sound4cl_cdef.h, 255
sound4cl_info
sound4cl_cdef.h, 251
sound4cl_InitProcess
sound4cl_cdef.h, 255
sound4cl_InitProcess2
sound4cl_cdef.h, 256
sound4cl_InitProcess3
sound4cl_cdef.h, 256
sound4cl_loggerfn
sound4cl_cdef.h, 249
sound4cl_LogSeverity
sound4cl_cdef.h, 250
sound4cl_MonoToStereo

sound4cl_cdef.h, 256
sound4cl_MonoToStereo_Planar
sound4cl_cdef.h, 256
sound4cl_NewBus
sound4cl_cdef.h, 256
sound4cl_NewClient
sound4cl_cdef.h, 256
sound4cl_NewParameters
sound4cl_cdef.h, 257
sound4cl_none
sound4cl_cdef.h, 251
sound4cl_PresetChange_Kind
sound4cl_cdef.h, 251
sound4cl_PresetManager_InformChange
sound4cl_cdef.h, 257
sound4cl_ProcessAudio
sound4cl_cdef.h, 257
sound4cl_ProcessAudio_Planar
sound4cl_cdef.h, 257
sound4cl_ProcessJson
sound4cl_cdef.h, 257
sound4cl_SampleFormat
sound4cl_cdef.h, 251
sound4cl_SaveState
sound4cl_cdef.h, 257
sound4cl_SetInstanceBus
sound4cl_cdef.h, 257
sound4cl_SetLoggerCallback
sound4cl_cdef.h, 257
sound4cl_SetLogSeverity
sound4cl_cdef.h, 258
sound4cl_SetMetadata
sound4cl_cdef.h, 258
sound4cl_SetMetadataMulti
sound4cl_cdef.h, 258
sound4cl_SetParameter
sound4cl_cdef.h, 258
sound4cl_SetPresetManager
sound4cl_cdef.h, 258
sound4cl_SOUND4_GetProcessInfo
sound4cl_cdef.h, 258
sound4cl_SOUND4_ProcessInfo, 178
name, 178
prefix, 178
shortname, 178
struct_size, 179
version, 179
sound4cl_StartUpdateThread
sound4cl_cdef.h, 258
sound4cl_StereoToMono
sound4cl_cdef.h, 259
sound4cl_StereoToMono_Planar
sound4cl_cdef.h, 259
sound4cl_StopUpdateThread
sound4cl_cdef.h, 259
sound4cl_storage_exists
sound4cl_cdef.h, 249
sound4cl_storage_getall
sound4cl_cdef.h, 249
sound4cl_storage_getall_free
sound4cl_cdef.h, 249
sound4cl_storage_reader
sound4cl_cdef.h, 250
sound4cl_storage_reader_free
sound4cl_cdef.h, 250
sound4cl_storage_remove
sound4cl_cdef.h, 250
sound4cl_storage_rename
sound4cl_cdef.h, 250
sound4cl_storage_writer
sound4cl_cdef.h, 250
sound4cl_TerminateProcess
sound4cl_cdef.h, 259
sound4cl_verbose
sound4cl_cdef.h, 251
sound4cl_verbose2
sound4cl_cdef.h, 251
sound4cl_verbose3
sound4cl_cdef.h, 251
sound4cl_verbose4
sound4cl_cdef.h, 251
sound4cl_verbose5
sound4cl_cdef.h, 251
sound4cl_WaitUpdateThreadReady
sound4cl_cdef.h, 259
sound4cl_warning
sound4cl_cdef.h, 251
sound4cl_Webserver
sound4cl_cdef.h, 259
sound4cl_Webserver_FreeString
sound4cl_cdef.h, 259
sound4cl_Webserver_GetAppHealth
sound4cl_cdef.h, 260
sound4cl_Webserver_SetAppHealth
sound4cl_cdef.h, 260
sound4cl_Webserver_Status
sound4cl_cdef.h, 260
sound4cl_Webserver_Stop
sound4cl_cdef.h, 260
sound4cl_Webserver_tcp
sound4cl_cdef.h, 260
sound4cl_Webserver_tcp2
sound4cl_cdef.h, 260
StartUpdateThread
sound4::bigvoice::dyn::CDynLib, 102
StartWebServer
sound4::bigvoice::CInstance, 117
sound4::bigvoice::dyn::CInstance, 127
sound4::CInstance, 137
std, 82
StereoToMono
sound4::bigvoice::dyn::CDynLib, 102
sound4::CProcessor, 151
StereoToMono_Planar
sound4::bigvoice::dyn::CDynLib, 102
sound4::CProcessor, 152

Stop
sound4::bigvoice::CInstance, 117
sound4::bigvoice::dyn::CInstance, 127
sound4::CInstance, 137
StopUpdateThread
sound4::bigvoice::dyn::CDynLib, 102
StopWebServer
sound4::bigvoice::CInstance, 117
sound4::bigvoice::dyn::CInstance, 127
sound4::CInstance, 137
struct_size
SOUND4_ProcessInfo, 177
sound4cl_SOUND4_ProcessInfo, 179

Terminate
sound4::bigvoice::CInstance, 118
sound4::bigvoice::dyn::CInstance, 128
sound4::CInstance, 138
TerminateProcess
sound4::bigvoice::dyn::CDynLib, 102

UNIXLIKE
sound4.bigvoice.cl_dyn.hpp, 218
sound4cl.hpp, 233
Update Management, 40
bigvoice_StartUpdateThread, 41
bigvoice_StopUpdateThread, 41
bigvoice_WaitUpdateThreadReady, 41
Usage
sound4.bigvoice.cl-pipe-cxx.cpp, 182
sound4.bigvoice.cl-pipe-dyn.cpp, 186
sound4.bigvoice.cl-pipe.cpp, 191

verbose
Global setup, 23
sound4, 75
verbose2
Global setup, 23
sound4, 75
verbose3
Global setup, 23
sound4, 75
verbose4
Global setup, 23
sound4, 75
verbose5
Global setup, 23
sound4, 75
version
SOUND4_ProcessInfo, 177
sound4cl_SOUND4_ProcessInfo, 179

WaitUpdateThreadReady
sound4::bigvoice::dyn::CDynLib, 102
warning
Global setup, 23
sound4, 75
Web server, 61
bigvoice_Webserver, 62
bigvoice_Webserver_FreeString, 62
bigvoice_Webserver_GetAppHealth, 63
bigvoice_Webserver_SetAppHealth, 63
bigvoice_Webserver_Status, 63
bigvoice_Webserver_Stop, 64
bigvoice_Webserver_tcp, 64
bigvoice_Webserver_tcp2, 65
SOUND4_INVALID_WEBSERVER_ID, 61
SOUND4_WEBSERVER_HTTP_OK, 61
SOUND4_WEBSERVER_HTTPS_OK, 62
Webserver
sound4::bigvoice::dyn::CDynLib, 102
Webserver_FreeString
sound4::bigvoice::dyn::CDynLib, 103
Webserver_GetAppHealth
sound4::bigvoice::dyn::CDynLib, 103
Webserver_SetAppHealth
sound4::bigvoice::dyn::CDynLib, 103
Webserver_Status
sound4::bigvoice::dyn::CDynLib, 103
Webserver_Stop
sound4::bigvoice::dyn::CDynLib, 103
Webserver_tcp
sound4::bigvoice::dyn::CDynLib, 103
Webserver_tcp2
sound4::bigvoice::dyn::CDynLib, 103
Write
MyPresetManager, 168
sound4::bigvoice::CPresetLoader, 139
sound4::bigvoice::dyn::CPresetLoader, 141
sound4::CPresetLoader, 143