SOUND4 x1.CLOUD Library [1.5.1]

Generated by Doxygen 1.9.1

1 SOUND4 x1.CLOUD Library 1
1.1 General usage . . . 1
1.2 Examples . . . 1

2 Deprecated List 3

3 Module Index 5
3.1 Modules . . . 5

4 Namespace Index 7
4.1 Namespace List . . . 7

5 Hierarchical Index 9
5.1 Class Hierarchy . . . 9

6 Class Index 13
6.1 Class List . . . 13

7 File Index 15
7.1 File List . . . 15

8 Module Documentation 17
8.1 Sample using C interface . . . 17
8.1.1 Sample to create the processing instance . . . 17
8.1.2 Sample processing loop . . . 18
8.1.3 Sample to stop the processing instance . . . 18
8.2 Environment variables and parameters . . . 19
8.2.1 Parameters defaulting from environment . . . 19
8.2.1.1 Webserver authentification setup : . . . 19
8.2.1.2 Webserver secure : . . . 19
8.2.1.3 Network access : . . . 19
8.2.2 Parameters . . . 19
8.2.3 Preset sharing . . . 20
8.3 Global setup . . . 20
8.3.1 Detailed Description . . . 21
8.3.2 Macro Definition Documentation . . . 21
8.3.2.1 CLOUDX1_AUDIO_INPUT_COUNT . . . 21
8.3.2.2 CLOUDX1_AUDIO_OUTPUT_COUNT . . . 21
8.3.2.3 CLOUDX1_AUDIOFRAME_COUNT . . . 21
8.3.2.4 CLOUDX1_CHANNEL_COUNT . . . 22
8.3.2.5 CLOUDX1_SAMPLE_RATE . . . 22
8.3.3 Typedef Documentation . . . 22
8.3.3.1 cloudx1_loggerfn . . . 22
8.3.4 Enumeration Type Documentation . . . 22
8.3.4.1 cloudx1_LogSeverity . . . 22

8.3.5 Function Documentation . . . 23
8.3.5.1 cloudx1_GetAudioInputCount() . . . 23
8.3.5.2 cloudx1_GetAudioOutputCount() . . . 23
8.3.5.3 cloudx1_GetChannelCount() . . . 24
8.3.5.4 cloudx1_GetChunkSizeInFrames() . . . 24
8.3.5.5 cloudx1_GetPossibleChunkSizeInFrames() . . . 24
8.3.5.6 cloudx1_GetSampleRate() . . . 25
8.3.5.7 cloudx1_GetVersion() . . . 25
8.3.5.8 cloudx1_SetLoggerCallback() . . . 25
8.3.5.9 cloudx1_SetLogSeverity() . . . 26
8.3.5.10 SOUND4_GetProcessInfo() . . . 26
8.4 Process creation/destruction . . . 26
8.4.1 Detailed Description . . . 27
8.4.2 Function Documentation . . . 27
8.4.2.1 cloudx1_ExitProcess() . . . 27
8.4.2.2 cloudx1_FreeParameters() . . . 27
8.4.2.3 cloudx1_FreeParameterValue() . . . 27
8.4.2.4 cloudx1_GetParameter() . . . 27
8.4.2.5 cloudx1_GetProcessChunkFrames() . . . 28
8.4.2.6 cloudx1_InitProcess() . . . 28
8.4.2.7 cloudx1_InitProcess2() . . . 29
8.4.2.8 cloudx1_InitProcess3() . . . 29
8.4.2.9 cloudx1_NewParameters() . . . 30
8.4.2.10 cloudx1_SetParameter() . . . 31
8.4.2.11 cloudx1_TerminateProcess() . . . 31
8.5 Custom preset managagement . . . 31
8.5.1 Detailed Description . . . 32
8.5.2 Typedef Documentation . . . 32
8.5.2.1 cloudx1_storage_exists . . . 32
8.5.2.2 cloudx1_storage_getall . . . 33
8.5.2.3 cloudx1_storage_getall_free . . . 33
8.5.2.4 cloudx1_storage_reader . . . 33
8.5.2.5 cloudx1_storage_reader_free . . . 34
8.5.2.6 cloudx1_storage_remove . . . 34
8.5.2.7 cloudx1_storage_rename . . . 35
8.5.2.8 cloudx1_storage_writer . . . 35
8.5.2.9 fs_char . . . 35
8.5.3 Enumeration Type Documentation . . . 35
8.5.3.1 cloudx1_PresetChange_Kind . . . 36
8.5.4 Function Documentation . . . 36
8.5.4.1 cloudx1_PresetManager_InformChange() . . . 36
8.5.4.2 cloudx1_SetPresetManager() . . . 36

8.6 Metadata Management . . . 37
8.6.1 Detailed Description . . . 37
8.6.2 Function Documentation . . . 37
8.6.2.1 cloudx1_FreeMetadataInfos() . . . 37
8.6.2.2 cloudx1_GetMetadataInfos() . . . 38
8.6.2.3 cloudx1_SetMetadata() . . . 38
8.6.2.4 cloudx1_SetMetadataMulti() . . . 39
8.7 Update Management . . . 39
8.7.1 Detailed Description . . . 39
8.7.2 Function Documentation . . . 39
8.7.2.1 cloudx1_StartUpdateThread() . . . 39
8.7.2.2 cloudx1_StopUpdateThread() . . . 40
8.7.2.3 cloudx1_WaitUpdateThreadReady() . . . 40
8.8 Direct processing . . . 40
8.8.1 Detailed Description . . . 41
8.8.2 Function Documentation . . . 41
8.8.2.1 cloudx1_GetBufferIn() . . . 41
8.8.2.2 cloudx1_GetBufferOut() . . . 41
8.8.2.3 cloudx1_GetEstimatedDelay() . . . 42
8.8.2.4 cloudx1_ProcessAudio() . . . 42
8.8.2.5 cloudx1_ProcessAudio_Planar() . . . 43
8.9 Audio managing (any audio) . . . 43
8.9.1 Detailed Description . . . 44
8.9.2 Enumeration Type Documentation . . . 44
8.9.2.1 cloudx1_SampleFormat . . . 44
8.9.3 Function Documentation . . . 45
8.9.3.1 cloudx1_AddAudio() . . . 45
8.9.3.2 cloudx1_AddPadAudio() . . . 45
8.9.3.3 cloudx1_GetAudio() . . . 45
8.9.3.4 cloudx1_GetBytesFromFormat() . . . 46
8.9.3.5 cloudx1_GetFormatFromName() . . . 46
8.9.3.6 cloudx1_GetFormatName() . . . 47
8.9.3.7 cloudx1_GetMaxPacketFrame() . . . 47
8.9.3.8 cloudx1_GetOutputCount() . . . 47
8.10 Audio convert helpers . . . 48
8.10.1 Detailed Description . . . 48
8.10.2 Function Documentation . . . 48
8.10.2.1 cloudx1_AudioConvertFrom() . . . 48
8.10.2.2 cloudx1_AudioConvertTo() . . . 49
8.10.2.3 cloudx1_AudioMonoFromLiveStereo() . . . 49
8.10.2.4 cloudx1_AudioMonoToLiveStereo() . . . 49
8.10.2.5 cloudx1_MonoToStereo() . . . 50

8.10.2.6 cloudx1_MonoToStereo_Planar() . . . 50
8.10.2.7 cloudx1_StereoToMono() . . . 51
8.10.2.8 cloudx1_StereoToMono_Planar() . . . 51
8.11 Client's JSon config interfacing . . . 52
8.11.1 Detailed Description . . . 52
8.11.2 Function Documentation . . . 52
8.11.2.1 cloudx1_DeleteClient() . . . 52
8.11.2.2 cloudx1_FreeJsonAnswer() . . . 52
8.11.2.3 cloudx1_NewClient() . . . 53
8.11.2.4 cloudx1_ProcessJson() . . . 53
8.11.2.5 cloudx1_SaveState() . . . 54
8.12 JSON syntax . . . 54
8.12.1 Commands . . . 54
8.12.2 Answers . . . 55
8.12.3 Examples . . . 56
8.12.3.1 Get status . . . 56
8.12.3.2 Get all changes since last request . . . 56
8.12.3.3 Check license . . . 56
8.12.3.4 Set Processor name . . . 56
8.12.3.5 Get presetlist . . . 56
8.12.3.6 Load a preset . . . 56
8.12.3.7 Get information on preset on air . . . 56
8.12.3.8 Set input gain . . . 57
8.12.3.9 Set bypass . . . 57
8.12.3.10 Export a preset . . . 57
8.12.3.11 Import a preset . . . 57
8.13 JSON syntax for internal setup . . . 57
8.13.1 Get process and modules information . . . 58
8.13.2 Authentication management . . . 58
8.13.2.1 Difference between setup and admin account . . . 58
8.13.2.2 Access encoding . . . 58
8.13.2.3 Change setup account . . . 58
8.13.2.4 Get user info . . . 59
8.13.2.5 Add or modify a user . . . 59
8.13.2.6 Removes a user . . . 59
8.13.2.7 List users . . . 59
8.13.2.8 Get/set unlogged access . . . 59
8.13.2.9 Get realm . . . 59
8.13.2.10 Get API version . . . 60
8.13.2.11 Get setup is admin . . . 60
8.13.2.12 Get/set setup account . . . 60
8.13.2.13 Get/set custom storage . . . 60

8.13.3 Get process' interface description . . . 60
8.13.3.1 Get XML decription . . . 60
8.13.3.2 Get Process Image . . . 60
8.13.3.3 Get Process Ico . . . 61
8.14 Web server . . . 61
8.14.1 Detailed Description . . . 61
8.14.2 Macro Definition Documentation . . . 61
8.14.2.1 SOUND4_INVALID_WEBSERVER_ID . . . 61
8.14.2.2 SOUND4_WEBSERVER_HTTP_OK . . . 62
8.14.2.3 SOUND4_WEBSERVER_HTTPS_OK . . . 62
8.14.3 Function Documentation . . . 62
8.14.3.1 cloudx1_Webserver() . . . 62
8.14.3.2 cloudx1_Webserver_FreeString() . . . 63
8.14.3.3 cloudx1_Webserver_GetAppHealth() . . . 63
8.14.3.4 cloudx1_Webserver_SetAppHealth() . . . 63
8.14.3.5 cloudx1_Webserver_Status() . . . 64
8.14.3.6 cloudx1_Webserver_Stop() . . . 64
8.14.3.7 cloudx1_Webserver_tcp() . . . 65
8.14.3.8 cloudx1_Webserver_tcp2() . . . 65
8.15 Sample using C++ interface wrapper . . . 66
8.15.1 Setup and sanity checks . . . 66
8.15.2 Sample to create the processing instance . . . 66
8.15.3 Communicating with the instance . . . 66
8.15.4 Starting a web-server . . . 67
8.15.5 Sample processing loop . . . 67
8.16 Sample using C++ interface wrapper with dynamic library loading . . . 67
8.16.1 Dynamic loading . . . 67
8.16.2 Setup and sanity checks . . . 67
8.16.3 Sample to create the processing instance . . . 67
8.16.4 Communicating with the instance . . . 67
8.16.5 Starting a web-server . . . 68
8.16.6 Sample processing loop . . . 68
8.16.7 Custom preset manager . . . 68
8.16.8 Custom preset manager loading . . . 69
8.17 Sample using Generic C++ interface (multi-process compatible) . . . 69
8.17.1 Loading the library . . . 69
8.17.2 Setup and configure . . . 69
8.17.3 Create the processing instance . . . 69
8.17.4 Processing loop . . . 70
8.17.5 Starting a web-server . . . 70
8.17.6 Communicating with the instance . . . 70
8.17.7 Custom preset manager example . . . 70

8.17.8 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 . . . 75
9.2.3.1 error . . . 75
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 . . . 76
9.2.3.7 verbose3 . . . 76
9.2.3.8 verbose4 . . . 76
9.2.3.9 verbose5 . . . 76
9.2.3.10 warning . . . 76
9.3 sound4::cloudx1 Namespace Reference . . . 76
9.3.1 Detailed Description . . . 77
9.3.2 Typedef Documentation . . . 77
9.3.2.1 log_cb_t . . . 77
9.3.2.2 LogSeverity . . . 77
9.3.2.3 PresetChange_Kind . . . 78
9.3.2.4 SampleFormat . . . 78
9.3.3 Variable Documentation . . . 78
9.3.3.1 ChunkFrames . . . 78
9.3.3.2 InputChannels . . . 78
9.3.3.3 InputSampleSize . . . 78
9.3.3.4 OutputChannels . . . 79
9.3.3.5 OutputSampleSize . . . 79
9.3.3.6 SampleRate . . . 79
9.4 sound4::cloudx1::dyn Namespace Reference . . . 79
9.4.1 Typedef Documentation . . . 80
9.4.1.1 log_cb_t . . . 80
9.4.1.2 LogSeverity . . . 80
9.4.1.3 PresetChange_Kind . . . 80
9.4.1.4 SampleFormat . . . 80

9.4.2 Variable Documentation . . . 81
9.4.2.1 ChunkFrames . . . 81
9.4.2.2 InputChannels . . . 81
9.4.2.3 InputSampleSize . . . 81
9.4.2.4 OutputChannels . . . 81
9.4.2.5 OutputSampleSize . . . 81
9.4.2.6 SampleRate . . . 82
9.5 sound4::cloudx1::dyn::helper Namespace Reference . . . 82
9.5.1 Detailed Description . . . 82
9.5.2 Typedef Documentation . . . 82
9.5.2.1 DynFunc_t . . . 82
9.5.2.2 DynLib_t . . . 83
9.6 sound4::cloudx1::helper Namespace Reference . . . 83
9.6.1 Detailed Description . . . 83
9.7 sound4::helper Namespace Reference . . . 83
9.7.1 Detailed Description . . . 83
9.7.2 Typedef Documentation . . . 84
9.7.2.1 DynFunc_t . . . 84
9.7.2.2 DynLib_t . . . 84
9.8 std Namespace Reference . . . 84

10 Class Documentation **85**

10.1 sound4::CBus Class Reference . . . 85
10.1.1 Detailed Description . . . 85
10.1.2 Constructor & Destructor Documentation . . . 85
10.1.2.1 CBus() . . . 85
10.1.2.2 ~CBus() . . . 86
10.1.3 Member Function Documentation . . . 86
10.1.3.1 Get() . . . 86
10.2 sound4::CInstance::CClient Class Reference . . . 86
10.2.1 Detailed Description . . . 86
10.2.2 Constructor & Destructor Documentation . . . 87
10.2.2.1 CClient() [1/2] . . . 87
10.2.2.2 CClient() [2/2] . . . 87
10.2.2.3 ~CClient() . . . 87
10.2.3 Member Function Documentation . . . 87
10.2.3.1 operator=() . . . 87
10.2.3.2 ProcessJson() . . . 87
10.3 sound4::cloudx1::CInstance::CClient Class Reference . . . 88
10.3.1 Detailed Description . . . 88
10.3.2 Constructor & Destructor Documentation . . . 89
10.3.2.1 CClient() [1/2] . . . 89

10.3.2.2 CClient() [2/2] . . . 89
10.3.2.3 ~CClient() . . . 89
10.3.3 Member Function Documentation . . . 89
10.3.3.1 operator=() . . . 89
10.3.3.2 ProcessJson() . . . 89
10.4 sound4::cloudx1::dyn::CInstance::CClient Class Reference . . . 90
10.4.1 Detailed Description . . . 90
10.4.2 Constructor & Destructor Documentation . . . 90
10.4.2.1 CClient() [1/2] . . . 91
10.4.2.2 CClient() [2/2] . . . 91
10.4.2.3 ~CClient() . . . 91
10.4.3 Member Function Documentation . . . 91
10.4.3.1 operator=() . . . 91
10.4.3.2 ProcessJson() . . . 91
10.5 sound4::cloudx1::dyn::CDynLib Class Reference . . . 92
10.5.1 Detailed Description . . . 93
10.5.2 Constructor & Destructor Documentation . . . 94
10.5.2.1 CDynLib() . . . 94
10.5.3 Member Function Documentation . . . 94
10.5.3.1 IsOk() . . . 94
10.5.3.2 Load() . . . 94
10.5.4 Member Data Documentation . . . 94
10.5.4.1 AddAudio . . . 95
10.5.4.2 AddPadAudio . . . 95
10.5.4.3 AudioConvertFrom . . . 95
10.5.4.4 AudioConvertTo . . . 95
10.5.4.5 AudioMonoFromLiveStereo . . . 95
10.5.4.6 AudioMonoToLiveStereo . . . 95
10.5.4.7 DeleteClient . . . 96
10.5.4.8 ExitProcess . . . 96
10.5.4.9 FreeJsonAnswer . . . 96
10.5.4.10 FreeMetadataInfos . . . 96
10.5.4.11 FreeParameters . . . 96
10.5.4.12 FreeParameterValue . . . 96
10.5.4.13 GetAudio . . . 97
10.5.4.14 GetAudioInputCount . . . 97
10.5.4.15 GetAudioOutputCount . . . 97
10.5.4.16 GetBufferIn . . . 97
10.5.4.17 GetBufferOut . . . 97
10.5.4.18 GetBytesFromFormat . . . 97
10.5.4.19 GetChannelCount . . . 98
10.5.4.20 GetChunkSizeInFrames . . . 98

10.5.4.21 GetEstimatedDelay . . . 98
10.5.4.22 GetFormatFromName . . . 98
10.5.4.23 GetFormatName . . . 98
10.5.4.24 GetMaxPacketFrame . . . 98
10.5.4.25 GetMetadataInfos . . . 99
10.5.4.26 GetOutputCount . . . 99
10.5.4.27 GetParameter . . . 99
10.5.4.28 GetPossibleChunkSizeInFrames . . . 99
10.5.4.29 GetProcessChunkFrames . . . 99
10.5.4.30 GetSampleRate . . . 99
10.5.4.31 GetVersion . . . 100
10.5.4.32 InitProcess . . . 100
10.5.4.33 InitProcess2 . . . 100
10.5.4.34 InitProcess3 . . . 100
10.5.4.35 MonoToStereo . . . 100
10.5.4.36 MonoToStereo_Planar . . . 100
10.5.4.37 NewClient . . . 101
10.5.4.38 NewParameters . . . 101
10.5.4.39 PresetManager_InformChange . . . 101
10.5.4.40 ProcessAudio . . . 101
10.5.4.41 ProcessAudio_Planar . . . 101
10.5.4.42 ProcessJson . . . 101
10.5.4.43 SaveState . . . 102
10.5.4.44 SetLoggerCallback . . . 102
10.5.4.45 SetLogSeverity . . . 102
10.5.4.46 SetMetadata . . . 102
10.5.4.47 SetMetadataMulti . . . 102
10.5.4.48 SetParameter . . . 102
10.5.4.49 SetPresetManager . . . 103
10.5.4.50 StartUpdateThread . . . 103
10.5.4.51 StereoToMono . . . 103
10.5.4.52 StereoToMono_Planar . . . 103
10.5.4.53 StopUpdateThread . . . 103
10.5.4.54 TerminateProcess . . . 103
10.5.4.55 WaitUpdateThreadReady . . . 104
10.5.4.56 Webserver . . . 104
10.5.4.57 Webserver_FreeString . . . 104
10.5.4.58 Webserver_GetAppHealth . . . 104
10.5.4.59 Webserver_SetAppHealth . . . 104
10.5.4.60 Webserver_Status . . . 104
10.5.4.61 Webserver_Stop . . . 105
10.5.4.62 Webserver_tcp . . . 105

10.5.4.63 Webserver_tcp2 . . . 105
10.6 sound4::cloudx1::dyn::helper::CDynLoader Class Reference . . . 105
10.6.1 Detailed Description . . . 106
10.6.2 Constructor & Destructor Documentation . . . 106
10.6.2.1 CDynLoader() [1/3] . . . 106
10.6.2.2 CDynLoader() [2/3] . . . 106
10.6.2.3 CDynLoader() [3/3] . . . 106
10.6.2.4 ~CDynLoader() . . . 106
10.6.3 Member Function Documentation . . . 106
10.6.3.1 Close() . . . 107
10.6.3.2 GetSymbol() . . . 107
10.6.3.3 GetSymbol_safe() . . . 107
10.6.3.4 GetThisLibraryPath() . . . 107
10.6.3.5 IsOk() . . . 107
10.6.3.6 Load() . . . 108
10.6.3.7 operator bool() . . . 108
10.6.3.8 operator=() . . . 108
10.7 sound4::helper::CDynLoader Class Reference . . . 108
10.7.1 Detailed Description . . . 109
10.7.2 Constructor & Destructor Documentation . . . 109
10.7.2.1 CDynLoader() [1/3] . . . 109
10.7.2.2 CDynLoader() [2/3] . . . 109
10.7.2.3 CDynLoader() [3/3] . . . 109
10.7.2.4 ~CDynLoader() . . . 109
10.7.3 Member Function Documentation . . . 109
10.7.3.1 Close() . . . 110
10.7.3.2 GetSymbol() . . . 110
10.7.3.3 GetSymbol_safe() . . . 110
10.7.3.4 GetThisLibraryPath() . . . 110
10.7.3.5 IsOk() . . . 110
10.7.3.6 Load() . . . 111
10.7.3.7 operator bool() . . . 111
10.7.3.8 operator=() . . . 111
10.8 sound4::CInstance Class Reference . . . 111
10.8.1 Detailed Description . . . 112
10.8.2 Constructor & Destructor Documentation . . . 113
10.8.2.1 CInstance() [1/2] . . . 113
10.8.2.2 CInstance() [2/2] . . . 113
10.8.2.3 ~CInstance() . . . 113
10.8.3 Member Function Documentation . . . 113
10.8.3.1 AddAudio() . . . 113
10.8.3.2 Create() . . . 113

10.8.3.3 GetAudio() . . . 114
10.8.3.4 GetBufferIn() . . . 114
10.8.3.5 GetBufferOut() . . . 115
10.8.3.6 GetChunkFrames() . . . 115
10.8.3.7 GetEstimatedDelay() . . . 115
10.8.3.8 GetMetadataInfos() . . . 116
10.8.3.9 GetParam() . . . 116
10.8.3.10 GetWebServerAppHealth() . . . 116
10.8.3.11 IsOk() . . . 116
10.8.3.12 NewClient() . . . 117
10.8.3.13 OnUpdateThreadStart() . . . 117
10.8.3.14 OnUpdateThreadStop() . . . 117
10.8.3.15 operator=() . . . 117
10.8.3.16 PresetManager_InformChange() . . . 117
10.8.3.17 ProcessAnyAudio() . . . 118
10.8.3.18 ProcessAudio() . . . 118
10.8.3.19 ProcessAudio_Planar() . . . 119
10.8.3.20 SetBus() . . . 119
10.8.3.21 SetMetadata() . . . 119
10.8.3.22 SetMetadataMulti() . . . 120
10.8.3.23 SetParam() . . . 120
10.8.3.24 SetPresetManager() . . . 121
10.8.3.25 SetWebServerAppHealth() . . . 121
10.8.3.26 StartWebServer() . . . 121
10.8.3.27 Stop() . . . 122
10.8.3.28 StopWebServer() . . . 122
10.8.3.29 Terminate() . . . 122
10.9 sound4::cloudx1::CInstance Class Reference . . . 122
10.9.1 Detailed Description . . . 123
10.9.2 Constructor & Destructor Documentation . . . 124
10.9.2.1 CInstance() [1/2] . . . 124
10.9.2.2 CInstance() [2/2] . . . 124
10.9.2.3 ~CInstance() . . . 124
10.9.3 Member Function Documentation . . . 124
10.9.3.1 AddAudio() . . . 124
10.9.3.2 Create() . . . 124
10.9.3.3 GetAudio() . . . 125
10.9.3.4 GetBufferIn() . . . 125
10.9.3.5 GetBufferOut() . . . 126
10.9.3.6 GetChunkFrames() . . . 126
10.9.3.7 GetEstimatedDelay() . . . 126
10.9.3.8 GetMetadataInfos() . . . 127

10.9.3.9 GetParam() . . . 127
10.9.3.10 GetWebServerAppHealth() . . . 127
10.9.3.11 IsOk() . . . 127
10.9.3.12 NewClient() . . . 128
10.9.3.13 OnUpdateThreadStart() . . . 128
10.9.3.14 OnUpdateThreadStop() . . . 128
10.9.3.15 operator=() . . . 128
10.9.3.16 PresetManager_InformChange() . . . 128
10.9.3.17 ProcessAnyAudio() . . . 129
10.9.3.18 ProcessAudio() . . . 129
10.9.3.19 ProcessAudio_Planar() . . . 130
10.9.3.20 SetMetadata() . . . 130
10.9.3.21 SetMetadataMulti() . . . 130
10.9.3.22 SetParam() . . . 131
10.9.3.23 SetPresetManager() . . . 131
10.9.3.24 SetWebServerAppHealth() . . . 132
10.9.3.25 StartWebServer() . . . 132
10.9.3.26 Stop() . . . 132
10.9.3.27 StopWebServer() . . . 133
10.9.3.28 Terminate() . . . 133
10.10 sound4::cloudx1::dyn::CInstance Class Reference . . . 133
10.10.1 Detailed Description . . . 134
10.10.2 Constructor & Destructor Documentation . . . 134
10.10.2.1 CInstance() [1/2] . . . 135
10.10.2.2 CInstance() [2/2] . . . 135
10.10.2.3 ∼CInstance() . . . 135
10.10.3 Member Function Documentation . . . 135
10.10.3.1 AddAudio() . . . 135
10.10.3.2 Create() . . . 135
10.10.3.3 GetAudio() . . . 136
10.10.3.4 GetBufferIn() . . . 136
10.10.3.5 GetBufferOut() . . . 137
10.10.3.6 GetChunkFrames() . . . 137
10.10.3.7 GetEstimatedDelay() . . . 137
10.10.3.8 GetMetadataInfos() . . . 138
10.10.3.9 GetParam() . . . 138
10.10.3.10 GetWebServerAppHealth() . . . 138
10.10.3.11 IsOk() . . . 138
10.10.3.12 NewClient() . . . 139
10.10.3.13 OnUpdateThreadStart() . . . 139
10.10.3.14 OnUpdateThreadStop() . . . 139
10.10.3.15 operator=() . . . 139

10.10.3.16 PresetManager_InformChange() . . . 139
10.10.3.17 ProcessAnyAudio() . . . 140
10.10.3.18 ProcessAudio() . . . 140
10.10.3.19 ProcessAudio_Planar() . . . 141
10.10.3.20 SetMetadata() . . . 141
10.10.3.21 SetMetadataMulti() . . . 141
10.10.3.22 SetParam() . . . 142
10.10.3.23 SetPresetManager() . . . 142
10.10.3.24 SetWebServerAppHealth() . . . 143
10.10.3.25 StartWebServer() . . . 143
10.10.3.26 Stop() . . . 143
10.10.3.27 StopWebServer() . . . 144
10.10.3.28 Terminate() . . . 144
10.11 cloudx1_CClientInstance Struct Reference . . . 144
10.11.1 Detailed Description . . . 144
10.12 cloudx1_CInstance Struct Reference . . . 144
10.12.1 Detailed Description . . . 145
10.13 cloudx1_CParameters Struct Reference . . . 145
10.13.1 Detailed Description . . . 145
10.14 sound4::cloudx1::CPresetLoader Class Reference . . . 145
10.14.1 Detailed Description . . . 145
10.14.2 Constructor & Destructor Documentation . . . 146
10.14.2.1 CPresetLoader() . . . 146
10.14.2.2 ∼CPresetLoader() . . . 146
10.14.3 Member Function Documentation . . . 146
10.14.3.1 Exists() . . . 146
10.14.3.2 GetAll() . . . 146
10.14.3.3 IsReadOnly() . . . 146
10.14.3.4 Read() . . . 146
10.14.3.5 Remove() . . . 147
10.14.3.6 Rename() . . . 147
10.14.3.7 Write() . . . 147
10.15 sound4::cloudx1::dyn::CPresetLoader Class Reference . . . 147
10.15.1 Detailed Description . . . 147
10.15.2 Constructor & Destructor Documentation . . . 148
10.15.2.1 CPresetLoader() . . . 148
10.15.2.2 ∼CPresetLoader() . . . 148
10.15.3 Member Function Documentation . . . 148
10.15.3.1 Exists() . . . 148
10.15.3.2 GetAll() . . . 148
10.15.3.3 IsReadOnly() . . . 148
10.15.3.4 Read() . . . 149

10.15.3.5 Remove() . . . 149
10.15.3.6 Rename() . . . 149
10.15.3.7 Write() . . . 149
10.16 sound4::CPresetLoader Class Reference . . . 149
10.16.1 Detailed Description . . . 150
10.16.2 Constructor & Destructor Documentation . . . 150
10.16.2.1 CPresetLoader() . . . 150
10.16.2.2 ~CPresetLoader() . . . 150
10.16.3 Member Function Documentation . . . 150
10.16.3.1 Exists() . . . 150
10.16.3.2 GetAll() . . . 151
10.16.3.3 IsReadOnly() . . . 151
10.16.3.4 Read() . . . 151
10.16.3.5 Remove() . . . 151
10.16.3.6 Rename() . . . 151
10.16.3.7 Write() . . . 152
10.17 sound4::CProcessor Class Reference . . . 152
10.17.1 Detailed Description . . . 154
10.17.2 Constructor & Destructor Documentation . . . 154
10.17.2.1 CProcessor() . . . 154
10.17.3 Member Function Documentation . . . 154
10.17.3.1 AudioConvertFrom() . . . 154
10.17.3.2 AudioConvertTo() . . . 155
10.17.3.3 AudioMonoFromLiveStereo() . . . 155
10.17.3.4 AudioMonoToLiveStereo() . . . 155
10.17.3.5 GetAudioInputCount() . . . 155
10.17.3.6 GetAudioOutputCount() . . . 156
10.17.3.7 GetBytesFromFormat() . . . 156
10.17.3.8 GetChannelCount() . . . 156
10.17.3.9 GetChunkSizeInFrames() . . . 157
10.17.3.10 GetFormatFromName() . . . 157
10.17.3.11 GetFormatName() . . . 157
10.17.3.12 GetPossibleChunkSizeInFrames() . . . 158
10.17.3.13 GetPrefixSymbol() . . . 158
10.17.3.14 GetSampleRate() . . . 158
10.17.3.15 GetVersion() . . . 159
10.17.3.16 IsOk() . . . 159
10.17.3.17 Load() . . . 159
10.17.3.18 MonoToStereo() . . . 159
10.17.3.19 MonoToStereo_Planar() . . . 160
10.17.3.20 SetLoggerCallback() . . . 160
10.17.3.21 SetLogSeverity() . . . 160

10.17.3.22 StereoToMono() . . . 160
10.17.3.23 StereoToMono_Planar() . . . 161
10.17.4 Friends And Related Function Documentation . . . 161
10.17.4.1 CBus . . . 161
10.17.4.2 CInstance . . . 161
10.17.5 Member Data Documentation . . . 161
10.17.5.1 fnAddAudio . . . 161
10.17.5.2 fnAddPadAudio . . . 161
10.17.5.3 fnAudioConvertFrom . . . 162
10.17.5.4 fnAudioConvertTo . . . 162
10.17.5.5 fnAudioMonoFromLiveStereo . . . 162
10.17.5.6 fnAudioMonoToLiveStereo . . . 162
10.17.5.7 fnDeleteClient . . . 162
10.17.5.8 fnExitProcess . . . 162
10.17.5.9 fnFreeBus . . . 163
10.17.5.10 fnFreeJsonAnswer . . . 163
10.17.5.11 fnFreeMetadataInfos . . . 163
10.17.5.12 fnFreeParameters . . . 163
10.17.5.13 fnFreeParameterValue . . . 163
10.17.5.14 fnGetAudio . . . 163
10.17.5.15 fnGetAudioInputCount . . . 164
10.17.5.16 fnGetAudioOutputCount . . . 164
10.17.5.17 fnGetBufferIn . . . 164
10.17.5.18 fnGetBufferOut . . . 164
10.17.5.19 fnGetBytesFromFormat . . . 164
10.17.5.20 fnGetChannelCount . . . 164
10.17.5.21 fnGetChunkSizeInFrames . . . 165
10.17.5.22 fnGetEstimatedDelay . . . 165
10.17.5.23 fnGetFormatFromName . . . 165
10.17.5.24 fnGetFormatName . . . 165
10.17.5.25 fnGetMaxPacketFrame . . . 165
10.17.5.26 fnGetMetadataInfos . . . 165
10.17.5.27 fnGetOutputCount . . . 166
10.17.5.28 fnGetParameter . . . 166
10.17.5.29 fnGetPossibleChunkSizeInFrames . . . 166
10.17.5.30 fnGetProcessChunkFrames . . . 166
10.17.5.31 fnGetSampleRate . . . 166
10.17.5.32 fnGetVersion . . . 166
10.17.5.33 fnInitProcess . . . 167
10.17.5.34 fnInitProcess2 . . . 167
10.17.5.35 fnInitProcess3 . . . 167
10.17.5.36 fnMonoToStereo . . . 167

10.17.5.37 fnMonoToStereo_Planar . . . 167
10.17.5.38 fnNewBus . . . 167
10.17.5.39 fnNewClient . . . 168
10.17.5.40 fnNewParameters . . . 168
10.17.5.41 fnPresetManager_InformChange . . . 168
10.17.5.42 fnProcessAudio . . . 168
10.17.5.43 fnProcessAudio_Planar . . . 168
10.17.5.44 fnProcessJson . . . 168
10.17.5.45 fnSaveState . . . 169
10.17.5.46 fnSetInstanceBus . . . 169
10.17.5.47 fnSetLoggerCallback . . . 169
10.17.5.48 fnSetLogSeverity . . . 169
10.17.5.49 fnSetMetadata . . . 169
10.17.5.50 fnSetMetadataMulti . . . 169
10.17.5.51 fnSetParameter . . . 170
10.17.5.52 fnSetPresetManager . . . 170
10.17.5.53 fnStartUpdateThread . . . 170
10.17.5.54 fnStereoToMono . . . 170
10.17.5.55 fnStereoToMono_Planar . . . 170
10.17.5.56 fnStopUpdateThread . . . 170
10.17.5.57 fnTerminateProcess . . . 171
10.17.5.58 fnWaitUpdateThreadReady . . . 171
10.17.5.59 fnWebserver . . . 171
10.17.5.60 fnWebserver_FreeString . . . 171
10.17.5.61 fnWebserver_GetAppHealth . . . 171
10.17.5.62 fnWebserver_SetAppHealth . . . 171
10.17.5.63 fnWebserver_Status . . . 172
10.17.5.64 fnWebserver_Stop . . . 172
10.17.5.65 fnWebserver_tcp . . . 172
10.17.5.66 fnWebserver_tcp2 . . . 172
10.18 sound4::cloudx1::dyn::helper::DynFuncHolder< T > Class Template Reference . . . 172
10.18.1 Detailed Description . . . 172
10.18.2 Constructor & Destructor Documentation . . . 173
10.18.2.1 DynFuncHolder() [1/2] . . . 173
10.18.2.2 DynFuncHolder() [2/2] . . . 173
10.18.3 Member Function Documentation . . . 173
10.18.3.1 IsOk() . . . 173
10.18.3.2 operator bool() . . . 173
10.18.3.3 operator T*() . . . 174
10.19 sound4::helper::DynFuncHolder< T > Class Template Reference . . . 174
10.19.1 Detailed Description . . . 174
10.19.2 Constructor & Destructor Documentation . . . 174

10.19.2.1 DynFuncHolder() [1/2] . . . 174
10.19.2.2 DynFuncHolder() [2/2] . . . 175
10.19.3 Member Function Documentation . . . 175
10.19.3.1 IsOk() . . . 175
10.19.3.2 operator bool() . . . 175
10.19.3.3 operator T∗() . . . 175
10.20 MyPresetManager Class Reference . . . 175
10.20.1 Detailed Description . . . 176
10.20.2 Constructor & Destructor Documentation . . . 176
10.20.2.1 MyPresetManager() [1/2] . . . 176
10.20.2.2 ∼MyPresetManager() [1/2] . . . 176
10.20.2.3 MyPresetManager() [2/2] . . . 176
10.20.2.4 ∼MyPresetManager() [2/2] . . . 177
10.20.3 Member Function Documentation . . . 177
10.20.3.1 Exists() [1/2] . . . 177
10.20.3.2 Exists() [2/2] . . . 177
10.20.3.3 GetAll() [1/2] . . . 177
10.20.3.4 GetAll() [2/2] . . . 177
10.20.3.5 IsReadOnly() [1/2] . . . 178
10.20.3.6 IsReadOnly() [2/2] . . . 178
10.20.3.7 Read() [1/2] . . . 178
10.20.3.8 Read() [2/2] . . . 178
10.20.3.9 Remove() [1/2] . . . 178
10.20.3.10 Remove() [2/2] . . . 179
10.20.3.11 Rename() [1/2] . . . 179
10.20.3.12 Rename() [2/2] . . . 179
10.20.3.13 SetupPath() [1/2] . . . 179
10.20.3.14 SetupPath() [2/2] . . . 179
10.20.3.15 Write() [1/2] . . . 180
10.20.3.16 Write() [2/2] . . . 180
10.21 sound4::cloudx1::dyn::helper::SampleFormat< T > Struct Template Reference . . . 180
10.21.1 Detailed Description . . . 180
10.22 sound4::cloudx1::helper::SampleFormat< T > Struct Template Reference . . . 181
10.22.1 Detailed Description . . . 181
10.23 sound4::helper::SampleFormat< T > Struct Template Reference . . . 181
10.23.1 Detailed Description . . . 181
10.24 sound4::cloudx1::dyn::helper::SampleFormat< float > Struct Reference . . . 182
10.24.1 Detailed Description . . . 182
10.24.2 Member Data Documentation . . . 182
10.24.2.1 format . . . 182
10.25 sound4::cloudx1::helper::SampleFormat< float > Struct Reference . . . 182
10.25.1 Detailed Description . . . 182

10.25.2 Member Data Documentation . . . 183
10.25.2.1 format . . . 183
10.26 sound4::helper::SampleFormat< float > Struct Reference . . . 183
10.26.1 Detailed Description . . . 183
10.26.2 Member Data Documentation . . . 183
10.26.2.1 format . . . 183
10.27 sound4::cloudx1::dyn::helper::SampleFormat< int16_t > Struct Reference . . . 183
10.27.1 Detailed Description . . . 184
10.27.2 Member Data Documentation . . . 184
10.27.2.1 format . . . 184
10.28 sound4::cloudx1::helper::SampleFormat< int16_t > Struct Reference . . . 184
10.28.1 Detailed Description . . . 184
10.28.2 Member Data Documentation . . . 184
10.28.2.1 format . . . 185
10.29 sound4::helper::SampleFormat< int16_t > Struct Reference . . . 185
10.29.1 Detailed Description . . . 185
10.29.2 Member Data Documentation . . . 185
10.29.2.1 format . . . 185
10.30 sound4::cloudx1::dyn::helper::SampleFormat< int32_t > Struct Reference . . . 185
10.30.1 Detailed Description . . . 186
10.30.2 Member Data Documentation . . . 186
10.30.2.1 format . . . 186
10.31 sound4::cloudx1::helper::SampleFormat< int32_t > Struct Reference . . . 186
10.31.1 Detailed Description . . . 186
10.31.2 Member Data Documentation . . . 186
10.31.2.1 format . . . 187
10.32 sound4::helper::SampleFormat< int32_t > Struct Reference . . . 187
10.32.1 Detailed Description . . . 187
10.32.2 Member Data Documentation . . . 187
10.32.2.1 format . . . 187
10.33 SOUND4_ProcessInfo Struct Reference . . . 187
10.33.1 Detailed Description . . . 188
10.33.2 Member Data Documentation . . . 188
10.33.2.1 name . . . 188
10.33.2.2 prefix . . . 188
10.33.2.3 shortname . . . 188
10.33.2.4 struct_size . . . 189
10.33.2.5 version . . . 189
10.34 sound4cl_SOUND4_ProcessInfo Struct Reference . . . 189
10.34.1 Detailed Description . . . 189
10.34.2 Member Data Documentation . . . 189
10.34.2.1 name . . . 190

10.34.2.2 prefix . . . 190
10.34.2.3 shortname . . . 190
10.34.2.4 struct_size . . . 190
10.34.2.5 version . . . 190

11 File Documentation **191**
11.1 sound4.x1.cloud-pipe-cxx.cpp File Reference . . . 191
11.1.1 Function Documentation . . . 191
11.1.1.1 GetEnv() . . . 191
11.1.1.2 main() . . . 192
11.1.1.3 MyLogger() . . . 192
11.1.1.4 ProcessLoop() . . . 192
11.1.1.5 Usage() . . . 192
11.2 sound4.x1.cloud-pipe-cxx.cpp . . . 193
11.3 sound4.x1.cloud-pipe-dyn.cpp File Reference . . . 194
11.3.1 Function Documentation . . . 195
11.3.1.1 GetEnv() . . . 195
11.3.1.2 main() . . . 196
11.3.1.3 MyLogger() . . . 196
11.3.1.4 ProcessLoop() . . . 196
11.3.1.5 Usage() . . . 197
11.4 sound4.x1.cloud-pipe-dyn.cpp . . . 197
11.5 sound4.x1.cloud-pipe.cpp File Reference . . . 200
11.5.1 Macro Definition Documentation . . . 200
11.5.1.1 CHANNEL_COUNT . . . 201
11.5.1.2 CHUNK_SIZE . . . 201
11.5.1.3 SAMPLE_RATE . . . 201
11.5.2 Function Documentation . . . 201
11.5.2.1 main() . . . 201
11.5.2.2 MyLogger() . . . 202
11.5.2.3 ProcessLoop() . . . 202
11.5.2.4 Usage() . . . 202
11.6 sound4.x1.cloud-pipe.cpp . . . 202
11.7 sound4.x1.cloud.h File Reference . . . 205
11.7.1 Macro Definition Documentation . . . 208
11.7.1.1 CLOUDX1_DECL . . . 208
11.7.1.2 CLOUDX1_HAS_CLOUDBUS . . . 208
11.7.1.3 CLOUDX1_HAS_WEBSERVER . . . 208
11.8 sound4.x1.cloud.h . . . 209
11.9 sound4.x1.cloud.hpp File Reference . . . 213
11.9.1 Macro Definition Documentation . . . 214
11.9.1.1 fs_strdup . . . 214

11.10 sound4.x1.cloud.hpp . . . 214
11.11 sound4.x1.cloud_dyn.hpp File Reference . . . 221
11.11.1 Detailed Description . . . 223
11.11.2 Macro Definition Documentation . . . 223
11.11.2.1 fs_strdup . . . 223
11.12 sound4.x1.cloud_dyn.hpp . . . 223
11.13 sound4cl-pipe.cpp File Reference . . . 236
11.13.1 Function Documentation . . . 237
11.13.1.1 GetEnv() . . . 237
11.13.1.2 main() . . . 237
11.13.1.3 ProcessLoop() . . . 238
11.14 sound4cl-pipe.cpp . . . 238
11.15 sound4cl.hpp File Reference . . . 241
11.15.1 Detailed Description . . . 242
11.16 sound4cl.hpp . . . 242
11.17 sound4cl_cdef.h File Reference . . . 255
11.17.1 Macro Definition Documentation . . . 257
11.17.1.1 fs_strdup . . . 258
11.17.1.2 SOUND4_INVALID_WEBSERVER_ID . . . 258
11.17.1.3 SOUND4_WEBSERVER_HTTP_OK . . . 258
11.17.1.4 SOUND4_WEBSERVER_HTTPS_OK . . . 258
11.17.2 Typedef Documentation . . . 258
11.17.2.1 fs_char . . . 258
11.17.2.2 sound4cl_loggerfn . . . 258
11.17.2.3 sound4cl_storage_exists . . . 259
11.17.2.4 sound4cl_storage_getall . . . 259
11.17.2.5 sound4cl_storage_getall_free . . . 259
11.17.2.6 sound4cl_storage_reader . . . 259
11.17.2.7 sound4cl_storage_reader_free . . . 259
11.17.2.8 sound4cl_storage_remove . . . 259
11.17.2.9 sound4cl_storage_rename . . . 260
11.17.2.10 sound4cl_storage_writer . . . 260
11.17.3 Enumeration Type Documentation . . . 260
11.17.3.1 sound4cl_LogSeverity . . . 260
11.17.3.2 sound4cl_PresetChange_Kind . . . 260
11.17.3.3 sound4cl_SampleFormat . . . 261
11.17.4 Function Documentation . . . 261
11.17.4.1 sound4cl_AddAudio() . . . 261
11.17.4.2 sound4cl_AddPadAudio() . . . 261
11.17.4.3 sound4cl_AudioConvertFrom() . . . 262
11.17.4.4 sound4cl_AudioConvertTo() . . . 262
11.17.4.5 sound4cl_AudioMonoFromLiveStereo() . . . 262

11.17.4.6 sound4cl_AudioMonoToLiveStereo() . . . 262
11.17.4.7 sound4cl_DeleteClient() . . . 262
11.17.4.8 sound4cl_ExitProcess() . . . 262
11.17.4.9 sound4cl_FreeBus() . . . 263
11.17.4.10 sound4cl_FreeJsonAnswer() . . . 263
11.17.4.11 sound4cl_FreeMetadataInfos() . . . 263
11.17.4.12 sound4cl_FreeParameters() . . . 263
11.17.4.13 sound4cl_FreeParameterValue() . . . 263
11.17.4.14 sound4cl_GetAudio() . . . 263
11.17.4.15 sound4cl_GetAudioInputCount() . . . 264
11.17.4.16 sound4cl_GetAudioOutputCount() . . . 264
11.17.4.17 sound4cl_GetBufferIn() . . . 264
11.17.4.18 sound4cl_GetBufferOut() . . . 264
11.17.4.19 sound4cl_GetBytesFromFormat() . . . 264
11.17.4.20 sound4cl_GetChannelCount() . . . 264
11.17.4.21 sound4cl_GetChunkSizeInFrames() . . . 264
11.17.4.22 sound4cl_GetEstimatedDelay() . . . 265
11.17.4.23 sound4cl_GetFormatFromName() . . . 265
11.17.4.24 sound4cl_GetFormatName() . . . 265
11.17.4.25 sound4cl_GetMaxPacketFrame() . . . 265
11.17.4.26 sound4cl_GetMetadataInfos() . . . 265
11.17.4.27 sound4cl_GetOutputCount() . . . 265
11.17.4.28 sound4cl_GetParameter() . . . 265
11.17.4.29 sound4cl_GetPossibleChunkSizeInFrames() . . . 266
11.17.4.30 sound4cl_GetProcessChunkFrames() . . . 266
11.17.4.31 sound4cl_GetSampleRate() . . . 266
11.17.4.32 sound4cl_GetVersion() . . . 266
11.17.4.33 sound4cl_InitProcess() . . . 266
11.17.4.34 sound4cl_InitProcess2() . . . 266
11.17.4.35 sound4cl_InitProcess3() . . . 267
11.17.4.36 sound4cl_MonoToStereo() . . . 267
11.17.4.37 sound4cl_MonoToStereo_Planar() . . . 267
11.17.4.38 sound4cl_NewBus() . . . 267
11.17.4.39 sound4cl_NewClient() . . . 267
11.17.4.40 sound4cl_NewParameters() . . . 267
11.17.4.41 sound4cl_PresetManager_InformChange() . . . 268
11.17.4.42 sound4cl_ProcessAudio() . . . 268
11.17.4.43 sound4cl_ProcessAudio_Planar() . . . 268
11.17.4.44 sound4cl_ProcessJson() . . . 268
11.17.4.45 sound4cl_SaveState() . . . 268
11.17.4.46 sound4cl_SetInstanceBus() . . . 268
11.17.4.47 sound4cl_SetLoggerCallback() . . . 269

11.17.4.48 sound4cl_SetLogSeverity() . . . 269
11.17.4.49 sound4cl_SetMetadata() . . . 269
11.17.4.50 sound4cl_SetMetadataMulti() . . . 269
11.17.4.51 sound4cl_SetParameter() . . . 269
11.17.4.52 sound4cl_SetPresetManager() . . . 269
11.17.4.53 sound4cl_SOUND4_GetProcessInfo() . . . 270
11.17.4.54 sound4cl_StartUpdateThread() . . . 270
11.17.4.55 sound4cl_StereoToMono() . . . 270
11.17.4.56 sound4cl_StereoToMono_Planar() . . . 270
11.17.4.57 sound4cl_StopUpdateThread() . . . 270
11.17.4.58 sound4cl_TerminateProcess() . . . 270
11.17.4.59 sound4cl_WaitUpdateThreadReady() . . . 271
11.17.4.60 sound4cl_Webserver() . . . 271
11.17.4.61 sound4cl_Webserver_FreeString() . . . 271
11.17.4.62 sound4cl_Webserver_GetAppHealth() . . . 271
11.17.4.63 sound4cl_Webserver_SetAppHealth() . . . 271
11.17.4.64 sound4cl_Webserver_Status() . . . 271
11.17.4.65 sound4cl_Webserver_Stop() . . . 272
11.17.4.66 sound4cl_Webserver_tcp() . . . 272
11.17.4.67 sound4cl_Webserver_tcp2() . . . 272
11.18 sound4cl_cdef.h . . . 272

Index **275**

Chapter 1

SOUND4 x1.CLOUD 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 cloudx1_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.x1.cloud-pipe.cpp

You can also use the C++ wrapper from header sound4.x1.cloud.hpp or the one with dynamic library loading sound4.x1.cloud_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 cloudx1_GetChunkSizeInFrames ()

Use cloudx1_GetPossibleChunkSizeInFrames

Member cloudx1_InitProcess (const char *LoginKey, const char *RadioName, const char *Access_Key_ID, const char *Access_Key_Secret, const char *save_path)

Use cloudx1_InitProcess3

Member cloudx1_InitProcess2 (const char *LoginKey, const char *RadioName, const char *Access_Key_ID, const char *Access_Key_Secret, const char *save_path, const struct cloudx1_CParameters *parameters)

Use cloudx1_InitProcess3

Chapter 3

Module Index

3.1 Modules

Here is a list of all modules:

Sample using C interface . . . 17
Environment variables and parameters . . . 19
Global setup . . . 20
Process creation/destruction . . . 26
Custom preset managagement . . . 31
Metadata Management . . . 37
Update Management . . . 39
Direct processing . . . 40
Audio managing (any audio) . . . 43
Audio convert helpers . . . 48
Client's JSon config interfacing . . . 52
JSON syntax . . . 54
JSON syntax for internal setup . . . 57
Web server . . . 61
Sample using C++ interface wrapper . . . 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::cloudx1 . . . 76
sound4::cloudx1::dyn . . . 79
sound4::cloudx1::dyn::helper . . . 82
sound4::cloudx1::helper . . . 83
sound4::helper . . . 83
std . . . 84

Chapter 5

Hierarchical Index

5.1 Class Hierarchy

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

sound4::CBus . . . 85
sound4::CInstance::CClient . . . 86
sound4::cloudx1::CInstance::CClient . . . 88
sound4::cloudx1::dyn::CInstance::CClient . . . 90
sound4::cloudx1::dyn::CDynLib . . . 92
sound4::cloudx1::dyn::helper::CDynLoader . . . 105
sound4::helper::CDynLoader . . . 108
sound4::CInstance . . . 111
sound4::cloudx1::CInstance . . . 122
sound4::cloudx1::dyn::CInstance . . . 133
cloudx1_CClientInstance . . . 144
cloudx1_CInstance . . . 144
cloudx1_CParameters . . . 145
sound4::cloudx1::CPresetLoader . . . 145
sound4::cloudx1::dyn::CPresetLoader . . . 147
MyPresetManager . . . 175
sound4::CPresetLoader . . . 149
MyPresetManager . . . 175
sound4::CProcessor . . . 152
sound4::cloudx1::dyn::helper::DynFuncHolder< T > . . . 172
sound4::helper::DynFuncHolder< T > . . . 174
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_AddAudio) > . . . 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_AddPadAudio) > . . . 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_AudioConvertFrom) > . . . 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_AudioConvertTo) > . . . 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_AudioMonoFromLiveStereo) > . . . 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_AudioMonoToLiveStereo) > . . . 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_DeleteClient) > . . . 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_ExitProcess) > . . . 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_FreeJsonAnswer) > . . . 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_FreeMetadataInfos) > . . . 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_FreeParameters) > . . . 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_FreeParameterValue) > . . . 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_GetAudio) > . . . 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_GetAudioInputCount) > . . . 172

sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_GetAudioOutputCount) > 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_GetBufferIn) > 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_GetBufferOut) > 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_GetBytesFromFormat) > 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_GetChannelCount) > 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_GetChunkSizeInFrames) > 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_GetEstimatedDelay) > 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_GetFormatFromName) > 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_GetFormatName) > 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_GetMaxPacketFrame) > 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_GetMetadataInfos) > 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_GetOutputCount) > 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_GetParameter) > 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_GetPossibleChunkSizeInFrames) > . 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_GetProcessChunkFrames) > 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_GetSampleRate) > 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_GetVersion) > 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_InitProcess) > 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_InitProcess2) > 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_InitProcess3) > 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_MonoToStereo) > 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_MonoToStereo_Planar) > 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_NewClient) > 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_NewParameters) > 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_PresetManager_InformChange) > . . 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_ProcessAudio) > 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_ProcessAudio_Planar) > 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_ProcessJson) > 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_SaveState) > 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_SetLoggerCallback) > 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_SetLogSeverity) > 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_SetMetadata) > 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_SetMetadataMulti) > 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_SetParameter) > 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_SetPresetManager) > 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_StartUpdateThread) > 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_StereoToMono) > 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_StereoToMono_Planar) > 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_StopUpdateThread) > 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_TerminateProcess) > 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_WaitUpdateThreadReady) > 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_Webserver) > 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_Webserver_FreeString) > 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_Webserver_GetAppHealth) > 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_Webserver_SetAppHealth) > 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_Webserver_Status) > 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_Webserver_Stop) > 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_Webserver_tcp) > 172
sound4::cloudx1::dyn::helper::DynFuncHolder< decltype(cloudx1_Webserver_tcp2) > 172
sound4::helper::DynFuncHolder< decltype(sound4cl_AddAudio) > 174
sound4::helper::DynFuncHolder< decltype(sound4cl_AddPadAudio) > 174
sound4::helper::DynFuncHolder< decltype(sound4cl_AudioConvertFrom) > 174
sound4::helper::DynFuncHolder< decltype(sound4cl_AudioConvertTo) > 174
sound4::helper::DynFuncHolder< decltype(sound4cl_AudioMonoFromLiveStereo) > 174
sound4::helper::DynFuncHolder< decltype(sound4cl_AudioMonoToLiveStereo) > 174
sound4::helper::DynFuncHolder< decltype(sound4cl_DeleteClient) > 174
sound4::helper::DynFuncHolder< decltype(sound4cl_ExitProcess) > 174
sound4::helper::DynFuncHolder< decltype(sound4cl_FreeBus) > 174

sound4::helper::DynFuncHolder< decltype(sound4cl_FreeJsonAnswer) > . . . 174
sound4::helper::DynFuncHolder< decltype(sound4cl_FreeMetadataInfos) > . . . 174
sound4::helper::DynFuncHolder< decltype(sound4cl_FreeParameters) > . . . 174
sound4::helper::DynFuncHolder< decltype(sound4cl_FreeParameterValue) > . . . 174
sound4::helper::DynFuncHolder< decltype(sound4cl_GetAudio) > . . . 174
sound4::helper::DynFuncHolder< decltype(sound4cl_GetAudioInputCount) > . . . 174
sound4::helper::DynFuncHolder< decltype(sound4cl_GetAudioOutputCount) > . . . 174
sound4::helper::DynFuncHolder< decltype(sound4cl_GetBufferIn) > . . . 174
sound4::helper::DynFuncHolder< decltype(sound4cl_GetBufferOut) > . . . 174
sound4::helper::DynFuncHolder< decltype(sound4cl_GetBytesFromFormat) > . . . 174
sound4::helper::DynFuncHolder< decltype(sound4cl_GetChannelCount) > . . . 174
sound4::helper::DynFuncHolder< decltype(sound4cl_GetChunkSizeInFrames) > . . . 174
sound4::helper::DynFuncHolder< decltype(sound4cl_GetEstimatedDelay) > . . . 174
sound4::helper::DynFuncHolder< decltype(sound4cl_GetFormatFromName) > . . . 174
sound4::helper::DynFuncHolder< decltype(sound4cl_GetFormatName) > . . . 174
sound4::helper::DynFuncHolder< decltype(sound4cl_GetMaxPacketFrame) > . . . 174
sound4::helper::DynFuncHolder< decltype(sound4cl_GetMetadataInfos) > . . . 174
sound4::helper::DynFuncHolder< decltype(sound4cl_GetOutputCount) > . . . 174
sound4::helper::DynFuncHolder< decltype(sound4cl_GetParameter) > . . . 174
sound4::helper::DynFuncHolder< decltype(sound4cl_GetPossibleChunkSizeInFrames) > . . . 174
sound4::helper::DynFuncHolder< decltype(sound4cl_GetProcessChunkFrames) > . . . 174
sound4::helper::DynFuncHolder< decltype(sound4cl_GetSampleRate) > . . . 174
sound4::helper::DynFuncHolder< decltype(sound4cl_GetVersion) > . . . 174
sound4::helper::DynFuncHolder< decltype(sound4cl_InitProcess) > . . . 174
sound4::helper::DynFuncHolder< decltype(sound4cl_InitProcess2) > . . . 174
sound4::helper::DynFuncHolder< decltype(sound4cl_InitProcess3) > . . . 174
sound4::helper::DynFuncHolder< decltype(sound4cl_MonoToStereo) > . . . 174
sound4::helper::DynFuncHolder< decltype(sound4cl_MonoToStereo_Planar) > . . . 174
sound4::helper::DynFuncHolder< decltype(sound4cl_NewBus) > . . . 174
sound4::helper::DynFuncHolder< decltype(sound4cl_NewClient) > . . . 174
sound4::helper::DynFuncHolder< decltype(sound4cl_NewParameters) > . . . 174
sound4::helper::DynFuncHolder< decltype(sound4cl_PresetManager_InformChange) > . . . 174
sound4::helper::DynFuncHolder< decltype(sound4cl_ProcessAudio) > . . . 174
sound4::helper::DynFuncHolder< decltype(sound4cl_ProcessAudio_Planar) > . . . 174
sound4::helper::DynFuncHolder< decltype(sound4cl_ProcessJson) > . . . 174
sound4::helper::DynFuncHolder< decltype(sound4cl_SaveState) > . . . 174
sound4::helper::DynFuncHolder< decltype(sound4cl_SetInstanceBus) > . . . 174
sound4::helper::DynFuncHolder< decltype(sound4cl_SetLoggerCallback) > . . . 174
sound4::helper::DynFuncHolder< decltype(sound4cl_SetLogSeverity) > . . . 174
sound4::helper::DynFuncHolder< decltype(sound4cl_SetMetadata) > . . . 174
sound4::helper::DynFuncHolder< decltype(sound4cl_SetMetadataMulti) > . . . 174
sound4::helper::DynFuncHolder< decltype(sound4cl_SetParameter) > . . . 174
sound4::helper::DynFuncHolder< decltype(sound4cl_SetPresetManager) > . . . 174
sound4::helper::DynFuncHolder< decltype(sound4cl_StartUpdateThread) > . . . 174
sound4::helper::DynFuncHolder< decltype(sound4cl_StereoToMono) > . . . 174
sound4::helper::DynFuncHolder< decltype(sound4cl_StereoToMono_Planar) > . . . 174
sound4::helper::DynFuncHolder< decltype(sound4cl_StopUpdateThread) > . . . 174
sound4::helper::DynFuncHolder< decltype(sound4cl_TerminateProcess) > . . . 174
sound4::helper::DynFuncHolder< decltype(sound4cl_WaitUpdateThreadReady) > . . . 174
sound4::helper::DynFuncHolder< decltype(sound4cl_Webserver) > . . . 174
sound4::helper::DynFuncHolder< decltype(sound4cl_Webserver_FreeString) > . . . 174
sound4::helper::DynFuncHolder< decltype(sound4cl_Webserver_GetAppHealth) > . . . 174
sound4::helper::DynFuncHolder< decltype(sound4cl_Webserver_SetAppHealth) > . . . 174
sound4::helper::DynFuncHolder< decltype(sound4cl_Webserver_Status) > . . . 174
sound4::helper::DynFuncHolder< decltype(sound4cl_Webserver_Stop) > . . . 174
sound4::helper::DynFuncHolder< decltype(sound4cl_Webserver_tcp) > . . . 174
sound4::helper::DynFuncHolder< decltype(sound4cl_Webserver_tcp2) > . . . 174
sound4::cloudx1::dyn::helper::SampleFormat< T > . . . 180

sound4::cloudx1::helper::SampleFormat< T > 181
sound4::helper::SampleFormat< T > 181
sound4::cloudx1::dyn::helper::SampleFormat< float > 182
sound4::cloudx1::helper::SampleFormat< float > 182
sound4::helper::SampleFormat< float > 183
sound4::cloudx1::dyn::helper::SampleFormat< int16_t > 183
sound4::cloudx1::helper::SampleFormat< int16_t > 184
sound4::helper::SampleFormat< int16_t > 185
sound4::cloudx1::dyn::helper::SampleFormat< int32_t > 185
sound4::cloudx1::helper::SampleFormat< int32_t > 186
sound4::helper::SampleFormat< int32_t > 187
SOUND4_ProcessInfo 187
sound4cl_SOUND4_ProcessInfo 189

Chapter 6

Class Index

6.1 Class List

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

sound4::CBus
: Shared Bus . . . 85
sound4::CInstance::CClient . . . 86
sound4::cloudx1::CInstance::CClient . . . 88
sound4::cloudx1::dyn::CInstance::CClient . . . 90
sound4::cloudx1::dyn::CDynLib
: Dynamic library interface . . . 92
sound4::cloudx1::dyn::helper::CDynLoader
: Helper to load dynamic library . . . 105
sound4::helper::CDynLoader
: Helper to load dynamic library . . . 108
sound4::CInstance
: Instance handling class . . . 111
sound4::cloudx1::CInstance . . . 122
sound4::cloudx1::dyn::CInstance . . . 133
cloudx1_CClientInstance . . . 144
cloudx1_CInstance . . . 144
cloudx1_CParameters . . . 145
sound4::cloudx1::CPresetLoader . . . 145
sound4::cloudx1::dyn::CPresetLoader . . . 147
sound4::CPresetLoader
: Custom preset handler helper . . . 149
sound4::CProcessor
: Dynamic library interface . . . 152
sound4::cloudx1::dyn::helper::DynFuncHolder< T >
: Helper to wrap a function pointer . . . 172
sound4::helper::DynFuncHolder< T >
: Helper to wrap a function pointer . . . 174
MyPresetManager
: [Custom preset manager] . . . 175
sound4::cloudx1::dyn::helper::SampleFormat< T >
: Helper for Sample format types . . . 180
sound4::cloudx1::helper::SampleFormat< T >
: Helper for Sample format types . . . 181
sound4::helper::SampleFormat< T >
: Helper for Sample format types . . . 181

sound4::cloudx1::dyn::helper::SampleFormat< float > 182
sound4::cloudx1::helper::SampleFormat< float > 182
sound4::helper::SampleFormat< float > 183
sound4::cloudx1::dyn::helper::SampleFormat< int16_t > 183
sound4::cloudx1::helper::SampleFormat< int16_t > 184
sound4::helper::SampleFormat< int16_t > 185
sound4::cloudx1::dyn::helper::SampleFormat< int32_t > 185
sound4::cloudx1::helper::SampleFormat< int32_t > 186
sound4::helper::SampleFormat< int32_t > 187
SOUND4_ProcessInfo 187
sound4cl_SOUND4_ProcessInfo 189

Chapter 7

File Index

7.1 File List

Here is a list of all files with brief descriptions:

sound4.x1.cloud-pipe-cxx.cpp
C++ sample using C++ interface wrapper . . . 191
sound4.x1.cloud-pipe-dyn.cpp
C++ sample using C++ interface wrapper with dynamic library loading . . . 194
sound4.x1.cloud-pipe.cpp
C++ sample using only C interface . . . 200
sound4.x1.cloud.h
C interface for library . . . 205
sound4.x1.cloud.hpp
C++ wrapper for library . . . 213
sound4.x1.cloud_dyn.hpp
C++ wrapper for dynamic loading library . . . 221
sound4cl-pipe.cpp
C++ sample using C++ interface wrapper with dynamic library loading Works with any SOUND4 processor provided . . . 236
sound4cl.hpp
C++ wrapper for dynamic loading library . . . 241
sound4cl_cdef.h
C function signature definition for all sound4.cl libraries . . . 255

Chapter 8

Module Documentation

8.1 Sample using C interface

See full example cpp/sound4.x1.cloud-pipe.cpp

8.1.1 Sample to create the processing instance

see Process creation/destruction and Update Management

```
    // Setup parameters
    struct cloudx1_CParameters *params = cloudx1_NewParameters();
    // Optional: activate web authentication, set admin password
    // This creates the authentication only for a new storage path
    cloudx1_SetParameter(params,"ADMIN_USER", "admin");
    cloudx1_SetParameter(params,"ADMIN_SECRET", "admin");

    // Create the processing instance
    pInstance=cloudx1_InitProcess2(pLoginKey,pRadioName,pKeyId,pAccessKey,store_path.c_str(), params);
    cloudx1_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 « "cloudx1_StartUpdateThread (no json port)" « std::endl;
       cloudx1_StartUpdateThread(pInstance, 0);
       std::cerr « "cloudx1_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 (cloudx1_WaitUpdateThreadReady(pInstance, 5000)<0) {
        std::cerr « "Timeout to create the process instance"« std::endl;
    }
    std::cerr « "Waiting update thread DONE" « std::endl;
#if (CLOUDX1_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 = cloudx1_Webserver(listenport,listenport_secure,pInstance);
        if (webserver==SOUND4_INVALID_WEBSERVER_ID) {
            std::cerr « "Starting web server FAILED" « std::endl;
        } else {
            int web_status=cloudx1_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 // (CLOUDX1_HAS_WEBSERVER)

     // Force load a factory preset now
     // Could be used instead of setting a store_path
     /*{
         cloudx1_CClientInstance *pClient=cloudx1_NewClient(pInstance);
         const char *answer=cloudx1_ProcessJson(pClient, "{ \"factoryload\": \"All Programs\" }", NULL);
         std::cerr « "Loaded factory preset answer: " « answer« std::endl;
         cloudx1_FreeJsonAnswer(answer);
         cloudx1_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
        cloudx1_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 (CLOUDX1_HAS_WEBSERVER)
    // Stopping the webserver
    std::cerr « "Stopping web server" « std::endl;
    cloudx1_Webserver_Stop(webserver, 1000);
    std::cerr « "Stopping web server DONE" « std::endl;
#endif // (CLOUDX1_HAS_WEBSERVER)

    // Stopping the processing instance
    std::cerr « "Stopping Update thread" « std::endl;
    cloudx1_StopUpdateThread(pInstance); // this will exit the update thread
    std::cerr « "Stopping Update thread DONE" « std::endl;
    updater.join();
    cloudx1_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 cloudx1_InitProcess2

8.2.1.1 Webserver authentification setup :

- `SETUP_USER` see cloudx1_InitProcess
- `SETUP_SECRET` see cloudx1_InitProcess
- `ADMIN_USER` see cloudx1_InitProcess
- `ADMIN_SECRET` see cloudx1_InitProcess

8.2.1.2 Webserver secure :

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

8.2.1.3 Network access :

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

8.2.2 Parameters

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

Run mode:

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

8.2.3 Preset sharing

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

In this case, you should use a different `STATE_FILENAME` per instance so they keep their settings and on air presets separated. If not, each time settings or onair preset are changed, the changes will be reflected to the other 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 CLOUDX1_AUDIOFRAME_COUNT 64
- #define CLOUDX1_CHANNEL_COUNT 2
- #define CLOUDX1_AUDIO_INPUT_COUNT 1
- #define CLOUDX1_AUDIO_OUTPUT_COUNT 1
- #define CLOUDX1_SAMPLE_RATE 48000

Typedefs

- typedef void(∗ cloudx1_loggerfn) (enum cloudx1_LogSeverity, const char ∗)

Enumerations

- enum cloudx1_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 ∗ cloudx1_GetVersion ()
- unsigned int cloudx1_GetChunkSizeInFrames ()
- unsigned int ∗ cloudx1_GetPossibleChunkSizeInFrames ()
- unsigned int cloudx1_GetChannelCount ()
- unsigned int cloudx1_GetAudioInputCount ()
- unsigned int cloudx1_GetAudioOutputCount ()
- unsigned int cloudx1_GetSampleRate ()
- void cloudx1_SetLoggerCallback (cloudx1_loggerfn logger)
- void cloudx1_SetLogSeverity (enum cloudx1_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 CLOUDX1_AUDIO_INPUT_COUNT

```
#define CLOUDX1_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 cloudx1_GetAudioInputCount()

Definition at line 170 of file sound4.x1.cloud.h.

8.3.2.2 CLOUDX1_AUDIO_OUTPUT_COUNT

```
#define CLOUDX1_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 cloudx1_GetAudioOutputCount()

Definition at line 176 of file sound4.x1.cloud.h.

8.3.2.3 CLOUDX1_AUDIOFRAME_COUNT

```
#define CLOUDX1_AUDIOFRAME_COUNT 64
```

This version number of frames in a process chunk.

Remarks

For binary compatibility, this should be runtime checked to be the same as cloudx1_GetChunkSizeInFrames()

Definition at line 158 of file sound4.x1.cloud.h.

8.3.2.4 CLOUDX1_CHANNEL_COUNT

```
#define CLOUDX1_CHANNEL_COUNT 2
```

This version number of channels.

Remarks

For binary compatibility, this should be runtime checked to be the same as cloudx1_GetChannelCount()

Definition at line 164 of file sound4.x1.cloud.h.

8.3.2.5 CLOUDX1_SAMPLE_RATE

```
#define CLOUDX1_SAMPLE_RATE 48000
```

This input/output sample-rate.

Remarks

For binary compatibility, this should be runtime checked to be the same as cloudx1_GetSampleRate()

Definition at line 182 of file sound4.x1.cloud.h.

8.3.3 Typedef Documentation

8.3.3.1 cloudx1_loggerfn

```
typedef void(* cloudx1_loggerfn) (enum cloudx1_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 297 of file sound4.x1.cloud.h.

8.3.4 Enumeration Type Documentation

8.3.4.1 cloudx1_LogSeverity

```
enum cloudx1_LogSeverity
```

The logging levels

Enumerator

none	
fatal	
error	
warning	
info	
verbose	
verbose2	
verbose3	
verbose4	
verbose5	

Definition at line 278 of file sound4.x1.cloud.h.

8.3.5 Function Documentation

8.3.5.1 cloudx1_GetAudioInputCount()

```
unsigned int cloudx1_GetAudioInputCount ( )
```

Get the number of inputs for this process

Remarks

All inputs have cloudx1_GetChannelCount() channels

Fixed to 1, but should be checked for backward compatibility

Returns

The process input count

8.3.5.2 cloudx1_GetAudioOutputCount()

```
unsigned int cloudx1_GetAudioOutputCount ( )
```

Get the number of outputs for this process

Remarks

All outputs have cloudx1_GetChannelCount() channels

Fixed to 1, but should be checked for backward compatibility

Returns

The process outputs count

8.3.5.3 cloudx1_GetChannelCount()

```
unsigned int cloudx1_GetChannelCount ( )
```

Get the number of channels needed by this process

Remarks

Fixed to 2, but should be checked for backward compatibility

Returns

The process channels count

8.3.5.4 cloudx1_GetChunkSizeInFrames()

```
unsigned int cloudx1_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 cloudx1_GetChannelCount samples

Remarks

Fixed to 64, but should be checked for backward compatibility

Returns

The process chunk size in frame

Deprecated Use cloudx1_GetPossibleChunkSizeInFrames

8.3.5.5 cloudx1_GetPossibleChunkSizeInFrames()

```
unsigned int* cloudx1_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 cloudx1_GetChannelCount samples

Returns

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

8.3.5.6 cloudx1_GetSampleRate()

```
unsigned int cloudx1_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 cloudx1_GetVersion()

```
const char* cloudx1_GetVersion ( )
```

Get the process version string

Returns

version string

8.3.5.8 cloudx1_SetLoggerCallback()

```
void cloudx1_SetLoggerCallback (
            cloudx1_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 cloudx1_SetLogSeverity()

```
void cloudx1_SetLogSeverity (
            enum cloudx1_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 cloudx1_CInstance
- struct cloudx1_CParameters

Functions

- struct cloudx1_CParameters * cloudx1_NewParameters ()
- void cloudx1_FreeParameters (struct cloudx1_CParameters *params)
- void cloudx1_SetParameter (struct cloudx1_CParameters *params, const char *name, const char *value)
- const char * cloudx1_GetParameter (struct cloudx1_CParameters *params, const char *name)
- void cloudx1_FreeParameterValue (const char *value)
- struct cloudx1_CInstance * cloudx1_InitProcess (const char *LoginKey, const char *RadioName, const char *Access_Key_ID, const char *Access_Key_Secret, const char *save_path)
- struct cloudx1_CInstance * cloudx1_InitProcess2 (const char *LoginKey, const char *RadioName, const char *Access_Key_ID, const char *Access_Key_Secret, const char *save_path, const struct cloudx1_CParameters *parameters)
- struct cloudx1_CInstance * cloudx1_InitProcess3 (const char *LoginKey, const char *RadioName, const char *Access_Key_ID, const char *Access_Key_Secret, const char *save_path, const struct cloudx1_CParameters *parameters, unsigned int frames_per_chunk)
- unsigned int cloudx1_GetProcessChunkFrames (struct cloudx1_CInstance *instance)
- int cloudx1_TerminateProcess (struct cloudx1_CInstance *instance)
- void cloudx1_ExitProcess (struct cloudx1_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 cloudx1_ExitProcess()

```
void cloudx1_ExitProcess (
            struct cloudx1_CInstance * instance )
```

Quit and destroy the instance

Parameters

instance	the instance to destroy

8.4.2.2 cloudx1_FreeParameters()

```
void cloudx1_FreeParameters (
            struct cloudx1_CParameters * params )
```

Free the parameters created with cloudx1_NewParameters.

8.4.2.3 cloudx1_FreeParameterValue()

```
void cloudx1_FreeParameterValue (
            const char * value )
```

Free a parameter value returned by cloudx1_GetParameter

Parameters

value	the returned value

8.4.2.4 cloudx1_GetParameter()

```
const char* cloudx1_GetParameter (
            struct cloudx1_CParameters * params,
            const char * name )
```

Get current parameter value.

Returned value must be freed with cloudx1_FreeParameterValue

8.4.2.5 cloudx1_GetProcessChunkFrames()

```
unsigned int cloudx1_GetProcessChunkFrames (
            struct cloudx1_CInstance * instance )
```

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

The value reflects the frames_per_chunk parameter passed to cloudx1_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 cloudx1_GetChannelCount samples

8.4.2.6 cloudx1_InitProcess()

```
struct cloudx1_CInstance* cloudx1_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 cloudx1_InitProcess2, but with default parameters (from env)

Deprecated Use cloudx1_InitProcess3

8.4.2.7 cloudx1_InitProcess2()

```
struct cloudx1_CInstance* cloudx1_InitProcess2 (
            const char * LoginKey,
            const char * RadioName,
            const char * Access_Key_ID,
            const char * Access_Key_Secret,
            const char * save_path,
            const struct cloudx1_CParameters * parameters )
```

Create a process instance with default frames_per_chunk (64)

Same as cloudx1_InitProcess3, but with default frames_per_chunk

Deprecated Use cloudx1_InitProcess3

8.4.2.8 cloudx1_InitProcess3()

```
struct cloudx1_CInstance* cloudx1_InitProcess3 (
            const char * LoginKey,
            const char * RadioName,
            const char * Access_Key_ID,
            const char * Access_Key_Secret,
            const char * save_path,
            const struct cloudx1_CParameters * parameters,
            unsigned int frames_per_chunk )
```

Create a process instance

The output will have to be freed with cloudx1_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 cloudx1_GetPossibleChunkSizeInFrames

Returns

A new process instance

8.4.2.9 cloudx1_NewParameters()

```
struct cloudx1_CParameters* cloudx1_NewParameters ( )
```

Create parameters to use with cloudx1_InitProcess2.

The output will have to be freed with cloudx1_FreeParameters

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

8.4.2.10 cloudx1_SetParameter()

```
void cloudx1_SetParameter (
            struct cloudx1_CParameters * params,
            const char * name,
            const char * value )
```

Set a new parameter.

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

See also Custom preset managagement

8.4.2.11 cloudx1_TerminateProcess()

```
int cloudx1_TerminateProcess (
            struct cloudx1_CInstance * instance )
```

Terminate the processing

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

Parameters

instance	the instance to terminate

Returns

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

8.5 Custom preset managagement

This allows to provide a customized preset management.

Typedefs

- typedef wchar_t fs_char
- typedef char ∗(∗ cloudx1_storage_reader) (const fs_char ∗filename, void ∗userdata)
- typedef void(∗ cloudx1_storage_reader_free) (char ∗content, void ∗userdata)
- typedef int(∗ cloudx1_storage_writer) (const fs_char ∗filename, const char ∗content, void ∗userdata)
- typedef int(∗ cloudx1_storage_exists) (const fs_char ∗filename, void ∗userdata)
- typedef fs_char ∗∗(∗ cloudx1_storage_getall) (void ∗userdata)
- typedef void(∗ cloudx1_storage_getall_free) (fs_char ∗∗all, void ∗userdata)
- typedef int(∗ cloudx1_storage_remove) (const fs_char ∗filename, void ∗userdata)
- typedef int(∗ cloudx1_storage_rename) (const fs_char ∗from, const fs_char ∗to, void ∗userdata)

Enumerations

- enum cloudx1_PresetChange_Kind { change_kind_created = 1 , change_kind_modified , change_kind_deleted }

Functions

- void cloudx1_SetPresetManager (struct cloudx1_CParameters *params, cloudx1_storage_reader reader, cloudx1_storage_reader_free, cloudx1_storage_writer writer, cloudx1_storage_exists exists, cloudx1_storage_getall getall, cloudx1_storage_getall_free getall_free, cloudx1_storage_remove remove, cloudx1_storage_rename rename, int IsReadOnly, void *userdata)
- void cloudx1_PresetManager_InformChange (struct cloudx1_CInstance *instance, const fs_char *relative_path, enum cloudx1_PresetChange_Kind change_kind)

8.5.1 Detailed Description

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

By default, the library store data on disk or in memory (see `save_path` parameter in cloudx1_InitProcess2), but you may use these functions to store data.

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

Known files:

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

Attention

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

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

8.5.2 Typedef Documentation

8.5.2.1 cloudx1_storage_exists

```
typedef int(* cloudx1_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 cloudx1_SetPresetManager

Returns

-1 if not exists, 0 if it exists

Definition at line 604 of file sound4.x1.cloud.h.

8.5.2.2 cloudx1_storage_getall

```
typedef fs_char**(* cloudx1_storage_getall) (void *userdata)
```

Function to call to get all stored files' name. Result has to be freed with cloudx1_storage_getall_free callback

Parameters

userdata	the userdata passed to cloudx1_SetPresetManager

Returns

A NULL terminated array of file names

Definition at line 613 of file sound4.x1.cloud.h.

8.5.2.3 cloudx1_storage_getall_free

```
typedef void(* cloudx1_storage_getall_free) (fs_char **all, void *userdata)
```

Function to call to free the result of getall.

Parameters

all	the returned value from cloudx1_storage_getall
userdata	the userdata passed to cloudx1_SetPresetManager

Definition at line 621 of file sound4.x1.cloud.h.

8.5.2.4 cloudx1_storage_reader

```
typedef char*(* cloudx1_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 cloudx1_SetPresetManager

Returns

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

Definition at line 577 of file sound4.x1.cloud.h.

8.5.2.5 cloudx1_storage_reader_free

```
typedef void(* cloudx1_storage_reader_free) (char *content, void *userdata)
```

Function to call to free the result of cloudx1_storage_reader.

Parameters

content	the returned value from cloudx1_storage_reader
userdata	the userdata passed to cloudx1_SetPresetManager

Definition at line 585 of file sound4.x1.cloud.h.

8.5.2.6 cloudx1_storage_remove

```
typedef int(* cloudx1_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 cloudx1_SetPresetManager

Returns

-1 if failed, 0 if OK

Definition at line 630 of file sound4.x1.cloud.h.

8.5.2.7 cloudx1_storage_rename

```
typedef int(* cloudx1_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 cloudx1_SetPresetManager

Returns

-1 if failed, 0 if OK

Definition at line 640 of file sound4.x1.cloud.h.

8.5.2.8 cloudx1_storage_writer

```
typedef int(* cloudx1_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 cloudx1_SetPresetManager

Returns

-1 if failed, 0 if OK

Definition at line 595 of file sound4.x1.cloud.h.

8.5.2.9 fs_char

```
typedef wchar_t fs_char
```

Definition at line 565 of file sound4.x1.cloud.h.

8.5.3 Enumeration Type Documentation

8.5.3.1 cloudx1_PresetChange_Kind

```
enum cloudx1_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 673 of file sound4.x1.cloud.h.

8.5.4 Function Documentation

8.5.4.1 cloudx1_PresetManager_InformChange()

```
void cloudx1_PresetManager_InformChange (
            struct cloudx1_CInstance * instance,
            const fs_char * relative_path,
            enum cloudx1_PresetChange_Kind change_kind )
```

Inform that a change happened in the storage. This must be used only in conjonction with cloudx1_SetPresetManager

Parameters

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

8.5.4.2 cloudx1_SetPresetManager()

```
void cloudx1_SetPresetManager (
            struct cloudx1_CParameters * params,
            cloudx1_storage_reader reader,
            cloudx1_storage_reader_free ,
            cloudx1_storage_writer writer,
            cloudx1_storage_exists exists,
            cloudx1_storage_getall getall,
            cloudx1_storage_getall_free getall_free,
            cloudx1_storage_remove remove,
            cloudx1_storage_rename rename,
```

```
            int IsReadOnly,
            void * userdata )
```

Setup a custom preset manager. This overrides the save_path in cloudx1_InitProcess2

To dynamically update when storage change, see cloudx1_PresetManager_InformChange

Parameters

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

8.6 Metadata Management

Functions

- void cloudx1_SetMetadata (struct cloudx1_CInstance *instance, const char *key, const char *value)
- void cloudx1_SetMetadataMulti (struct cloudx1_CInstance *instance, const char **keyvalue)
- const char ** cloudx1_GetMetadataInfos (struct cloudx1_CInstance *instance)
- void cloudx1_FreeMetadataInfos (struct cloudx1_CInstance *instance, const char **infos)

8.6.1 Detailed Description

Metadata handling.

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

8.6.2 Function Documentation

8.6.2.1 cloudx1_FreeMetadataInfos()

```
void cloudx1_FreeMetadataInfos (
            struct cloudx1_CInstance * instance,
            const char ** infos )
```

Frees the information returned by cloudx1_GetMetadataInfos.

Parameters

instance	the instance
infos	the infos

8.6.2.2 cloudx1_GetMetadataInfos()

```
const char** cloudx1_GetMetadataInfos (
            struct cloudx1_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 cloudx1_FreeMetadataInfos

8.6.2.3 cloudx1_SetMetadata()

```
void cloudx1_SetMetadata (
            struct cloudx1_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

cloudx1_SetMetadataMulti

Parameters

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

8.6.2.4 cloudx1_SetMetadataMulti()

```
void cloudx1_SetMetadataMulti (
            struct cloudx1_CInstance * instance,
            const char ** keyvalue )
```

Adds multiple metadata to the process.

This works like cloudx1_SetMetadata but with multiple metadata at once.

See also

cloudx1_SetMetadata

Parameters

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

8.7 Update Management

Functions

- void cloudx1_StartUpdateThread (struct cloudx1_CInstance *instance, unsigned int port)
- void cloudx1_StopUpdateThread (struct cloudx1_CInstance *instance)
- int cloudx1_WaitUpdateThreadReady (struct cloudx1_CInstance *instance, int milliseconds)

8.7.1 Detailed Description

Manage the process update.

8.7.2 Function Documentation

8.7.2.1 cloudx1_StartUpdateThread()

```
void cloudx1_StartUpdateThread (
            struct cloudx1_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 cloudx1_StopUpdateThread

Parameters

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

8.7.2.2 cloudx1_StopUpdateThread()

```
void cloudx1_StopUpdateThread (
            struct cloudx1_CInstance * instance )
```

Stop the update thread

Parameters

instance	the processing instance

8.7.2.3 cloudx1_WaitUpdateThreadReady()

```
int cloudx1_WaitUpdateThreadReady (
            struct cloudx1_CInstance * instance,
            int milliseconds )
```

Wait for the update thread to be ready

Parameters

instance	the processing instance
milliseconds	maximum time to wait in milliseconds

Returns

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

8.8 Direct processing

Functions

- void cloudx1_ProcessAudio (struct cloudx1_CInstance *instance, const float *input, float *output)
- void cloudx1_ProcessAudio_Planar (struct cloudx1_CInstance *instance, float const *const *input, float *const *output)
- float * cloudx1_GetBufferIn (struct cloudx1_CInstance *instance)
- float * cloudx1_GetBufferOut (struct cloudx1_CInstance *instance)
- unsigned int cloudx1_GetEstimatedDelay (struct cloudx1_CInstance *instance)

8.8.1 Detailed Description

Optimized processing with direct in->out.

8.8.2 Function Documentation

8.8.2.1 cloudx1_GetBufferIn()

```
float* cloudx1_GetBufferIn (
            struct cloudx1_CInstance * instance )
```

Get preallocated aligned buffer for audio input for cloudx1_ProcessAudio()

The buffer is a float[256 * cloudx1_GetAudioInputCount() * cloudx1_GetChannelCount()]

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

Note

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

Parameters

instance	the processing instance

Returns

An audio input dedicated buffer.

8.8.2.2 cloudx1_GetBufferOut()

```
float* cloudx1_GetBufferOut (
            struct cloudx1_CInstance * instance )
```

Get preallocated aligned buffer for audio output for cloudx1_ProcessAudio()

The buffer is a float[256 * cloudx1_GetAudioOutputCount() * cloudx1_GetChannelCount()]

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

Note

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

Parameters

instance	the processing instance

Returns

An audio output dedicated buffer.

8.8.2.3 cloudx1_GetEstimatedDelay()

```
unsigned int cloudx1_GetEstimatedDelay (
            struct cloudx1_CInstance * instance )
```

Get the delay (estimated) of current processing algorithm

Remarks

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

Returns

Estimated delay in frames

8.8.2.4 cloudx1_ProcessAudio()

```
void cloudx1_ProcessAudio (
            struct cloudx1_CInstance * instance,
            const float * input,
            float * output )
```

Process an audio chunk

A chunk contains cloudx1_GetProcessChunkFrames frames, set at init by frames_per_chunk in cloudx1_InitProcess3

Input has cloudx1_GetAudioInputCount() ∗ cloudx1_GetChannelCount() channels

Ouput has cloudx1_GetAudioOutputCount() ∗ cloudx1_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 cloudx1_GetBufferIn
output	audio output buffer (64 byte aligned) or NULL to use internal buffer cloudx1_GetBufferOut

8.8.2.5 cloudx1_ProcessAudio_Planar()

```
void cloudx1_ProcessAudio_Planar (
            struct cloudx1_CInstance * instance,
            float const *const * input,
            float *const * output )
```

Process an audio chunk of planar audio

A chunk contains cloudx1_GetProcessChunkFrames frames, set at init by frames_per_chunk in cloudx1_InitProcess3

Input has cloudx1_GetAudioInputCount() * cloudx1_GetChannelCount() channels

Ouput has cloudx1_GetAudioOutputCount() * cloudx1_GetChannelCount() channels

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

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

Parameters

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

8.9 Audio managing (any audio)

Enumerations

- enum cloudx1_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 * cloudx1_GetFormatName (const enum cloudx1_SampleFormat fmt)
- enum cloudx1_SampleFormat cloudx1_GetFormatFromName (const char *name)

- unsigned int cloudx1_GetBytesFromFormat (const enum cloudx1_SampleFormat fmt)
- unsigned int cloudx1_GetMaxPacketFrame (struct cloudx1_CInstance *instance)
- unsigned int cloudx1_AddAudio (struct cloudx1_CInstance *instance, const uint8_t *payload, unsigned int nFrame, enum cloudx1_SampleFormat fmt)
- unsigned int cloudx1_AddPadAudio (struct cloudx1_CInstance *instance)
- unsigned int cloudx1_GetOutputCount (struct cloudx1_CInstance *instance)
- unsigned int cloudx1_GetAudio (struct cloudx1_CInstance *instance, uint8_t *payload, unsigned int max_↩ nFrame, enum cloudx1_SampleFormat fmt)

8.9.1 Detailed Description

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

Remarks

It is more optimized to use Direct processing.

8.9.2 Enumeration Type Documentation

8.9.2.1 cloudx1_SampleFormat

`enum cloudx1_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 896 of file sound4.x1.cloud.h.

8.9.3 Function Documentation

8.9.3.1 cloudx1_AddAudio()

```
unsigned int cloudx1_AddAudio (
            struct cloudx1_CInstance * instance,
            const uint8_t * payload,
            unsigned int nFrame,
            enum cloudx1_SampleFormat fmt )
```

Convert and process the payload.

Parameters

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

Returns

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

8.9.3.2 cloudx1_AddPadAudio()

```
unsigned int cloudx1_AddPadAudio (
            struct cloudx1_CInstance * instance )
```

Add silence to finish the current audio chunk and process it

Parameters

instance	the processing instance

Returns

number of added silence frame

8.9.3.3 cloudx1_GetAudio()

```
unsigned int cloudx1_GetAudio (
            struct cloudx1_CInstance * instance,
```

```
            uint8_t * payload,
            unsigned int max_nFrame,
            enum cloudx1_SampleFormat fmt )
```

Consume the output

Parameters

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

Returns

number of frames put in the buffer

8.9.3.4 cloudx1_GetBytesFromFormat()

```
unsigned int cloudx1_GetBytesFromFormat (
            const enum cloudx1_SampleFormat fmt )
```

Get the format byte size

Parameters

fmt	the format

Returns

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

8.9.3.5 cloudx1_GetFormatFromName()

```
enum cloudx1_SampleFormat cloudx1_GetFormatFromName (
            const char * name )
```

Convert string to format

Parameters

name	the format string

Returns

the matching format, or INVALID_FORMAT if invalid.

8.9.3.6 cloudx1_GetFormatName()

```
const char* cloudx1_GetFormatName (
            const enum cloudx1_SampleFormat fmt )
```

Convert format to string

Parameters

fmt	the format

Returns

the string name of the format

8.9.3.7 cloudx1_GetMaxPacketFrame()

```
unsigned int cloudx1_GetMaxPacketFrame (
            struct cloudx1_CInstance * instance )
```

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

Parameters

instance	The process instance

Returns

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

8.9.3.8 cloudx1_GetOutputCount()

```
unsigned int cloudx1_GetOutputCount (
            struct cloudx1_CInstance * instance )
```

Get the available output size

Parameters

instance	the processing instance

Returns

number of available frames at output

8.10 Audio convert helpers

Functions

- void cloudx1_AudioConvertFrom (const uint8_t ∗payload, float ∗output, size_t nSpl, enum cloudx1_SampleFormat fmt)
- void cloudx1_AudioConvertTo (const float ∗input, uint8_t ∗payload, size_t nSpl, enum cloudx1_SampleFormat fmt)
- void cloudx1_StereoToMono (const float ∗input, float ∗output, size_t nFrame)
- void cloudx1_MonoToStereo (const float ∗input, float ∗output, size_t nFrame)
- void cloudx1_StereoToMono_Planar (const float ∗inputL, const float ∗inputR, float ∗output, size_t nFrame)
- void cloudx1_MonoToStereo_Planar (const float ∗input, float ∗outputL, float ∗outputR, size_t nFrame)
- void cloudx1_AudioMonoFromLiveStereo (const uint8_t ∗payload, float ∗output)
- void cloudx1_AudioMonoToLiveStereo (const float ∗input, uint8_t ∗payload)

8.10.1 Detailed Description

8.10.2 Function Documentation

8.10.2.1 cloudx1_AudioConvertFrom()

```
void cloudx1_AudioConvertFrom (
            const uint8_t * payload,
            float * output,
            size_t nSpl,
            enum cloudx1_SampleFormat fmt )
```

Convert the payload to floating-point array

Warning

Output buffer must be 64-byte aligned

Parameters

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

8.10.2.2 cloudx1_AudioConvertTo()

```
void cloudx1_AudioConvertTo (
            const float * input,
            uint8_t * payload,
            size_t nSpl,
            enum cloudx1_SampleFormat fmt )
```

Convert floating-point array to payload

Warning

Input buffer must be 64-byte aligned

Parameters

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

8.10.2.3 cloudx1_AudioMonoFromLiveStereo()

```
void cloudx1_AudioMonoFromLiveStereo (
            const uint8_t * payload,
            float * output )
```

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

Parameters

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

8.10.2.4 cloudx1_AudioMonoToLiveStereo()

```
void cloudx1_AudioMonoToLiveStereo (
            const float * input,
            uint8_t * payload )
```

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

Parameters

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

8.10.2.5 cloudx1_MonoToStereo()

```
void cloudx1_MonoToStereo (
            const float * input,
            float * output,
            size_t nFrame )
```

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

Warning

Buffers must be 64-byte aligned

Parameters

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

8.10.2.6 cloudx1_MonoToStereo_Planar()

```
void cloudx1_MonoToStereo_Planar (
            const float * input,
            float * outputL,
            float * outputR,
            size_t nFrame )
```

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

Warning

Buffers must be 64-byte aligned

Parameters

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

8.10.2.7 cloudx1_StereoToMono()

```
void cloudx1_StereoToMono (
            const float * input,
            float * output,
            size_t nFrame )
```

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

Warning

Buffers must be 64-byte aligned

Parameters

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

8.10.2.8 cloudx1_StereoToMono_Planar()

```
void cloudx1_StereoToMono_Planar (
            const float * inputL,
            const float * inputR,
            float * output,
            size_t nFrame )
```

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

Warning

Buffers must be 64-byte aligned

Parameters

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

8.11 Client's JSon config interfacing

Classes

- struct cloudx1_CClientInstance

Functions

- struct cloudx1_CClientInstance * cloudx1_NewClient (struct cloudx1_CInstance *instance)
- void cloudx1_DeleteClient (struct cloudx1_CClientInstance *client)
- const char * cloudx1_ProcessJson (struct cloudx1_CClientInstance *client, const char *json_str, int *need↩ _save)
- void cloudx1_FreeJsonAnswer (const char *json_str)
- int cloudx1_SaveState (struct cloudx1_CInstance *instance)

8.11.1 Detailed Description

Allow low-level JSON configuration. see also JSON syntax

8.11.2 Function Documentation

8.11.2.1 cloudx1_DeleteClient()

```
void cloudx1_DeleteClient (
            struct cloudx1_CClientInstance * client )
```

Deletes a client

Parameters

client	the client instance

8.11.2.2 cloudx1_FreeJsonAnswer()

```
void cloudx1_FreeJsonAnswer (
            const char * json_str )
```

Free a JSON answer returned by cloudx1_ProcessJson

Parameters

json_str	the JSON request

8.11.2.3 cloudx1_NewClient()

```
struct cloudx1_CClientInstance* cloudx1_NewClient (
            struct cloudx1_CInstance * instance )
```

Creates a new client

Remarks

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

Parameters

instance	the processing instance

Returns

A new client instance

8.11.2.4 cloudx1_ProcessJson()

```
const char* cloudx1_ProcessJson (
            struct cloudx1_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 cloudx1_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 cloudx1_FreeJsonAnswer)

8.11.2.5 cloudx1_SaveState()

```
int cloudx1_SaveState (
            struct cloudx1_CInstance * instance )
```

Save the current state

Needed when cloudx1_ProcessJson returned with need_save to 1. However it can be postponed as much as wanted.

Remarks

Works only when update thread is running

Parameters

instance	the processing instance

Returns

0 if success, <0 on error

8.12 JSON syntax

JSON protocol to configure the process

8.12.1 Commands

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

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

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

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

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

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

Note

In x1.Cloud version <= 1.3.7, the state resetted also for filtered out objects. So avoid alterning `object` filter between calls, or you will miss some changes.

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

load the preset

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

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

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

delete the preset.
can also be an array of presetnames

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

Create a preset from factory and load it

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

Rename a preset

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

Import a preset

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

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

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

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

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

Undo last(s) `count` preset change

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

Redo last(s) `count` preset change

See also JSON syntax for internal setup

8.12.2 Answers

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

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

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

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

If there is an error, answer has the syntax

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

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

Note

In x1.Cloud version <= 1.2.2, the `OK` and `ERROR` answers had the syntax

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

8.12.3 Examples

8.12.3.1 Get status

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

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

8.12.3.2 Get all changes since last request

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

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

8.12.3.3 Check license

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

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

8.12.3.4 Set Processor name

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

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

8.12.3.5 Get presetlist

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

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

8.12.3.6 Load a preset

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

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

8.12.3.7 Get information on preset on air

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

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

8.12.3.8 Set input gain

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

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

8.12.3.9 Set bypass

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

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

8.12.3.10 Export a preset

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

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

Note

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

8.12.3.11 Import a preset

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

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

8.13 JSON syntax for internal setup

JSON commands for internal setup and information.

This is used internally for the web server.

This API might change in the future.

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

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

8.13.1 Get process and modules information

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

answer:

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

8.13.2 Authentication management

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

Secret stored is

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

8.13.2.1 Difference between setup and admin account

There are 2 modes:

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

From JSON, changing setup_is_admin can change the mode.

8.13.2.2 Access encoding

Access is a combinaison of letters.

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

8.13.2.3 Change setup account

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

8.13.2.3.1 Set the setup/admin account

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

answer:

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

8.13.2.3.2 Set the setup/admin account with clear password

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

answer:

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

8.13.2.3.3 Remove the setup/admin account

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

answer:

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

8.13.2.4 Get user info

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

answer:

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

8.13.2.5 Add or modify a user

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

answer:

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

8.13.2.6 Removes a user

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

answer:

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

8.13.2.7 List users

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

answer:

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

8.13.2.8 Get/set unlogged access

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

answer:

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

8.13.2.9 Get realm

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

answer:

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

8.13.2.10 Get API version

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

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

8.13.2.11 Get setup is admin

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

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

8.13.2.12 Get/set setup account

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

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

8.13.2.13 Get/set custom storage

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

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

8.13.3 Get process' interface description

8.13.3.1 Get XML decription

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

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

Attention

The answer is XML, not JSON !

8.13.3.2 Get Process Image

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

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

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

8.13.3.3 Get Process Ico

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

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

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

8.14 Web server

Macros

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

Functions

- uint64_t cloudx1_Webserver_tcp (unsigned int listenport, unsigned int listenport_secure, const char *socket_ip, unsigned int socket_port)
- uint64_t cloudx1_Webserver_tcp2 (unsigned int listenport, unsigned int listenport_secure, const char *socket_ip, unsigned int socket_port, const struct cloudx1_CParameters *parameters)
- uint64_t cloudx1_Webserver (unsigned int listenport, unsigned int listenport_secure, struct cloudx1_CInstance *instance)
- int cloudx1_Webserver_Stop (uint64_t id, int timeout_ms)
- int cloudx1_Webserver_Status (uint64_t id)
- void cloudx1_Webserver_SetAppHealth (struct cloudx1_CInstance *instance, int httpcode, const char *contenttype, const char *content)
- void cloudx1_Webserver_GetAppHealth (struct cloudx1_CInstance *instance, int *httpcode, char **contenttype, char **content)
- void cloudx1_Webserver_FreeString (char *str)

8.14.1 Detailed Description

Run a web server to configure the processor

See also how to setup authentication with Environment variables and parameters. Their use is documented in cloudx1_InitProcess.

8.14.2 Macro Definition Documentation

8.14.2.1 SOUND4_INVALID_WEBSERVER_ID

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

This represents an invalid webserver id cloudx1_Webserver and cloudx1_Webserver_tcp will never return this value

Definition at line 1572 of file sound4.x1.cloud.h.

8.14.2.2 SOUND4_WEBSERVER_HTTP_OK

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

If this bit is set in the return value from cloudx1_Webserver_Status, then HTTP server is OK

Definition at line 1577 of file sound4.x1.cloud.h.

8.14.2.3 SOUND4_WEBSERVER_HTTPS_OK

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

If this bit is set in the return value from cloudx1_Webserver_Status, then HTTPS server is OK

Definition at line 1581 of file sound4.x1.cloud.h.

8.14.3 Function Documentation

8.14.3.1 cloudx1_Webserver()

```
uint64_t cloudx1_Webserver (
            unsigned int listenport,
            unsigned int listenport_secure,
            struct cloudx1_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 cloudx1_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 cloudx1_Webserver_Stop

8.14.3.2 cloudx1_Webserver_FreeString()

```
void cloudx1_Webserver_FreeString (
            char * str )
```

Free a a string returned by cloudx1_Webserver_GetAppHealth

Parameters

str	the string to free

8.14.3.3 cloudx1_Webserver_GetAppHealth()

```
void cloudx1_Webserver_GetAppHealth (
            struct cloudx1_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 cloudx1_Webserver_FreeString)
content	The http body content pointer (to be freed with cloudx1_Webserver_FreeString)

8.14.3.4 cloudx1_Webserver_SetAppHealth()

```
void cloudx1_Webserver_SetAppHealth (
            struct cloudx1_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

8.14.3.5 cloudx1_Webserver_Status()

```
int cloudx1_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 cloudx1_Webserver_tcp or cloudx1_Webserver

Returns

a combinaison of SOUND4_WEBSERVER_HTTP_OK and SOUND4_WEBSERVER_HTTPS_OK depending on the server status

8.14.3.6 cloudx1_Webserver_Stop()

```
int cloudx1_Webserver_Stop (
            uint64_t id,
            int timeout_ms )
```

Stop a web server started with cloudx1_Webserver_tcp

Warning

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

Parameters

id	The web server Id, returned by cloudx1_Webserver_tcp or cloudx1_Webserver
timeout_ms	The timeout to wait gracefull shutdown

Returns

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

8.14.3.7 cloudx1_Webserver_tcp()

```
uint64_t cloudx1_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 cloudx1_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 cloudx1_Webserver_Stop

8.14.3.8 cloudx1_Webserver_tcp2()

```
uint64_t cloudx1_Webserver_tcp2 (
            unsigned int listenport,
            unsigned int listenport_secure,
            const char * socket_ip,
            unsigned int socket_port,
            const struct cloudx1_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 cloudx1_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 cloudx1_NewParameters

Returns

id of webserver (starting at 0), to use with cloudx1_Webserver_Stop

8.15 Sample using C++ interface wrapper

See full example cpp/sound4.x1.cloud-pipe-cxx.cpp

8.15.1 Setup and sanity checks

```
sound4::cloudx1::SetLogSeverity(verbose5);
sound4::cloudx1::SetLoggerCallback(MyLogger);
sound4::cloudx1::CInstance instance;
instance.SetParam("ADMIN_USER","admin");
instance.SetParam("ADMIN_SECRET","admin");
```

8.15.2 Sample to create the processing instance

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

8.15.3 Communicating with the instance

See also

JSON syntax

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

8.15.4 Starting a web-server

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

8.15.5 Sample processing loop

```
while (!stopthread)
{
    // Read a chunk from stdin
    if (std::fread(instance.GetBufferIn().data(), sizeof(float), sound4::cloudx1::InputSampleSize,
   stdin) != sound4::cloudx1::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::cloudx1::OutputSampleSize, stdout)
   != sound4::cloudx1::OutputSampleSize) {
        std::cerr « "Failed to write audio output" « std::endl;
        stopthread=true;
    }
}
```

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

See full example cpp/sound4.x1.cloud-pipe-dyn.cpp

8.16.1 Dynamic loading

```
sound4::cloudx1::dyn::CDynLib dynlib;
if (!dynlib.Load()) {
    std::cerr « "Failed to load processing library" « std::endl;
    return 1;
}
```

8.16.2 Setup and sanity checks

```
sound4::cloudx1::dyn::SetLogSeverity(dynlib, verbose5);
sound4::cloudx1::dyn::SetLoggerCallback(dynlib, MyLogger);
sound4::cloudx1::dyn::CInstance instance(dynlib);
instance.SetParam("ADMIN_USER","admin");
instance.SetParam("ADMIN_SECRET","admin");
```

8.16.3 Sample to create the processing instance

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

8.16.4 Communicating with the instance

See also

JSON syntax

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

8.16.5 Starting a web-server

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

8.16.6 Sample processing loop

```
while (!stopthread)
{
    // Read a chunk from stdin
    if (std::fread(instance.GetBufferIn().data(), sizeof(float), sound4::cloudx1::dyn::InputSampleSize,
   stdin) != sound4::cloudx1::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::cloudx1::dyn::OutputSampleSize,
   stdout) != sound4::cloudx1::dyn::OutputSampleSize) {
        std::cerr « "Failed to write audio output" « std::endl;
        stopthread=true;
    }
}
```

8.16.7 Custom preset manager

```
class MyPresetManager : public sound4::cloudx1::dyn::CPresetLoader {
public:
    MyPresetManager() = default;
    virtual ~MyPresetManager() = default;
    bool SetupPath(const std::filesystem::path& a_preset_path)
        {
            m_preset_path=a_preset_path;
            if ( !std::filesystem::exists(m_preset_path ) ) {
                return false;
            }
            std::error_code ec;
            if ( !std::filesystem::is_directory(m_preset_path, ec) ) {
                return false;
            }
            return true;
        }
    virtual bool IsReadOnly() override { return false; };
    virtual bool Exists(const std::filesystem::path &name) override
        {
            std::error_code ec;
            return std::filesystem::exists(GetRealPath(name), ec);
        }
    virtual bool Remove(const std::filesystem::path &name) override
        {
            std::error_code ec;
            return std::filesystem::remove(GetRealPath(name), ec);
        }
    virtual bool Rename(const std::filesystem::path &from, const std::filesystem::path &to) override
        {
            std::error_code ec;
            std::filesystem::rename(GetRealPath(from), GetRealPath(to), ec);
            return (!ec);
        }
    virtual std::vector<std::filesystem::path> GetAll() override
        {
            std::vector<std::filesystem::path> list;
            for (const auto & entry : std::filesystem::directory_iterator(m_preset_path.c_str(),
       std::filesystem::directory_options::skip_permission_denied)) {
                if (entry.is_regular_file()) {
                    list.push_back(entry.path().lexically_relative(m_preset_path));
                }
            }
            return list;
        }
    virtual std::string Read(const std::filesystem::path &filename) override
        {
            std::ifstream f(GetRealPath(filename));
            if ( f.fail() ) {
                return std::string();
            }
```

```
            std::ostringstream ss;
            ss « f.rdbuf();
            return ss.str();
        }
    virtual bool Write(const std::filesystem::path &filename, const std::string &content) override
        {
            std::ofstream f(GetRealPath(filename));
            if ( f.fail() ) {
                return false;
            }
            f « content;
            if ( f.fail() ) {
                return false;
            }
            return true;
        }
private:
    std::filesystem::path m_preset_path;
    std::filesystem::path GetRealPath(const std::filesystem::path &name)
        {
            return m_preset_path / name;
        }
};
```

8.16.8 Custom preset manager loading

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

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

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

It uses sound4cl.hpp header.

See full example cpp/sound4cl-pipe.cpp

8.17.1 Loading the library

```
    std::string libname =  "libsound4.x1.cloud.so";
    sound4::CProcessor processor;
    if (!processor.Load(libname)) {
        std::cerr « "Failed to load processing library" « std::endl;
        return 1;
    }
```

8.17.2 Setup and configure

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

8.17.3 Create the processing instance

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

8.17.4 Processing loop

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

8.17.5 Starting a web-server

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

8.17.6 Communicating with the instance

See also

JSON syntax

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

8.17.7 Custom preset manager example

```
class MyPresetManager : public sound4::CPresetLoader {
public:
    MyPresetManager() = default;
    virtual ~MyPresetManager() = default;
    bool SetupPath(const std::filesystem::path& a_preset_path)
        {
            m_preset_path=a_preset_path;
            if ( !std::filesystem::exists(m_preset_path ) ) {
                return false;
            }
            std::error_code ec;
            if ( !std::filesystem::is_directory(m_preset_path, ec) ) {
                return false;
            }
            return true;
        }
    virtual bool IsReadOnly() override { return false; };
    virtual bool Exists(const std::filesystem::path &name) override
        {
            std::error_code ec;
            return std::filesystem::exists(GetRealPath(name), ec);
        }
    virtual bool Remove(const std::filesystem::path &name) override
        {
            std::error_code ec;
            return std::filesystem::remove(GetRealPath(name), ec);
        }
    virtual bool Rename(const std::filesystem::path &from, const std::filesystem::path &to) override
        {
            std::error_code ec;
            std::filesystem::rename(GetRealPath(from), GetRealPath(to), ec);
            return (!ec);
        }
    virtual std::vector<std::filesystem::path> GetAll() override
```

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

8.17.8 Custom preset manager loading

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

Chapter 9

Namespace Documentation

9.1 dyn Namespace Reference

9.1.1 Detailed Description

Dynamic loading namespace

9.2 sound4 Namespace Reference

Namespaces

- cloudx1
- helper

Classes

- class CProcessor

  Dynamic library interface.
- class CBus

  Shared Bus.
- class CPresetLoader

  Custom preset handler helper.
- class CInstance

  Instance handling class.

Typedefs

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

Variables

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

9.2.1 Detailed Description

SOUND4 namespace

9.2.2 Typedef Documentation

9.2.2.1 log_cb_t

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

Log callback type

Definition at line 252 of file sound4cl.hpp.

9.2.2.2 LogSeverity

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

Log Severity levels

Definition at line 237 of file sound4cl.hpp.

9.2.2.3 PresetChange_Kind

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

Definition at line 743 of file sound4cl.hpp.

9.2.3 Variable Documentation

9.2.3.1 error

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

Definition at line 240 of file sound4cl.hpp.

9.2.3.2 fatal

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

Definition at line 239 of file sound4cl.hpp.

9.2.3.3 info

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

Definition at line 242 of file sound4cl.hpp.

9.2.3.4 none

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

Definition at line 238 of file sound4cl.hpp.

9.2.3.5 verbose

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

Definition at line 243 of file sound4cl.hpp.

9.2.3.6 verbose2

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

Definition at line 244 of file sound4cl.hpp.

9.2.3.7 verbose3

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

Definition at line 245 of file sound4cl.hpp.

9.2.3.8 verbose4

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

Definition at line 246 of file sound4cl.hpp.

9.2.3.9 verbose5

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

Definition at line 247 of file sound4cl.hpp.

9.2.3.10 warning

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

Definition at line 241 of file sound4cl.hpp.

9.3 sound4::cloudx1 Namespace Reference

Namespaces

- dyn
- helper

Classes

- class CPresetLoader
- class CInstance

Typedefs

- using SampleFormat = cloudx1_SampleFormat
- using LogSeverity = cloudx1_LogSeverity
- using log_cb_t = std::function< void(LogSeverity, const std::string &)>
- using PresetChange_Kind = cloudx1_PresetChange_Kind

Variables

- const size_t ChunkFrames =CLOUDX1_AUDIOFRAME_COUNT
- const size_t InputChannels =CLOUDX1_CHANNEL_COUNT∗CLOUDX1_AUDIO_INPUT_COUNT
- const size_t OutputChannels =CLOUDX1_CHANNEL_COUNT∗CLOUDX1_AUDIO_OUTPUT_COUNT
- const size_t InputSampleSize =ChunkFrames∗InputChannels
- const size_t OutputSampleSize =ChunkFrames∗OutputChannels
- const size_t SampleRate =CLOUDX1_SAMPLE_RATE

9.3.1 Detailed Description

cloudx1 namespace

9.3.2 Typedef Documentation

9.3.2.1 log_cb_t

```
using sound4::cloudx1::log_cb_t = typedef std::function<void(LogSeverity,const std::string&)>
```

Log callback type

Definition at line 320 of file sound4.x1.cloud.hpp.

9.3.2.2 LogSeverity

```
using sound4::cloudx1::LogSeverity = typedef cloudx1_LogSeverity
```

Log Severity levels

Definition at line 316 of file sound4.x1.cloud.hpp.

9.3.2.3 PresetChange_Kind

```
using sound4::cloudx1::PresetChange_Kind = typedef cloudx1_PresetChange_Kind
```

Definition at line 380 of file sound4.x1.cloud.hpp.

9.3.2.4 SampleFormat

```
using sound4::cloudx1::SampleFormat = typedef cloudx1_SampleFormat
```

Definition at line 53 of file sound4.x1.cloud.hpp.

9.3.3 Variable Documentation

9.3.3.1 ChunkFrames

```
const size_t sound4::cloudx1::ChunkFrames =CLOUDX1_AUDIOFRAME_COUNT
```

Number of frames in a chunk

Definition at line 57 of file sound4.x1.cloud.hpp.

9.3.3.2 InputChannels

```
const size_t sound4::cloudx1::InputChannels =CLOUDX1_CHANNEL_COUNT*CLOUDX1_AUDIO_INPUT_COUNT
```

Number of float values in an input frame

Definition at line 61 of file sound4.x1.cloud.hpp.

9.3.3.3 InputSampleSize

```
const size_t sound4::cloudx1::InputSampleSize =ChunkFrames*InputChannels
```

Number of float values for an input chunk

Definition at line 69 of file sound4.x1.cloud.hpp.

9.3.3.4 OutputChannels

```
const size_t sound4::cloudx1::OutputChannels =CLOUDX1_CHANNEL_COUNT*CLOUDX1_AUDIO_OUTPUT_COUNT
```

Number of float values in an output frame

Definition at line 65 of file sound4.x1.cloud.hpp.

9.3.3.5 OutputSampleSize

```
const size_t sound4::cloudx1::OutputSampleSize =ChunkFrames*OutputChannels
```

Number of float values for an output chunk

Definition at line 73 of file sound4.x1.cloud.hpp.

9.3.3.6 SampleRate

```
const size_t sound4::cloudx1::SampleRate =CLOUDX1_SAMPLE_RATE
```

Sample rate of library

Definition at line 77 of file sound4.x1.cloud.hpp.

9.4 sound4::cloudx1::dyn Namespace Reference

Namespaces

- helper

Classes

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

Typedefs

- using SampleFormat = cloudx1_SampleFormat
- using LogSeverity = cloudx1_LogSeverity
- using log_cb_t = std::function< void(LogSeverity, const std::string &)>
- using PresetChange_Kind = cloudx1_PresetChange_Kind

Variables

- const size_t ChunkFrames =CLOUDX1_AUDIOFRAME_COUNT
- const size_t InputChannels =CLOUDX1_CHANNEL_COUNT∗CLOUDX1_AUDIO_INPUT_COUNT
- const size_t OutputChannels =CLOUDX1_CHANNEL_COUNT∗CLOUDX1_AUDIO_OUTPUT_COUNT
- const size_t InputSampleSize =ChunkFrames∗InputChannels
- const size_t OutputSampleSize =ChunkFrames∗OutputChannels
- const size_t SampleRate =CLOUDX1_SAMPLE_RATE

9.4.1 Typedef Documentation

9.4.1.1 log_cb_t

```
using sound4::cloudx1::dyn::log_cb_t = typedef std::function<void(LogSeverity,const std↩
::string&)>
```

Log callback type

Definition at line 731 of file sound4.x1.cloud_dyn.hpp.

9.4.1.2 LogSeverity

```
using sound4::cloudx1::dyn::LogSeverity = typedef cloudx1_LogSeverity
```

Log Severity levels

Definition at line 727 of file sound4.x1.cloud_dyn.hpp.

9.4.1.3 PresetChange_Kind

```
using sound4::cloudx1::dyn::PresetChange_Kind = typedef cloudx1_PresetChange_Kind
```

Definition at line 799 of file sound4.x1.cloud_dyn.hpp.

9.4.1.4 SampleFormat

```
using sound4::cloudx1::dyn::SampleFormat = typedef cloudx1_SampleFormat
```

Definition at line 460 of file sound4.x1.cloud_dyn.hpp.

9.4.2 Variable Documentation

9.4.2.1 ChunkFrames

```
const size_t sound4::cloudx1::dyn::ChunkFrames =CLOUDX1_AUDIOFRAME_COUNT
```

Number of frames in a chunk

Definition at line 464 of file sound4.x1.cloud_dyn.hpp.

9.4.2.2 InputChannels

```
const size_t sound4::cloudx1::dyn::InputChannels =CLOUDX1_CHANNEL_COUNT*CLOUDX1_AUDIO_INPUT_COUNT
```

Number of float values in an input frame

Definition at line 468 of file sound4.x1.cloud_dyn.hpp.

9.4.2.3 InputSampleSize

```
const size_t sound4::cloudx1::dyn::InputSampleSize =ChunkFrames*InputChannels
```

Number of float values for an input chunk

Definition at line 476 of file sound4.x1.cloud_dyn.hpp.

9.4.2.4 OutputChannels

```
const size_t sound4::cloudx1::dyn::OutputChannels =CLOUDX1_CHANNEL_COUNT*CLOUDX1_AUDIO_OUTPUT_COUNT
```

Number of float values in an output frame

Definition at line 472 of file sound4.x1.cloud_dyn.hpp.

9.4.2.5 OutputSampleSize

```
const size_t sound4::cloudx1::dyn::OutputSampleSize =ChunkFrames*OutputChannels
```

Number of float values for an output chunk

Definition at line 480 of file sound4.x1.cloud_dyn.hpp.

9.4.2.6 SampleRate

```
const size_t sound4::cloudx1::dyn::SampleRate =CLOUDX1_SAMPLE_RATE
```

Sample rate of library

Definition at line 484 of file sound4.x1.cloud_dyn.hpp.

9.5 sound4::cloudx1::dyn::helper Namespace Reference

Classes

- class DynFuncHolder

  Helper to wrap a function pointer.
- class CDynLoader

  Helper to load dynamic library.
- struct SampleFormat

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

Typedefs

- using DynLib_t = HMODULE
- using DynFunc_t = FARPROC

9.5.1 Detailed Description

helper functions for unicode strings in Windows

9.5.2 Typedef Documentation

9.5.2.1 DynFunc_t

```
using sound4::cloudx1::dyn::helper::DynFunc_t = typedef FARPROC
```

Definition at line 77 of file sound4.x1.cloud_dyn.hpp.

9.5.2.2 DynLib_t

```
using sound4::cloudx1::dyn::helper::DynLib_t = typedef HMODULE
```

Definition at line 76 of file sound4.x1.cloud_dyn.hpp.

9.6 sound4::cloudx1::helper Namespace Reference

Classes

- struct SampleFormat

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

9.6.1 Detailed Description

helper functions for unicode strings in Windows

9.7 sound4::helper Namespace Reference

Classes

- struct SampleFormat

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

  Helper to wrap a function pointer.
- class CDynLoader

  Helper to load dynamic library.

Typedefs

- using DynLib_t = HMODULE
- using DynFunc_t = FARPROC

9.7.1 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 106 of file sound4cl.hpp.

9.7.2.2 DynLib_t

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

Definition at line 105 of file sound4cl.hpp.

9.8 std Namespace Reference

Chapter 10

Class Documentation

10.1 sound4::CBus Class Reference

Shared Bus.

`#include <sound4cl.hpp>`

Public Member Functions

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

10.1.1 Detailed Description

Some processor can share a bus for realtime data exchange

Definition at line 703 of file sound4cl.hpp.

10.1.2 Constructor & Destructor Documentation

10.1.2.1 CBus()

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

Definition at line 705 of file sound4cl.hpp.

10.1.2.2 ~CBus()

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

Definition at line 712 of file sound4cl.hpp.

10.1.3 Member Function Documentation

10.1.3.1 Get()

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

Definition at line 719 of file sound4cl.hpp.

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

- sound4cl.hpp

10.2 sound4::CInstance::CClient Class Reference

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

Public Member Functions

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

  Process a JSON request and returns the answer.

10.2.1 Detailed Description

Low-level JSON client.

See also

JSON syntax

Definition at line 1215 of file sound4cl.hpp.

10.2.2 Constructor & Destructor Documentation

10.2.2.1 CClient() [1/2]

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

Definition at line 1217 of file sound4cl.hpp.

10.2.2.2 CClient() [2/2]

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

10.2.2.3 ~CClient()

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

Definition at line 1227 of file sound4cl.hpp.

10.2.3 Member Function Documentation

10.2.3.1 operator=()

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

10.2.3.2 ProcessJson()

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

Parameters

request	the JSON request.

See also

JSON syntax

Parameters

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

Returns

the JSON answer

Definition at line 1241 of file sound4cl.hpp.

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

- sound4cl.hpp

10.3 sound4::cloudx1::CInstance::CClient Class Reference

`#include <sound4.x1.cloud.hpp>`

Public Member Functions

- CClient (cloudx1_CInstance *instance)
- CClient (const CInstance &)=delete
- CClient & operator= (const CInstance &)=delete
- ~CClient ()
- std::string ProcessJson (const std::string &request, bool *NeedSave=nullptr)

  Process a JSON request and returns the answer.

10.3.1 Detailed Description

Low-level JSON client.

See also

JSON syntax

Definition at line 827 of file sound4.x1.cloud.hpp.

10.3.2 Constructor & Destructor Documentation

10.3.2.1 CClient() [1/2]

```
sound4::cloudx1::CInstance::CClient::CClient (
            cloudx1_CInstance * instance )  [inline]
```

Definition at line 829 of file sound4.x1.cloud.hpp.

10.3.2.2 CClient() [2/2]

```
sound4::cloudx1::CInstance::CClient::CClient (
            const CInstance &  )  [delete]
```

10.3.2.3 ~CClient()

```
sound4::cloudx1::CInstance::CClient::~CClient ( )  [inline]
```

Definition at line 838 of file sound4.x1.cloud.hpp.

10.3.3 Member Function Documentation

10.3.3.1 operator=()

```
CClient& sound4::cloudx1::CInstance::CClient::operator= (
            const CInstance &  )  [delete]
```

10.3.3.2 ProcessJson()

```
std::string sound4::cloudx1::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.x1.cloud.hpp.

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

- sound4.x1.cloud.hpp

10.4 sound4::cloudx1::dyn::CInstance::CClient Class Reference

`#include <sound4.x1.cloud_dyn.hpp>`

Public Member Functions

- CClient (cloudx1_CInstance *instance, CDynLib &dynlib)
- CClient (const CInstance &)=delete
- CClient & operator= (const CInstance &)=delete
- ~CClient ()
- std::string ProcessJson (const std::string &request, bool *NeedSave=nullptr)

  Process a JSON request and returns the answer.

10.4.1 Detailed Description

Low-level JSON client.

See also

JSON syntax

Definition at line 1270 of file sound4.x1.cloud_dyn.hpp.

10.4.2 Constructor & Destructor Documentation

10.4.2.1 CClient() [1/2]

```
sound4::cloudx1::dyn::CInstance::CClient::CClient (
            cloudx1_CInstance * instance,
            CDynLib & dynlib )  [inline]
```

Definition at line 1272 of file sound4.x1.cloud_dyn.hpp.

10.4.2.2 CClient() [2/2]

```
sound4::cloudx1::dyn::CInstance::CClient::CClient (
            const CInstance &  )  [delete]
```

10.4.2.3 ~CClient()

```
sound4::cloudx1::dyn::CInstance::CClient::~CClient ( )  [inline]
```

Definition at line 1282 of file sound4.x1.cloud_dyn.hpp.

10.4.3 Member Function Documentation

10.4.3.1 operator=()

```
CClient& sound4::cloudx1::dyn::CInstance::CClient::operator= (
            const CInstance &  )  [delete]
```

10.4.3.2 ProcessJson()

```
std::string sound4::cloudx1::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 1296 of file sound4.x1.cloud_dyn.hpp.

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

- sound4.x1.cloud_dyn.hpp

10.5 sound4::cloudx1::dyn::CDynLib Class Reference

Dynamic library interface.

`#include <sound4.x1.cloud_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(cloudx1_GetVersion) > GetVersion
- helper::DynFuncHolder< decltype(cloudx1_GetChunkSizeInFrames) > GetChunkSizeInFrames
- helper::DynFuncHolder< decltype(cloudx1_GetChannelCount) > GetChannelCount
- helper::DynFuncHolder< decltype(cloudx1_GetAudioInputCount) > GetAudioInputCount
- helper::DynFuncHolder< decltype(cloudx1_GetAudioOutputCount) > GetAudioOutputCount
- helper::DynFuncHolder< decltype(cloudx1_GetSampleRate) > GetSampleRate
- helper::DynFuncHolder< decltype(cloudx1_SetLoggerCallback) > SetLoggerCallback
- helper::DynFuncHolder< decltype(cloudx1_SetLogSeverity) > SetLogSeverity
- helper::DynFuncHolder< decltype(cloudx1_NewParameters) > NewParameters
- helper::DynFuncHolder< decltype(cloudx1_FreeParameters) > FreeParameters
- helper::DynFuncHolder< decltype(cloudx1_SetParameter) > SetParameter
- helper::DynFuncHolder< decltype(cloudx1_GetParameter) > GetParameter
- helper::DynFuncHolder< decltype(cloudx1_FreeParameterValue) > FreeParameterValue
- helper::DynFuncHolder< decltype(cloudx1_InitProcess) > InitProcess
- helper::DynFuncHolder< decltype(cloudx1_InitProcess2) > InitProcess2

- helper::DynFuncHolder< decltype(cloudx1_TerminateProcess) > TerminateProcess
- helper::DynFuncHolder< decltype(cloudx1_ExitProcess) > ExitProcess
- helper::DynFuncHolder< decltype(cloudx1_StartUpdateThread) > StartUpdateThread
- helper::DynFuncHolder< decltype(cloudx1_StopUpdateThread) > StopUpdateThread
- helper::DynFuncHolder< decltype(cloudx1_WaitUpdateThreadReady) > WaitUpdateThreadReady
- helper::DynFuncHolder< decltype(cloudx1_ProcessAudio) > ProcessAudio
- helper::DynFuncHolder< decltype(cloudx1_ProcessAudio_Planar) > ProcessAudio_Planar
- helper::DynFuncHolder< decltype(cloudx1_GetBufferIn) > GetBufferIn
- helper::DynFuncHolder< decltype(cloudx1_GetBufferOut) > GetBufferOut
- helper::DynFuncHolder< decltype(cloudx1_GetEstimatedDelay) > GetEstimatedDelay
- helper::DynFuncHolder< decltype(cloudx1_GetFormatName) > GetFormatName
- helper::DynFuncHolder< decltype(cloudx1_GetFormatFromName) > GetFormatFromName
- helper::DynFuncHolder< decltype(cloudx1_GetBytesFromFormat) > GetBytesFromFormat
- helper::DynFuncHolder< decltype(cloudx1_GetMaxPacketFrame) > GetMaxPacketFrame
- helper::DynFuncHolder< decltype(cloudx1_AddAudio) > AddAudio
- helper::DynFuncHolder< decltype(cloudx1_AddPadAudio) > AddPadAudio
- helper::DynFuncHolder< decltype(cloudx1_GetOutputCount) > GetOutputCount
- helper::DynFuncHolder< decltype(cloudx1_GetAudio) > GetAudio
- helper::DynFuncHolder< decltype(cloudx1_AudioConvertFrom) > AudioConvertFrom
- helper::DynFuncHolder< decltype(cloudx1_AudioConvertTo) > AudioConvertTo
- helper::DynFuncHolder< decltype(cloudx1_StereoToMono) > StereoToMono
- helper::DynFuncHolder< decltype(cloudx1_MonoToStereo) > MonoToStereo
- helper::DynFuncHolder< decltype(cloudx1_AudioMonoFromLiveStereo) > AudioMonoFromLiveStereo
- helper::DynFuncHolder< decltype(cloudx1_AudioMonoToLiveStereo) > AudioMonoToLiveStereo
- helper::DynFuncHolder< decltype(cloudx1_NewClient) > NewClient
- helper::DynFuncHolder< decltype(cloudx1_DeleteClient) > DeleteClient
- helper::DynFuncHolder< decltype(cloudx1_ProcessJson) > ProcessJson
- helper::DynFuncHolder< decltype(cloudx1_FreeJsonAnswer) > FreeJsonAnswer
- helper::DynFuncHolder< decltype(cloudx1_SaveState) > SaveState
- helper::DynFuncHolder< decltype(cloudx1_Webserver_tcp) > Webserver_tcp
- helper::DynFuncHolder< decltype(cloudx1_Webserver_tcp2) > Webserver_tcp2
- helper::DynFuncHolder< decltype(cloudx1_Webserver) > Webserver
- helper::DynFuncHolder< decltype(cloudx1_Webserver_Stop) > Webserver_Stop
- helper::DynFuncHolder< decltype(cloudx1_Webserver_Status) > Webserver_Status
- helper::DynFuncHolder< decltype(cloudx1_StereoToMono_Planar) > StereoToMono_Planar
- helper::DynFuncHolder< decltype(cloudx1_MonoToStereo_Planar) > MonoToStereo_Planar
- helper::DynFuncHolder< decltype(cloudx1_SetMetadata) > SetMetadata
- helper::DynFuncHolder< decltype(cloudx1_GetMetadataInfos) > GetMetadataInfos
- helper::DynFuncHolder< decltype(cloudx1_FreeMetadataInfos) > FreeMetadataInfos
- helper::DynFuncHolder< decltype(cloudx1_SetPresetManager) > SetPresetManager
- helper::DynFuncHolder< decltype(cloudx1_PresetManager_InformChange) > PresetManager_InformChange
- helper::DynFuncHolder< decltype(cloudx1_GetPossibleChunkSizeInFrames) > GetPossibleChunkSizeInFrames
- helper::DynFuncHolder< decltype(cloudx1_GetProcessChunkFrames) > GetProcessChunkFrames
- helper::DynFuncHolder< decltype(cloudx1_InitProcess3) > InitProcess3
- helper::DynFuncHolder< decltype(cloudx1_SetMetadataMulti) > SetMetadataMulti
- helper::DynFuncHolder< decltype(cloudx1_Webserver_SetAppHealth) > Webserver_SetAppHealth
- helper::DynFuncHolder< decltype(cloudx1_Webserver_GetAppHealth) > Webserver_GetAppHealth
- helper::DynFuncHolder< decltype(cloudx1_Webserver_FreeString) > Webserver_FreeString

10.5.1 Detailed Description

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

Definition at line 210 of file sound4.x1.cloud_dyn.hpp.

10.5.2 Constructor & Destructor Documentation

10.5.2.1 CDynLib()

```
sound4::cloudx1::dyn::CDynLib::CDynLib ( )  [default]
```

10.5.3 Member Function Documentation

10.5.3.1 IsOk()

```
bool sound4::cloudx1::dyn::CDynLib::IsOk ( ) const  [inline]
```

Returns

true if OK.

false if not OK.

Definition at line 450 of file sound4.x1.cloud_dyn.hpp.

10.5.3.2 Load()

```
bool sound4::cloudx1::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 312 of file sound4.x1.cloud_dyn.hpp.

10.5.4 Member Data Documentation

10.5.4.1 AddAudio

```
helper::DynFuncHolder< decltype(cloudx1_AddAudio ) > sound4::cloudx1::dyn::CDynLib::AddAudio
```

Definition at line 247 of file sound4.x1.cloud_dyn.hpp.

10.5.4.2 AddPadAudio

```
helper::DynFuncHolder< decltype(cloudx1_AddPadAudio ) > sound4::cloudx1::dyn::CDynLib::Add↩
PadAudio
```

Definition at line 248 of file sound4.x1.cloud_dyn.hpp.

10.5.4.3 AudioConvertFrom

```
helper::DynFuncHolder< decltype(cloudx1_AudioConvertFrom ) > sound4::cloudx1::dyn::CDynLib::↩
AudioConvertFrom
```

Definition at line 251 of file sound4.x1.cloud_dyn.hpp.

10.5.4.4 AudioConvertTo

```
helper::DynFuncHolder< decltype(cloudx1_AudioConvertTo ) > sound4::cloudx1::dyn::CDynLib::↩
AudioConvertTo
```

Definition at line 252 of file sound4.x1.cloud_dyn.hpp.

10.5.4.5 AudioMonoFromLiveStereo

```
helper::DynFuncHolder< decltype(cloudx1_AudioMonoFromLiveStereo) > sound4::cloudx1::dyn::↩
CDynLib::AudioMonoFromLiveStereo
```

Definition at line 255 of file sound4.x1.cloud_dyn.hpp.

10.5.4.6 AudioMonoToLiveStereo

```
helper::DynFuncHolder< decltype(cloudx1_AudioMonoToLiveStereo ) > sound4::cloudx1::dyn::CDyn↩
Lib::AudioMonoToLiveStereo
```

Definition at line 256 of file sound4.x1.cloud_dyn.hpp.

10.5.4.7 DeleteClient

```
helper::DynFuncHolder< decltype(cloudx1_DeleteClient ) > sound4::cloudx1::dyn::CDynLib::↩
DeleteClient
```

Definition at line 258 of file sound4.x1.cloud_dyn.hpp.

10.5.4.8 ExitProcess

```
helper::DynFuncHolder< decltype(cloudx1_ExitProcess ) > sound4::cloudx1::dyn::CDynLib::Exit↩
Process
```

Definition at line 234 of file sound4.x1.cloud_dyn.hpp.

10.5.4.9 FreeJsonAnswer

```
helper::DynFuncHolder< decltype(cloudx1_FreeJsonAnswer ) > sound4::cloudx1::dyn::CDynLib::↩
FreeJsonAnswer
```

Definition at line 260 of file sound4.x1.cloud_dyn.hpp.

10.5.4.10 FreeMetadataInfos

```
helper::DynFuncHolder< decltype(cloudx1_FreeMetadataInfos ) > sound4::cloudx1::dyn::CDynLib↩
::FreeMetadataInfos
```

Definition at line 277 of file sound4.x1.cloud_dyn.hpp.

10.5.4.11 FreeParameters

```
helper::DynFuncHolder< decltype(cloudx1_FreeParameters ) > sound4::cloudx1::dyn::CDynLib::↩
FreeParameters
```

Definition at line 227 of file sound4.x1.cloud_dyn.hpp.

10.5.4.12 FreeParameterValue

```
helper::DynFuncHolder< decltype(cloudx1_FreeParameterValue ) > sound4::cloudx1::dyn::CDynLib↩
::FreeParameterValue
```

Definition at line 230 of file sound4.x1.cloud_dyn.hpp.

10.5.4.13 GetAudio

```
helper::DynFuncHolder< decltype(cloudx1_GetAudio ) > sound4::cloudx1::dyn::CDynLib::GetAudio
```

Definition at line 250 of file sound4.x1.cloud_dyn.hpp.

10.5.4.14 GetAudioInputCount

```
helper::DynFuncHolder< decltype(cloudx1_GetAudioInputCount ) > sound4::cloudx1::dyn::CDynLib↩
::GetAudioInputCount
```

Definition at line 221 of file sound4.x1.cloud_dyn.hpp.

10.5.4.15 GetAudioOutputCount

```
helper::DynFuncHolder< decltype(cloudx1_GetAudioOutputCount ) > sound4::cloudx1::dyn::CDyn↩
Lib::GetAudioOutputCount
```

Definition at line 222 of file sound4.x1.cloud_dyn.hpp.

10.5.4.16 GetBufferIn

```
helper::DynFuncHolder< decltype(cloudx1_GetBufferIn ) > sound4::cloudx1::dyn::CDynLib::Get↩
BufferIn
```

Definition at line 240 of file sound4.x1.cloud_dyn.hpp.

10.5.4.17 GetBufferOut

```
helper::DynFuncHolder< decltype(cloudx1_GetBufferOut ) > sound4::cloudx1::dyn::CDynLib::Get↩
BufferOut
```

Definition at line 241 of file sound4.x1.cloud_dyn.hpp.

10.5.4.18 GetBytesFromFormat

```
helper::DynFuncHolder< decltype(cloudx1_GetBytesFromFormat ) > sound4::cloudx1::dyn::CDynLib↩
::GetBytesFromFormat
```

Definition at line 245 of file sound4.x1.cloud_dyn.hpp.

10.5.4.19 GetChannelCount

```
helper::DynFuncHolder< decltype(cloudx1_GetChannelCount ) > sound4::cloudx1::dyn::CDynLib::↩
GetChannelCount
```

Definition at line 220 of file sound4.x1.cloud_dyn.hpp.

10.5.4.20 GetChunkSizeInFrames

```
helper::DynFuncHolder< decltype(cloudx1_GetChunkSizeInFrames ) > sound4::cloudx1::dyn::CDyn↩
Lib::GetChunkSizeInFrames
```

Definition at line 219 of file sound4.x1.cloud_dyn.hpp.

10.5.4.21 GetEstimatedDelay

```
helper::DynFuncHolder< decltype(cloudx1_GetEstimatedDelay ) > sound4::cloudx1::dyn::CDynLib↩
::GetEstimatedDelay
```

Definition at line 242 of file sound4.x1.cloud_dyn.hpp.

10.5.4.22 GetFormatFromName

```
helper::DynFuncHolder< decltype(cloudx1_GetFormatFromName ) > sound4::cloudx1::dyn::CDynLib↩
::GetFormatFromName
```

Definition at line 244 of file sound4.x1.cloud_dyn.hpp.

10.5.4.23 GetFormatName

```
helper::DynFuncHolder< decltype(cloudx1_GetFormatName ) > sound4::cloudx1::dyn::CDynLib::Get↩
FormatName
```

Definition at line 243 of file sound4.x1.cloud_dyn.hpp.

10.5.4.24 GetMaxPacketFrame

```
helper::DynFuncHolder< decltype(cloudx1_GetMaxPacketFrame ) > sound4::cloudx1::dyn::CDynLib↩
::GetMaxPacketFrame
```

Definition at line 246 of file sound4.x1.cloud_dyn.hpp.

10.5.4.25 GetMetadataInfos

helper::DynFuncHolder< decltype(cloudx1_GetMetadataInfos) > sound4::cloudx1::dyn::CDynLib::↩
GetMetadataInfos

Definition at line 276 of file sound4.x1.cloud_dyn.hpp.

10.5.4.26 GetOutputCount

helper::DynFuncHolder< decltype(cloudx1_GetOutputCount) > sound4::cloudx1::dyn::CDynLib::↩
GetOutputCount

Definition at line 249 of file sound4.x1.cloud_dyn.hpp.

10.5.4.27 GetParameter

helper::DynFuncHolder< decltype(cloudx1_GetParameter) > sound4::cloudx1::dyn::CDynLib::Get↩
Parameter

Definition at line 229 of file sound4.x1.cloud_dyn.hpp.

10.5.4.28 GetPossibleChunkSizeInFrames

helper::DynFuncHolder< decltype(cloudx1_GetPossibleChunkSizeInFrames) > sound4::cloudx1↩
::dyn::CDynLib::GetPossibleChunkSizeInFrames

Definition at line 284 of file sound4.x1.cloud_dyn.hpp.

10.5.4.29 GetProcessChunkFrames

helper::DynFuncHolder< decltype(cloudx1_GetProcessChunkFrames) > sound4::cloudx1::dyn::CDyn↩
Lib::GetProcessChunkFrames

Definition at line 285 of file sound4.x1.cloud_dyn.hpp.

10.5.4.30 GetSampleRate

helper::DynFuncHolder< decltype(cloudx1_GetSampleRate) > sound4::cloudx1::dyn::CDynLib::Get↩
SampleRate

Definition at line 223 of file sound4.x1.cloud_dyn.hpp.

10.5.4.31 GetVersion

```
helper::DynFuncHolder< decltype(cloudx1_GetVersion ) > sound4::cloudx1::dyn::CDynLib::Get↩
Version
```

Definition at line 218 of file sound4.x1.cloud_dyn.hpp.

10.5.4.32 InitProcess

```
helper::DynFuncHolder< decltype(cloudx1_InitProcess ) > sound4::cloudx1::dyn::CDynLib::Init↩
Process
```

Definition at line 231 of file sound4.x1.cloud_dyn.hpp.

10.5.4.33 InitProcess2

```
helper::DynFuncHolder< decltype(cloudx1_InitProcess2 ) > sound4::cloudx1::dyn::CDynLib::Init↩
Process2
```

Definition at line 232 of file sound4.x1.cloud_dyn.hpp.

10.5.4.34 InitProcess3

```
helper::DynFuncHolder< decltype(cloudx1_InitProcess3 ) > sound4::cloudx1::dyn::CDynLib::Init↩
Process3
```

Definition at line 286 of file sound4.x1.cloud_dyn.hpp.

10.5.4.35 MonoToStereo

```
helper::DynFuncHolder< decltype(cloudx1_MonoToStereo ) > sound4::cloudx1::dyn::CDynLib::Mono↩
ToStereo
```

Definition at line 254 of file sound4.x1.cloud_dyn.hpp.

10.5.4.36 MonoToStereo_Planar

```
helper::DynFuncHolder< decltype(cloudx1_MonoToStereo_Planar ) > sound4::cloudx1::dyn::CDyn↩
Lib::MonoToStereo_Planar
```

Definition at line 272 of file sound4.x1.cloud_dyn.hpp.

10.5.4.37 NewClient

```
helper::DynFuncHolder< decltype(cloudx1_NewClient ) > sound4::cloudx1::dyn::CDynLib::NewClient
```

Definition at line 257 of file sound4.x1.cloud_dyn.hpp.

10.5.4.38 NewParameters

```
helper::DynFuncHolder< decltype(cloudx1_NewParameters ) > sound4::cloudx1::dyn::CDynLib::New↩
Parameters
```

Definition at line 226 of file sound4.x1.cloud_dyn.hpp.

10.5.4.39 PresetManager_InformChange

```
helper::DynFuncHolder< decltype(cloudx1_PresetManager_InformChange ) > sound4::cloudx1::dyn↩
::CDynLib::PresetManager_InformChange
```

Definition at line 281 of file sound4.x1.cloud_dyn.hpp.

10.5.4.40 ProcessAudio

```
helper::DynFuncHolder< decltype(cloudx1_ProcessAudio ) > sound4::cloudx1::dyn::CDynLib::↩
ProcessAudio
```

Definition at line 238 of file sound4.x1.cloud_dyn.hpp.

10.5.4.41 ProcessAudio_Planar

```
helper::DynFuncHolder< decltype(cloudx1_ProcessAudio_Planar ) > sound4::cloudx1::dyn::CDyn↩
Lib::ProcessAudio_Planar
```

Definition at line 239 of file sound4.x1.cloud_dyn.hpp.

10.5.4.42 ProcessJson

```
helper::DynFuncHolder< decltype(cloudx1_ProcessJson ) > sound4::cloudx1::dyn::CDynLib::↩
ProcessJson
```

Definition at line 259 of file sound4.x1.cloud_dyn.hpp.

10.5.4.43 SaveState

```
helper::DynFuncHolder< decltype(cloudx1_SaveState ) > sound4::cloudx1::dyn::CDynLib::SaveState
```

Definition at line 261 of file sound4.x1.cloud_dyn.hpp.

10.5.4.44 SetLoggerCallback

```
helper::DynFuncHolder< decltype(cloudx1_SetLoggerCallback ) > sound4::cloudx1::dyn::CDynLib↩
::SetLoggerCallback
```

Definition at line 224 of file sound4.x1.cloud_dyn.hpp.

10.5.4.45 SetLogSeverity

```
helper::DynFuncHolder< decltype(cloudx1_SetLogSeverity ) > sound4::cloudx1::dyn::CDynLib::↩
SetLogSeverity
```

Definition at line 225 of file sound4.x1.cloud_dyn.hpp.

10.5.4.46 SetMetadata

```
helper::DynFuncHolder< decltype(cloudx1_SetMetadata ) > sound4::cloudx1::dyn::CDynLib::Set↩
Metadata
```

Definition at line 275 of file sound4.x1.cloud_dyn.hpp.

10.5.4.47 SetMetadataMulti

```
helper::DynFuncHolder< decltype(cloudx1_SetMetadataMulti ) > sound4::cloudx1::dyn::CDynLib::↩
SetMetadataMulti
```

Definition at line 296 of file sound4.x1.cloud_dyn.hpp.

10.5.4.48 SetParameter

```
helper::DynFuncHolder< decltype(cloudx1_SetParameter ) > sound4::cloudx1::dyn::CDynLib::Set↩
Parameter
```

Definition at line 228 of file sound4.x1.cloud_dyn.hpp.

10.5.4.49 SetPresetManager

```
helper::DynFuncHolder< decltype(cloudx1_SetPresetManager ) > sound4::cloudx1::dyn::CDynLib::↩
SetPresetManager
```

Definition at line 280 of file sound4.x1.cloud_dyn.hpp.

10.5.4.50 StartUpdateThread

```
helper::DynFuncHolder< decltype(cloudx1_StartUpdateThread ) > sound4::cloudx1::dyn::CDynLib↩
::StartUpdateThread
```

Definition at line 235 of file sound4.x1.cloud_dyn.hpp.

10.5.4.51 StereoToMono

```
helper::DynFuncHolder< decltype(cloudx1_StereoToMono ) > sound4::cloudx1::dyn::CDynLib::↩
StereoToMono
```

Definition at line 253 of file sound4.x1.cloud_dyn.hpp.

10.5.4.52 StereoToMono_Planar

```
helper::DynFuncHolder< decltype(cloudx1_StereoToMono_Planar ) > sound4::cloudx1::dyn::CDyn↩
Lib::StereoToMono_Planar
```

Definition at line 271 of file sound4.x1.cloud_dyn.hpp.

10.5.4.53 StopUpdateThread

```
helper::DynFuncHolder< decltype(cloudx1_StopUpdateThread ) > sound4::cloudx1::dyn::CDynLib::↩
StopUpdateThread
```

Definition at line 236 of file sound4.x1.cloud_dyn.hpp.

10.5.4.54 TerminateProcess

```
helper::DynFuncHolder< decltype(cloudx1_TerminateProcess ) > sound4::cloudx1::dyn::CDynLib::↩
TerminateProcess
```

Definition at line 233 of file sound4.x1.cloud_dyn.hpp.

10.5.4.55 WaitUpdateThreadReady

```
helper::DynFuncHolder< decltype(cloudx1_WaitUpdateThreadReady ) > sound4::cloudx1::dyn::CDyn↩
Lib::WaitUpdateThreadReady
```

Definition at line 237 of file sound4.x1.cloud_dyn.hpp.

10.5.4.56 Webserver

```
helper::DynFuncHolder< decltype(cloudx1_Webserver ) > sound4::cloudx1::dyn::CDynLib::Webserver
```

Definition at line 265 of file sound4.x1.cloud_dyn.hpp.

10.5.4.57 Webserver_FreeString

```
helper::DynFuncHolder< decltype(cloudx1_Webserver_FreeString ) > sound4::cloudx1::dyn::CDyn↩
Lib::Webserver_FreeString
```

Definition at line 302 of file sound4.x1.cloud_dyn.hpp.

10.5.4.58 Webserver_GetAppHealth

```
helper::DynFuncHolder< decltype(cloudx1_Webserver_GetAppHealth ) > sound4::cloudx1::dyn::↩
CDynLib::Webserver_GetAppHealth
```

Definition at line 301 of file sound4.x1.cloud_dyn.hpp.

10.5.4.59 Webserver_SetAppHealth

```
helper::DynFuncHolder< decltype(cloudx1_Webserver_SetAppHealth ) > sound4::cloudx1::dyn::↩
CDynLib::Webserver_SetAppHealth
```

Definition at line 300 of file sound4.x1.cloud_dyn.hpp.

10.5.4.60 Webserver_Status

```
helper::DynFuncHolder< decltype(cloudx1_Webserver_Status ) > sound4::cloudx1::dyn::CDynLib::↩
Webserver_Status
```

Definition at line 267 of file sound4.x1.cloud_dyn.hpp.

10.5.4.61 Webserver_Stop

```
helper::DynFuncHolder< decltype(cloudx1_Webserver_Stop ) > sound4::cloudx1::dyn::CDynLib::↩
Webserver_Stop
```

Definition at line 266 of file sound4.x1.cloud_dyn.hpp.

10.5.4.62 Webserver_tcp

```
helper::DynFuncHolder< decltype(cloudx1_Webserver_tcp ) > sound4::cloudx1::dyn::CDynLib::↩
Webserver_tcp
```

Definition at line 263 of file sound4.x1.cloud_dyn.hpp.

10.5.4.63 Webserver_tcp2

```
helper::DynFuncHolder< decltype(cloudx1_Webserver_tcp2 ) > sound4::cloudx1::dyn::CDynLib::↩
Webserver_tcp2
```

Definition at line 264 of file sound4.x1.cloud_dyn.hpp.

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

- sound4.x1.cloud_dyn.hpp

10.6 sound4::cloudx1::dyn::helper::CDynLoader Class Reference

Helper to load dynamic library.

```
#include <sound4.x1.cloud_dyn.hpp>
```

Public Member Functions

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

Static Public Member Functions

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

10.6.1 Detailed Description

Definition at line 101 of file sound4.x1.cloud_dyn.hpp.

10.6.2 Constructor & Destructor Documentation

10.6.2.1 CDynLoader() [1/3]

```
sound4::cloudx1::dyn::helper::CDynLoader::CDynLoader ( )  [default]
```

10.6.2.2 CDynLoader() [2/3]

```
sound4::cloudx1::dyn::helper::CDynLoader::CDynLoader (
            CDynLoader &&  )  [default]
```

10.6.2.3 CDynLoader() [3/3]

```
sound4::cloudx1::dyn::helper::CDynLoader::CDynLoader (
            const CDynLoader &  )  [delete]
```

10.6.2.4 ~CDynLoader()

```
virtual sound4::cloudx1::dyn::helper::CDynLoader::~CDynLoader ( )  [inline], [virtual]
```

Definition at line 198 of file sound4.x1.cloud_dyn.hpp.

10.6.3 Member Function Documentation

10.6.3.1 Close()

```
void sound4::cloudx1::dyn::helper::CDynLoader::Close ( )  [inline]
```

Definition at line 159 of file sound4.x1.cloud_dyn.hpp.

10.6.3.2 GetSymbol()

```
template<typename T >
DynFuncHolder<T> sound4::cloudx1::dyn::helper::CDynLoader::GetSymbol (
            const std::string & name )  [inline]
```

Definition at line 172 of file sound4.x1.cloud_dyn.hpp.

10.6.3.3 GetSymbol_safe()

```
template<typename T >
DynFuncHolder<T> sound4::cloudx1::dyn::helper::CDynLoader::GetSymbol_safe (
            const std::string & name )  [inline]
```

Definition at line 180 of file sound4.x1.cloud_dyn.hpp.

10.6.3.4 GetThisLibraryPath()

```
static std::filesystem::path sound4::cloudx1::dyn::helper::CDynLoader::GetThisLibraryPath (
            void  )  [inline], [static]
```

Definition at line 184 of file sound4.x1.cloud_dyn.hpp.

10.6.3.5 IsOk()

```
bool sound4::cloudx1::dyn::helper::CDynLoader::IsOk ( ) const  [inline]
```

Definition at line 113 of file sound4.x1.cloud_dyn.hpp.

10.6.3.6 Load()

```
bool sound4::cloudx1::dyn::helper::CDynLoader::Load (
            const std::filesystem::path & dynlib,
            uint32_t loadflags = LOAD_LIBRARY_SEARCH_DLL_LOAD_DIR|LOAD_LIBRARY_SEARCH_APPLICATION↩
_DIR|LOAD_LIBRARY_SEARCH_USER_DIRS|LOAD_LIBRARY_SEARCH_SYSTEM32 )  [inline]
```

Definition at line 153 of file sound4.x1.cloud_dyn.hpp.

10.6.3.7 operator bool()

```
sound4::cloudx1::dyn::helper::CDynLoader::operator bool ( ) const  [inline]
```

Definition at line 110 of file sound4.x1.cloud_dyn.hpp.

10.6.3.8 operator=()

```
CDynLoader& sound4::cloudx1::dyn::helper::CDynLoader::operator= (
            CDynLoader const &  )  [delete]
```

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

- sound4.x1.cloud_dyn.hpp

10.7 sound4::helper::CDynLoader Class Reference

Helper to load dynamic library.

`#include <sound4cl.hpp>`

Public Member Functions

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

Static Public Member Functions

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

10.7.1 Detailed Description

Definition at line 130 of file sound4cl.hpp.

10.7.2 Constructor & Destructor Documentation

10.7.2.1 CDynLoader() [1/3]

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

10.7.2.2 CDynLoader() [2/3]

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

10.7.2.3 CDynLoader() [3/3]

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

10.7.2.4 ~CDynLoader()

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

Definition at line 227 of file sound4cl.hpp.

10.7.3 Member Function Documentation

10.7.3.1 Close()

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

Definition at line 188 of file sound4cl.hpp.

10.7.3.2 GetSymbol()

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

Definition at line 201 of file sound4cl.hpp.

10.7.3.3 GetSymbol_safe()

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

Definition at line 209 of file sound4cl.hpp.

10.7.3.4 GetThisLibraryPath()

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

Definition at line 213 of file sound4cl.hpp.

10.7.3.5 IsOk()

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

Definition at line 142 of file sound4cl.hpp.

10.7.3.6 Load()

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

Definition at line 182 of file sound4cl.hpp.

10.7.3.7 operator bool()

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

Definition at line 139 of file sound4cl.hpp.

10.7.3.8 operator=()

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

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

- sound4cl.hpp

10.8 sound4::CInstance Class Reference

Instance handling class.

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

Classes

- class CClient

Public Member Functions

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

Protected Member Functions

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

10.8.1 Detailed Description

Holds an instance of processing

Definition at line 750 of file sound4cl.hpp.

10.8.2 Constructor & Destructor Documentation

10.8.2.1 CInstance() [1/2]

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

Creates the instance, with default parameters

Definition at line 755 of file sound4cl.hpp.

10.8.2.2 CInstance() [2/2]

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

10.8.2.3 ~CInstance()

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

Destructor, stopping any running tasks

Definition at line 767 of file sound4cl.hpp.

10.8.3 Member Function Documentation

10.8.3.1 AddAudio()

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

Definition at line 1273 of file sound4cl.hpp.

10.8.3.2 Create()

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

Creates and initialize the instance, and starts its update thread

See also

sound4cl_InitProcess2

Parameters

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

Returns

false if failed

Definition at line 854 of file sound4cl.hpp.

10.8.3.3 GetAudio()

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

Definition at line 1279 of file sound4cl.hpp.

10.8.3.4 GetBufferIn()

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

Gets the input buffer used in ProcessAudio

Input has GetAudioInputCount() * GetChannelCount() channels

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

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

Definition at line 1128 of file sound4cl.hpp.

10.8.3.5 GetBufferOut()

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

Gets the output buffer used in ProcessAudio

Output has GetAudioOutputCount() * GetChannelCount() channels

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

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

Definition at line 1144 of file sound4cl.hpp.

10.8.3.6 GetChunkFrames()

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

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

The value reflects the frames_per_chunk parameter passed to Create

Returns

the number of audio frames in each chunk of processing

Definition at line 900 of file sound4cl.hpp.

10.8.3.7 GetEstimatedDelay()

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

Gets an estimated delay of current processing

See also

sound4cl_GetEstimatedDelay

Definition at line 1112 of file sound4cl.hpp.

10.8.3.8 GetMetadataInfos()

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

Get information on known metadata keys.

Returns

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

Definition at line 989 of file sound4cl.hpp.

10.8.3.9 GetParam()

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

Gets a parameter value

See also

sound4cl_GetParameter

Definition at line 793 of file sound4cl.hpp.

10.8.3.10 GetWebServerAppHealth()

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

Get the web server /apphealth status info

Definition at line 1066 of file sound4cl.hpp.

10.8.3.11 IsOk()

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

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

Definition at line 775 of file sound4cl.hpp.

10.8.3.12 NewClient()

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

Creates a low-level JSON client

See also

JSON syntax

Returns

a new client to communicate with the processing

Definition at line 1265 of file sound4cl.hpp.

10.8.3.13 OnUpdateThreadStart()

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

Definition at line 1011 of file sound4cl.hpp.

10.8.3.14 OnUpdateThreadStop()

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

Definition at line 1013 of file sound4cl.hpp.

10.8.3.15 operator=()

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

10.8.3.16 PresetManager_InformChange()

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

Inform that a change happened in the storage. This must be used only in 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 914 of file sound4cl.hpp.

10.8.3.17 ProcessAnyAudio()

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

Process audio (different formats, less optimized)

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

Attention

the return count is independent from the input count

Template Parameters

T	the C type for one audio sample

Parameters

input	audio to process

Returns

processed audio

Definition at line 1191 of file sound4cl.hpp.

10.8.3.18 ProcessAudio()

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

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

See also

sound4cl_ProcessAudio

Definition at line 1154 of file sound4cl.hpp.

10.8.3.19 ProcessAudio_Planar()

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

Process one audio chunk, but with planar audio

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

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

See also

sound4cl_ProcessAudio_Planar

Parameters

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

Definition at line 1173 of file sound4cl.hpp.

10.8.3.20 SetBus()

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

Set the instance bus

Definition at line 805 of file sound4cl.hpp.

10.8.3.21 SetMetadata()

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

Adds the specific metadata to the process.

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

Parameters

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 932 of file sound4cl.hpp.

10.8.3.22 SetMetadataMulti()

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

Adds multiple metadata to the process.

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

See also

CInstance::SetMetadata

Parameters

list	a map of key/value pair.

Definition at line 953 of file sound4cl.hpp.

10.8.3.23 SetParam()

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

Set a parameter value

Must be done before Create

See also

sound4cl_SetParameter

Definition at line 783 of file sound4cl.hpp.

10.8.3.24 SetPresetManager()

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

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

See also

sound4cl_SetPresetManager

Must be done before Create

Parameters

preset_manager	The custom preset manager

Definition at line 821 of file sound4cl.hpp.

10.8.3.25 SetWebServerAppHealth()

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

Set the web server /apphealth status info

Definition at line 1057 of file sound4cl.hpp.

10.8.3.26 StartWebServer()

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

Starts a Web server on requested ports

See also

sound4cl_Webserver

Definition at line 1020 of file sound4cl.hpp.

10.8.3.27 Stop()

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

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

See also

sound4cl_ExitProcess,
sound4cl_StopUpdateThread and
sound4cl_Webserver_Stop

Definition at line 1095 of file sound4cl.hpp.

10.8.3.28 StopWebServer()

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

Stops the Web server

See also

sound4cl_Webserver_Stop

Definition at line 1045 of file sound4cl.hpp.

10.8.3.29 Terminate()

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

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

See also

sound4cl_TerminateProcess

Definition at line 1084 of file sound4cl.hpp.

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

- sound4cl.hpp

10.9 sound4::cloudx1::CInstance Class Reference

```
#include <sound4.x1.cloud.hpp>
```

Classes

- class CClient

Public Member Functions

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

Protected Member Functions

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

10.9.1 Detailed Description

Instance handling class

Definition at line 385 of file sound4.x1.cloud.hpp.

10.9.2 Constructor & Destructor Documentation

10.9.2.1 CInstance() [1/2]

```
sound4::cloudx1::CInstance::CInstance ( )  [inline]
```

Creates the instance, with default parameters

Definition at line 390 of file sound4.x1.cloud.hpp.

10.9.2.2 CInstance() [2/2]

```
sound4::cloudx1::CInstance::CInstance (
            const CInstance &  )  [delete]
```

10.9.2.3 ~CInstance()

```
sound4::cloudx1::CInstance::~CInstance ( )  [inline]
```

Destructor, stopping any running tasks

Definition at line 401 of file sound4.x1.cloud.hpp.

10.9.3 Member Function Documentation

10.9.3.1 AddAudio()

```
template<typename T >
unsigned int sound4::cloudx1::CInstance::AddAudio (
            const T * payload,
            unsigned int nFrame )  [inline], [protected]
```

Definition at line 883 of file sound4.x1.cloud.hpp.

10.9.3.2 Create()

```
bool sound4::cloudx1::CInstance::Create (
            const std::string & LoginKey,
            const std::string & RadioName,
            const std::string & AWS_Access_ID,
            const std::string & AWS_Access_Secret,
            const std::filesystem::path & save_path,
            int json_port = 0,
            unsigned int frames_per_chunk = 64 )  [inline]
```

Creates and initialize the instance, and starts its update thread

See also

cloudx1_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 cloudx1_GetPossibleChunkSizeInFrames, or 0 to set default

Returns

false if failed

Definition at line 487 of file sound4.x1.cloud.hpp.

10.9.3.3 GetAudio()

```
template<typename T >
unsigned int sound4::cloudx1::CInstance::GetAudio (
            T * payload,
            unsigned int max_nFrame )  [inline], [protected]
```

Definition at line 889 of file sound4.x1.cloud.hpp.

10.9.3.4 GetBufferIn()

```
std::array<float,InputSampleSize>& sound4::cloudx1::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.x1.cloud.hpp.

10.9.3.5 GetBufferOut()

```
std::array<float,OutputSampleSize>& sound4::cloudx1::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.x1.cloud.hpp.

10.9.3.6 GetChunkFrames()

```
unsigned int sound4::cloudx1::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.x1.cloud.hpp.

10.9.3.7 GetEstimatedDelay()

```
unsigned int sound4::cloudx1::CInstance::GetEstimatedDelay ( )  [inline]
```

Gets an estimated delay of current processing

See also

cloudx1_GetEstimatedDelay

Definition at line 722 of file sound4.x1.cloud.hpp.

10.9.3.8 GetMetadataInfos()

```
std::vector< std::tuple<std::string,std::string> > sound4::cloudx1::CInstance::GetMetadata↩
Infos ( )  [inline]
```

Get information on known metadata keys.

Returns

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

Definition at line 602 of file sound4.x1.cloud.hpp.

10.9.3.9 GetParam()

```
std::string sound4::cloudx1::CInstance::GetParam (
            const std::string & name )  [inline]
```

Gets a parameter value

See also

cloudx1_GetParameter

Definition at line 427 of file sound4.x1.cloud.hpp.

10.9.3.10 GetWebServerAppHealth()

```
void sound4::cloudx1::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.x1.cloud.hpp.

10.9.3.11 IsOk()

```
bool sound4::cloudx1::CInstance::IsOk ( ) const  [inline]
```

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

Definition at line 409 of file sound4.x1.cloud.hpp.

10.9.3.12 NewClient()

```
std::shared_ptr<CClient> sound4::cloudx1::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.x1.cloud.hpp.

10.9.3.13 OnUpdateThreadStart()

```
virtual void sound4::cloudx1::CInstance::OnUpdateThreadStart ( )  [inline], [virtual]
```

Definition at line 622 of file sound4.x1.cloud.hpp.

10.9.3.14 OnUpdateThreadStop()

```
virtual void sound4::cloudx1::CInstance::OnUpdateThreadStop ( )  [inline], [virtual]
```

Definition at line 624 of file sound4.x1.cloud.hpp.

10.9.3.15 operator=()

```
CInstance& sound4::cloudx1::CInstance::operator= (
            const CInstance &  )  [delete]
```

10.9.3.16 PresetManager_InformChange()

```
void sound4::cloudx1::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.x1.cloud.hpp.

10.9.3.17 ProcessAnyAudio()

```
template<typename T >
std::vector<T> sound4::cloudx1::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 CLOUDX1_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.x1.cloud.hpp.

10.9.3.18 ProcessAudio()

```
void sound4::cloudx1::CInstance::ProcessAudio ( )  [inline]
```

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

See also

cloudx1_ProcessAudio

Definition at line 766 of file sound4.x1.cloud.hpp.

10.9.3.19 ProcessAudio_Planar()

```
void sound4::cloudx1::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

cloudx1_ProcessAudio_Planar

Parameters

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

Definition at line 785 of file sound4.x1.cloud.hpp.

10.9.3.20 SetMetadata()

```
void sound4::cloudx1::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.x1.cloud.hpp.

10.9.3.21 SetMetadataMulti()

```
void sound4::cloudx1::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.x1.cloud.hpp.

10.9.3.22 SetParam()

```
void sound4::cloudx1::CInstance::SetParam (
            const std::string & name,
            const std::string & value )  [inline]
```

Set a parameter value

Must be done before Create

See also

cloudx1_SetParameter

Definition at line 417 of file sound4.x1.cloud.hpp.

10.9.3.23 SetPresetManager()

```
void sound4::cloudx1::CInstance::SetPresetManager (
            CPresetLoader * preset_manager )  [inline]
```

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

See also

cloudx1_SetPresetManager

Must be done before Create

Parameters

preset_manager	The custom preset manager

Definition at line 445 of file sound4.x1.cloud.hpp.

10.9.3.24 SetWebServerAppHealth()

```
void sound4::cloudx1::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.x1.cloud.hpp.

10.9.3.25 StartWebServer()

```
bool sound4::cloudx1::CInstance::StartWebServer (
            int http_port,
            int https_port = 0 )  [inline]
```

Starts a Web server on requested ports

See also

cloudx1_Webserver

Definition at line 632 of file sound4.x1.cloud.hpp.

10.9.3.26 Stop()

```
void sound4::cloudx1::CInstance::Stop ( )  [inline]
```

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

See also

cloudx1_ExitProcess,
cloudx1_StopUpdateThread and
cloudx1_Webserver_Stop

Definition at line 703 of file sound4.x1.cloud.hpp.

10.9.3.27 StopWebServer()

```
void sound4::cloudx1::CInstance::StopWebServer ( )  [inline]
```

Stops the Web server

See also

cloudx1_Webserver_Stop

Definition at line 654 of file sound4.x1.cloud.hpp.

10.9.3.28 Terminate()

```
int sound4::cloudx1::CInstance::Terminate ( )  [inline]
```

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

See also

cloudx1_TerminateProcess

Definition at line 692 of file sound4.x1.cloud.hpp.

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

- sound4.x1.cloud.hpp

10.10 sound4::cloudx1::dyn::CInstance Class Reference

```
#include <sound4.x1.cloud_dyn.hpp>
```

Classes

- class CClient

Public Member Functions

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

Protected Member Functions

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

10.10.1 Detailed Description

Instance handling class

Definition at line 804 of file sound4.x1.cloud_dyn.hpp.

10.10.2 Constructor & Destructor Documentation

10.10.2.1 CInstance() [1/2]

```
sound4::cloudx1::dyn::CInstance::CInstance (
            CDynLib & dynlib )  [inline]
```

Creates the instance, with default parameters

Definition at line 809 of file sound4.x1.cloud_dyn.hpp.

10.10.2.2 CInstance() [2/2]

```
sound4::cloudx1::dyn::CInstance::CInstance (
            const CInstance &  )  [delete]
```

10.10.2.3 ~CInstance()

```
sound4::cloudx1::dyn::CInstance::~CInstance ( )  [inline]
```

Destructor, stopping any running tasks

Definition at line 821 of file sound4.x1.cloud_dyn.hpp.

10.10.3 Member Function Documentation

10.10.3.1 AddAudio()

```
template<typename T >
unsigned int sound4::cloudx1::dyn::CInstance::AddAudio (
            const T * payload,
            unsigned int nFrame )  [inline], [protected]
```

Definition at line 1328 of file sound4.x1.cloud_dyn.hpp.

10.10.3.2 Create()

```
bool sound4::cloudx1::dyn::CInstance::Create (
            const std::string & LoginKey,
            const std::string & RadioName,
            const std::string & Access_Key_ID,
            const std::string & Access_Key_Secret,
            const std::filesystem::path & save_path,
            int json_port = 0,
            unsigned int frames_per_chunk = 64 )  [inline]
```

Creates and initialize the instance, and starts its update thread

See also

cloudx1_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 cloudx1_GetPossibleChunkSizeInFrames, or 0 to set default

Returns

false if failed

Definition at line 909 of file sound4.x1.cloud_dyn.hpp.

10.10.3.3 GetAudio()

```
template<typename T >
unsigned int sound4::cloudx1::dyn::CInstance::GetAudio (
            T * payload,
            unsigned int max_nFrame )  [inline], [protected]
```

Definition at line 1334 of file sound4.x1.cloud_dyn.hpp.

10.10.3.4 GetBufferIn()

```
std::array<float,InputSampleSize>& sound4::cloudx1::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 1181 of file sound4.x1.cloud_dyn.hpp.

10.10.3.5 GetBufferOut()

```
std::array<float,OutputSampleSize>& sound4::cloudx1::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 1198 of file sound4.x1.cloud_dyn.hpp.

10.10.3.6 GetChunkFrames()

```
unsigned int sound4::cloudx1::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 952 of file sound4.x1.cloud_dyn.hpp.

10.10.3.7 GetEstimatedDelay()

```
unsigned int sound4::cloudx1::dyn::CInstance::GetEstimatedDelay ( )  [inline]
```

Gets an estimated delay of current processing

See also

cloudx1_GetEstimatedDelay

Definition at line 1165 of file sound4.x1.cloud_dyn.hpp.

10.10.3.8 GetMetadataInfos()

```
std::vector< std::tuple<std::string,std::string> > sound4::cloudx1::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 1041 of file sound4.x1.cloud_dyn.hpp.

10.10.3.9 GetParam()

```
std::string sound4::cloudx1::dyn::CInstance::GetParam (
            const std::string & name )  [inline]
```

Gets a parameter value

See also

cloudx1_GetParameter

Definition at line 847 of file sound4.x1.cloud_dyn.hpp.

10.10.3.10 GetWebServerAppHealth()

```
void sound4::cloudx1::dyn::CInstance::GetWebServerAppHealth (
            int & httpcode,
            std::string & contenttype,
            std::string & content )  [inline]
```

Get the web server /apphealth status info

Definition at line 1115 of file sound4.x1.cloud_dyn.hpp.

10.10.3.11 IsOk()

```
bool sound4::cloudx1::dyn::CInstance::IsOk ( ) const  [inline]
```

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

Definition at line 829 of file sound4.x1.cloud_dyn.hpp.

10.10.3.12 NewClient()

```
std::shared_ptr<CClient> sound4::cloudx1::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 1320 of file sound4.x1.cloud_dyn.hpp.

10.10.3.13 OnUpdateThreadStart()

```
virtual void sound4::cloudx1::dyn::CInstance::OnUpdateThreadStart ( )  [inline], [virtual]
```

Definition at line 1063 of file sound4.x1.cloud_dyn.hpp.

10.10.3.14 OnUpdateThreadStop()

```
virtual void sound4::cloudx1::dyn::CInstance::OnUpdateThreadStop ( )  [inline], [virtual]
```

Definition at line 1065 of file sound4.x1.cloud_dyn.hpp.

10.10.3.15 operator=()

```
CInstance& sound4::cloudx1::dyn::CInstance::operator= (
            const CInstance &  )  [delete]
```

10.10.3.16 PresetManager_InformChange()

```
void sound4::cloudx1::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 966 of file sound4.x1.cloud_dyn.hpp.

10.10.3.17 ProcessAnyAudio()

```
template<typename T >
std::vector<T> sound4::cloudx1::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 CLOUDX1_AUDIOFRAME_COUNT

Template Parameters

T	the C type for one audio sample

Parameters

input	audio to process

Returns

processed audio

Definition at line 1246 of file sound4.x1.cloud_dyn.hpp.

10.10.3.18 ProcessAudio()

```
void sound4::cloudx1::dyn::CInstance::ProcessAudio ( )  [inline]
```

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

See also

cloudx1_ProcessAudio

Definition at line 1209 of file sound4.x1.cloud_dyn.hpp.

10.10.3.19 ProcessAudio_Planar()

```
void sound4::cloudx1::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

cloudx1_ProcessAudio_Planar

Parameters

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

Definition at line 1228 of file sound4.x1.cloud_dyn.hpp.

10.10.3.20 SetMetadata()

```
void sound4::cloudx1::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 984 of file sound4.x1.cloud_dyn.hpp.

10.10.3.21 SetMetadataMulti()

```
void sound4::cloudx1::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 1005 of file sound4.x1.cloud_dyn.hpp.

10.10.3.22 SetParam()

```
void sound4::cloudx1::dyn::CInstance::SetParam (
            const std::string & name,
            const std::string & value )  [inline]
```

Set a parameter value

Must be done before Create

See also

cloudx1_SetParameter

Definition at line 837 of file sound4.x1.cloud_dyn.hpp.

10.10.3.23 SetPresetManager()

```
void sound4::cloudx1::dyn::CInstance::SetPresetManager (
            CPresetLoader * preset_manager )  [inline]
```

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

See also

cloudx1_SetPresetManager

Must be done before Create

Parameters

preset_manager	The custom preset manager

Definition at line 876 of file sound4.x1.cloud_dyn.hpp.

10.10.3.24 SetWebServerAppHealth()

```
void sound4::cloudx1::dyn::CInstance::SetWebServerAppHealth (
            int httpcode,
            const std::string & contenttype,
            const std::string & content )  [inline]
```

Set the web server /apphealth status info

Definition at line 1106 of file sound4.x1.cloud_dyn.hpp.

10.10.3.25 StartWebServer()

```
bool sound4::cloudx1::dyn::CInstance::StartWebServer (
            int http_port,
            int https_port = 0 )  [inline]
```

Starts a Web server on requested ports

See also

cloudx1_Webserver

Definition at line 1073 of file sound4.x1.cloud_dyn.hpp.

10.10.3.26 Stop()

```
void sound4::cloudx1::dyn::CInstance::Stop ( )  [inline]
```

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

See also

cloudx1_ExitProcess,
cloudx1_StopUpdateThread and
cloudx1_Webserver_Stop

Definition at line 1146 of file sound4.x1.cloud_dyn.hpp.

10.10.3.27 StopWebServer()

```
void sound4::cloudx1::dyn::CInstance::StopWebServer ( )  [inline]
```

Stops the Web server

See also

cloudx1_Webserver_Stop

Definition at line 1095 of file sound4.x1.cloud_dyn.hpp.

10.10.3.28 Terminate()

```
int sound4::cloudx1::dyn::CInstance::Terminate ( )  [inline]
```

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

See also

cloudx1_TerminateProcess

Definition at line 1135 of file sound4.x1.cloud_dyn.hpp.

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

- sound4.x1.cloud_dyn.hpp

10.11 cloudx1_CClientInstance Struct Reference

```
#include <sound4.x1.cloud.h>
```

10.11.1 Detailed Description

Abstract pointer for a client instance

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

- sound4.x1.cloud.h

10.12 cloudx1_CInstance Struct Reference

```
#include <sound4.x1.cloud.h>
```

10.12.1 Detailed Description

Abstract holder for a process instance

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

- sound4.x1.cloud.h

10.13 cloudx1_CParameters Struct Reference

`#include <sound4.x1.cloud.h>`

10.13.1 Detailed Description

Abstract holder for a process instance parameters

Definition at line 333 of file sound4.x1.cloud.h.

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

- sound4.x1.cloud.h

10.14 sound4::cloudx1::CPresetLoader Class Reference

`#include <sound4.x1.cloud.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

10.14.1 Detailed Description

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

Definition at line 367 of file sound4.x1.cloud.hpp.

10.14.2 Constructor & Destructor Documentation

10.14.2.1 CPresetLoader()

```
sound4::cloudx1::CPresetLoader::CPresetLoader ( )  [default]
```

10.14.2.2 ~CPresetLoader()

```
virtual sound4::cloudx1::CPresetLoader::~CPresetLoader ( )  [virtual], [default]
```

10.14.3 Member Function Documentation

10.14.3.1 Exists()

```
virtual bool sound4::cloudx1::CPresetLoader::Exists (
            const std::filesystem::path & name )  [pure virtual]
```

10.14.3.2 GetAll()

```
virtual std::vector<std::filesystem::path> sound4::cloudx1::CPresetLoader::GetAll ( )  [pure
virtual]
```

10.14.3.3 IsReadOnly()

```
virtual bool sound4::cloudx1::CPresetLoader::IsReadOnly ( )  [pure virtual]
```

10.14.3.4 Read()

```
virtual std::string sound4::cloudx1::CPresetLoader::Read (
            const std::filesystem::path & filename )  [pure virtual]
```

10.14.3.5 Remove()

```
virtual bool sound4::cloudx1::CPresetLoader::Remove (
            const std::filesystem::path & name )  [pure virtual]
```

10.14.3.6 Rename()

```
virtual bool sound4::cloudx1::CPresetLoader::Rename (
            const std::filesystem::path & from,
            const std::filesystem::path & to )  [pure virtual]
```

10.14.3.7 Write()

```
virtual bool sound4::cloudx1::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.x1.cloud.hpp

10.15 sound4::cloudx1::dyn::CPresetLoader Class Reference

`#include <sound4.x1.cloud_dyn.hpp>`

Inherited by MyPresetManager.

Public Member Functions

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

10.15.1 Detailed Description

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

Definition at line 786 of file sound4.x1.cloud_dyn.hpp.

10.15.2 Constructor & Destructor Documentation

10.15.2.1 CPresetLoader()

```
sound4::cloudx1::dyn::CPresetLoader::CPresetLoader ( )  [default]
```

10.15.2.2 ~CPresetLoader()

```
virtual sound4::cloudx1::dyn::CPresetLoader::~CPresetLoader ( )  [virtual], [default]
```

10.15.3 Member Function Documentation

10.15.3.1 Exists()

```
virtual bool sound4::cloudx1::dyn::CPresetLoader::Exists (
            const std::filesystem::path & name )  [pure virtual]
```

Implemented in MyPresetManager, and MyPresetManager.

10.15.3.2 GetAll()

```
virtual std::vector<std::filesystem::path> sound4::cloudx1::dyn::CPresetLoader::GetAll ( )
[pure virtual]
```

Implemented in MyPresetManager, and MyPresetManager.

10.15.3.3 IsReadOnly()

```
virtual bool sound4::cloudx1::dyn::CPresetLoader::IsReadOnly ( )  [pure virtual]
```

Implemented in MyPresetManager, and MyPresetManager.

10.15.3.4 Read()

```
virtual std::string sound4::cloudx1::dyn::CPresetLoader::Read (
            const std::filesystem::path & filename )  [pure virtual]
```

Implemented in MyPresetManager, and MyPresetManager.

10.15.3.5 Remove()

```
virtual bool sound4::cloudx1::dyn::CPresetLoader::Remove (
            const std::filesystem::path & name )  [pure virtual]
```

Implemented in MyPresetManager, and MyPresetManager.

10.15.3.6 Rename()

```
virtual bool sound4::cloudx1::dyn::CPresetLoader::Rename (
            const std::filesystem::path & from,
            const std::filesystem::path & to )  [pure virtual]
```

Implemented in MyPresetManager, and MyPresetManager.

10.15.3.7 Write()

```
virtual bool sound4::cloudx1::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.x1.cloud_dyn.hpp

10.16 sound4::CPresetLoader Class Reference

Custom preset handler helper.

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

Inherited by MyPresetManager.

Public Member Functions

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

10.16.1 Detailed Description

Derive from this class and call CInstance::SetPresetManager

Definition at line 730 of file sound4cl.hpp.

10.16.2 Constructor & Destructor Documentation

10.16.2.1 CPresetLoader()

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

10.16.2.2 ~CPresetLoader()

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

10.16.3 Member Function Documentation

10.16.3.1 Exists()

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

Implemented in MyPresetManager, and MyPresetManager.

10.16.3.2 GetAll()

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

Implemented in MyPresetManager, and MyPresetManager.

10.16.3.3 IsReadOnly()

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

Implemented in MyPresetManager, and MyPresetManager.

10.16.3.4 Read()

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

Implemented in MyPresetManager, and MyPresetManager.

10.16.3.5 Remove()

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

Implemented in MyPresetManager, and MyPresetManager.

10.16.3.6 Rename()

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

Implemented in MyPresetManager, and MyPresetManager.

10.16.3.7 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

10.17 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

10.17.1 Detailed Description

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

Definition at line 270 of file sound4cl.hpp.

10.17.2 Constructor & Destructor Documentation

10.17.2.1 CProcessor()

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

10.17.3 Member Function Documentation

10.17.3.1 AudioConvertFrom()

```
void sound4::CProcessor::AudioConvertFrom (
            const uint8_t * payload,
            float * output,
            size_t nSpl,
            sound4cl_SampleFormat fmt )  [inline]
```

Conversion helpers

Definition at line 563 of file sound4cl.hpp.

10.17.3.2 AudioConvertTo()

```
void sound4::CProcessor::AudioConvertTo (
            const float * input,
            uint8_t * payload,
            size_t nSpl,
            sound4cl_SampleFormat fmt )  [inline]
```

Definition at line 565 of file sound4cl.hpp.

10.17.3.3 AudioMonoFromLiveStereo()

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

Definition at line 575 of file sound4cl.hpp.

10.17.3.4 AudioMonoToLiveStereo()

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

Definition at line 577 of file sound4cl.hpp.

10.17.3.5 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 635 of file sound4cl.hpp.

10.17.3.6 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 645 of file sound4cl.hpp.

10.17.3.7 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 682 of file sound4cl.hpp.

10.17.3.8 GetChannelCount()

```
unsigned int sound4::CProcessor::GetChannelCount ( )  [inline]
```

Get the number of channels needed by this process

Returns

The process channels count

Definition at line 625 of file sound4cl.hpp.

10.17.3.9 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 596 of file sound4cl.hpp.

10.17.3.10 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 673 of file sound4cl.hpp.

10.17.3.11 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 664 of file sound4cl.hpp.

10.17.3.12 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 607 of file sound4cl.hpp.

10.17.3.13 GetPrefixSymbol()

```
template<typename T >
helper::DynFuncHolder<T> sound4::CProcessor::GetPrefixSymbol (
            const std::string & name )  [inline], [protected]
```

Definition at line 364 of file sound4cl.hpp.

10.17.3.14 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 655 of file sound4cl.hpp.

10.17.3.15 GetVersion()

```
std::string sound4::CProcessor::GetVersion ( )  [inline]
```

Get the process version string

Returns

version string

Definition at line 585 of file sound4cl.hpp.

10.17.3.16 IsOk()

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

Returns

true if OK.

false if not OK.

Definition at line 555 of file sound4cl.hpp.

10.17.3.17 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 379 of file sound4cl.hpp.

10.17.3.18 MonoToStereo()

```
void sound4::CProcessor::MonoToStereo (
            const float * input,
```

```
            float * output,
            size_t nSpl )  [inline]
```

Definition at line 569 of file sound4cl.hpp.

10.17.3.19 MonoToStereo_Planar()

```
void sound4::CProcessor::MonoToStereo_Planar (
            const float * input,
            float * outputL,
            float * outputR,
            size_t nSpl )  [inline]
```

Definition at line 573 of file sound4cl.hpp.

10.17.3.20 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 694 of file sound4cl.hpp.

10.17.3.21 SetLogSeverity()

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

Sets the log severity (global)

Definition at line 688 of file sound4cl.hpp.

10.17.3.22 StereoToMono()

```
void sound4::CProcessor::StereoToMono (
            const float * input,
            float * output,
            size_t nSpl )  [inline]
```

Definition at line 567 of file sound4cl.hpp.

10.17.3.23 StereoToMono_Planar()

```
void sound4::CProcessor::StereoToMono_Planar (
            const float * inputL,
            const float * inputR,
            float * output,
            size_t nSpl )  [inline]
```

Definition at line 571 of file sound4cl.hpp.

10.17.4 Friends And Related Function Documentation

10.17.4.1 CBus

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

Definition at line 276 of file sound4cl.hpp.

10.17.4.2 CInstance

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

Definition at line 277 of file sound4cl.hpp.

10.17.5 Member Data Documentation

10.17.5.1 fnAddAudio

```
helper::DynFuncHolder< decltype(sound4cl_AddAudio ) > sound4::CProcessor::fnAddAudio  [protected]
```

Definition at line 309 of file sound4cl.hpp.

10.17.5.2 fnAddPadAudio

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

Definition at line 310 of file sound4cl.hpp.

10.17.5.3 fnAudioConvertFrom

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

Definition at line 313 of file sound4cl.hpp.

10.17.5.4 fnAudioConvertTo

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

Definition at line 314 of file sound4cl.hpp.

10.17.5.5 fnAudioMonoFromLiveStereo

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

Definition at line 317 of file sound4cl.hpp.

10.17.5.6 fnAudioMonoToLiveStereo

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

Definition at line 318 of file sound4cl.hpp.

10.17.5.7 fnDeleteClient

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

Definition at line 320 of file sound4cl.hpp.

10.17.5.8 fnExitProcess

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

Definition at line 296 of file sound4cl.hpp.

10.17.5.9 fnFreeBus

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

Definition at line 352 of file sound4cl.hpp.

10.17.5.10 fnFreeJsonAnswer

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

Definition at line 322 of file sound4cl.hpp.

10.17.5.11 fnFreeMetadataInfos

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

Definition at line 339 of file sound4cl.hpp.

10.17.5.12 fnFreeParameters

helper::DynFuncHolder< decltype(sound4cl_FreeParameters) > sound4::CProcessor::fnFreeParameters
[protected]

Definition at line 289 of file sound4cl.hpp.

10.17.5.13 fnFreeParameterValue

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

Definition at line 292 of file sound4cl.hpp.

10.17.5.14 fnGetAudio

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

Definition at line 312 of file sound4cl.hpp.

10.17.5.15 fnGetAudioInputCount

helper::DynFuncHolder< decltype(sound4cl_GetAudioInputCount) > sound4::CProcessor::fnGetAudioInputCount [protected]

Definition at line 283 of file sound4cl.hpp.

10.17.5.16 fnGetAudioOutputCount

helper::DynFuncHolder< decltype(sound4cl_GetAudioOutputCount) > sound4::CProcessor::fnGetAudioOutputCount [protected]

Definition at line 284 of file sound4cl.hpp.

10.17.5.17 fnGetBufferIn

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

Definition at line 302 of file sound4cl.hpp.

10.17.5.18 fnGetBufferOut

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

Definition at line 303 of file sound4cl.hpp.

10.17.5.19 fnGetBytesFromFormat

helper::DynFuncHolder< decltype(sound4cl_GetBytesFromFormat) > sound4::CProcessor::fnGetBytesFromFormat [protected]

Definition at line 307 of file sound4cl.hpp.

10.17.5.20 fnGetChannelCount

helper::DynFuncHolder< decltype(sound4cl_GetChannelCount) > sound4::CProcessor::fnGetChannelCount [protected]

Definition at line 282 of file sound4cl.hpp.

10.17.5.21 fnGetChunkSizeInFrames

helper::DynFuncHolder< decltype(sound4cl_GetChunkSizeInFrames) > sound4::CProcessor::fnGetChunkSizeInFrames [protected]

Definition at line 281 of file sound4cl.hpp.

10.17.5.22 fnGetEstimatedDelay

helper::DynFuncHolder< decltype(sound4cl_GetEstimatedDelay) > sound4::CProcessor::fnGetEstimatedDelay [protected]

Definition at line 304 of file sound4cl.hpp.

10.17.5.23 fnGetFormatFromName

helper::DynFuncHolder< decltype(sound4cl_GetFormatFromName) > sound4::CProcessor::fnGetFormatFromName [protected]

Definition at line 306 of file sound4cl.hpp.

10.17.5.24 fnGetFormatName

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

Definition at line 305 of file sound4cl.hpp.

10.17.5.25 fnGetMaxPacketFrame

helper::DynFuncHolder< decltype(sound4cl_GetMaxPacketFrame) > sound4::CProcessor::fnGetMaxPacketFrame [protected]

Definition at line 308 of file sound4cl.hpp.

10.17.5.26 fnGetMetadataInfos

helper::DynFuncHolder< decltype(sound4cl_GetMetadataInfos) > sound4::CProcessor::fnGetMetadataInfos [protected]

Definition at line 338 of file sound4cl.hpp.

10.17.5.27 fnGetOutputCount

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

Definition at line 311 of file sound4cl.hpp.

10.17.5.28 fnGetParameter

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

Definition at line 291 of file sound4cl.hpp.

10.17.5.29 fnGetPossibleChunkSizeInFrames

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

Definition at line 346 of file sound4cl.hpp.

10.17.5.30 fnGetProcessChunkFrames

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

Definition at line 347 of file sound4cl.hpp.

10.17.5.31 fnGetSampleRate

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

Definition at line 285 of file sound4cl.hpp.

10.17.5.32 fnGetVersion

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

Definition at line 280 of file sound4cl.hpp.

10.17.5.33 fnInitProcess

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

Definition at line 293 of file sound4cl.hpp.

10.17.5.34 fnInitProcess2

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

Definition at line 294 of file sound4cl.hpp.

10.17.5.35 fnInitProcess3

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

Definition at line 348 of file sound4cl.hpp.

10.17.5.36 fnMonoToStereo

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

Definition at line 316 of file sound4cl.hpp.

10.17.5.37 fnMonoToStereo_Planar

```
helper::DynFuncHolder< decltype(sound4cl_MonoToStereo_Planar ) > sound4::CProcessor::fnMono↩
ToStereo_Planar  [protected]
```

Definition at line 334 of file sound4cl.hpp.

10.17.5.38 fnNewBus

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

Definition at line 351 of file sound4cl.hpp.

10.17.5.39 fnNewClient

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

Definition at line 319 of file sound4cl.hpp.

10.17.5.40 fnNewParameters

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

Definition at line 288 of file sound4cl.hpp.

10.17.5.41 fnPresetManager_InformChange

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

Definition at line 343 of file sound4cl.hpp.

10.17.5.42 fnProcessAudio

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

Definition at line 300 of file sound4cl.hpp.

10.17.5.43 fnProcessAudio_Planar

```
helper::DynFuncHolder< decltype(sound4cl_ProcessAudio_Planar ) > sound4::CProcessor::fn↩
ProcessAudio_Planar  [protected]
```

Definition at line 301 of file sound4cl.hpp.

10.17.5.44 fnProcessJson

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

Definition at line 321 of file sound4cl.hpp.

10.17.5.45 fnSaveState

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

Definition at line 323 of file sound4cl.hpp.

10.17.5.46 fnSetInstanceBus

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

Definition at line 353 of file sound4cl.hpp.

10.17.5.47 fnSetLoggerCallback

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

Definition at line 286 of file sound4cl.hpp.

10.17.5.48 fnSetLogSeverity

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

Definition at line 287 of file sound4cl.hpp.

10.17.5.49 fnSetMetadata

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

Definition at line 337 of file sound4cl.hpp.

10.17.5.50 fnSetMetadataMulti

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

Definition at line 356 of file sound4cl.hpp.

10.17.5.51 fnSetParameter

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

Definition at line 290 of file sound4cl.hpp.

10.17.5.52 fnSetPresetManager

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

Definition at line 342 of file sound4cl.hpp.

10.17.5.53 fnStartUpdateThread

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

Definition at line 297 of file sound4cl.hpp.

10.17.5.54 fnStereoToMono

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

Definition at line 315 of file sound4cl.hpp.

10.17.5.55 fnStereoToMono_Planar

```
helper::DynFuncHolder< decltype(sound4cl_StereoToMono_Planar ) > sound4::CProcessor::fn↩
StereoToMono_Planar [protected]
```

Definition at line 333 of file sound4cl.hpp.

10.17.5.56 fnStopUpdateThread

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

Definition at line 298 of file sound4cl.hpp.

10.17.5.57 fnTerminateProcess

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

Definition at line 295 of file sound4cl.hpp.

10.17.5.58 fnWaitUpdateThreadReady

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

Definition at line 299 of file sound4cl.hpp.

10.17.5.59 fnWebserver

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

Definition at line 328 of file sound4cl.hpp.

10.17.5.60 fnWebserver_FreeString

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

Definition at line 361 of file sound4cl.hpp.

10.17.5.61 fnWebserver_GetAppHealth

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

Definition at line 360 of file sound4cl.hpp.

10.17.5.62 fnWebserver_SetAppHealth

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

Definition at line 359 of file sound4cl.hpp.

10.17.5.63 fnWebserver_Status

helper::DynFuncHolder< decltype(sound4cl_Webserver_Status) > sound4::CProcessor::fnWebserver_Status [protected]

Definition at line 330 of file sound4cl.hpp.

10.17.5.64 fnWebserver_Stop

helper::DynFuncHolder< decltype(sound4cl_Webserver_Stop) > sound4::CProcessor::fnWebserver_Stop [protected]

Definition at line 329 of file sound4cl.hpp.

10.17.5.65 fnWebserver_tcp

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

Definition at line 326 of file sound4cl.hpp.

10.17.5.66 fnWebserver_tcp2

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

Definition at line 327 of file sound4cl.hpp.

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

- sound4cl.hpp

10.18 sound4::cloudx1::dyn::helper::DynFuncHolder< T > Class Template Reference

Helper to wrap a function pointer.

`#include <sound4.x1.cloud_dyn.hpp>`

Public Member Functions

- DynFuncHolder ()=default
- DynFuncHolder (DynFunc_t a_ptr)
- operator bool () const
- bool IsOk () const
- operator T* () const

10.18.1 Detailed Description

template<typename T>
class sound4::cloudx1::dyn::helper::DynFuncHolder< T >

Parameters

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

Definition at line 86 of file sound4.x1.cloud_dyn.hpp.

10.18.2 Constructor & Destructor Documentation

10.18.2.1 DynFuncHolder() [1/2]

```
template<typename T >
sound4::cloudx1::dyn::helper::DynFuncHolder< T >::DynFuncHolder ( )  [default]
```

10.18.2.2 DynFuncHolder() [2/2]

```
template<typename T >
sound4::cloudx1::dyn::helper::DynFuncHolder< T >::DynFuncHolder (
            DynFunc_t a_ptr )  [inline]
```

Definition at line 91 of file sound4.x1.cloud_dyn.hpp.

10.18.3 Member Function Documentation

10.18.3.1 IsOk()

```
template<typename T >
bool sound4::cloudx1::dyn::helper::DynFuncHolder< T >::IsOk ( ) const  [inline]
```

Definition at line 93 of file sound4.x1.cloud_dyn.hpp.

10.18.3.2 operator bool()

```
template<typename T >
sound4::cloudx1::dyn::helper::DynFuncHolder< T >::operator bool ( ) const  [inline]
```

Definition at line 92 of file sound4.x1.cloud_dyn.hpp.

10.18.3.3 operator T∗()

```
template<typename T >
sound4::cloudx1::dyn::helper::DynFuncHolder< T >::operator T* ( ) const  [inline]
```

Definition at line 94 of file sound4.x1.cloud_dyn.hpp.

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

- sound4.x1.cloud_dyn.hpp

10.19 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

10.19.1 Detailed Description

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

Parameters

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

Definition at line 115 of file sound4cl.hpp.

10.19.2 Constructor & Destructor Documentation

10.19.2.1 DynFuncHolder() [1/2]

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

10.19.2.2 DynFuncHolder() [2/2]

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

Definition at line 120 of file sound4cl.hpp.

10.19.3 Member Function Documentation

10.19.3.1 IsOk()

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

Definition at line 122 of file sound4cl.hpp.

10.19.3.2 operator bool()

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

Definition at line 121 of file sound4cl.hpp.

10.19.3.3 operator T*()

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

Definition at line 123 of file sound4cl.hpp.

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

- sound4cl.hpp

10.20 MyPresetManager Class Reference

[Custom preset manager]

Inherits sound4::cloudx1::dyn::CPresetLoader, and sound4::CPresetLoader.

Public Member Functions

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

10.20.1 Detailed Description

Definition at line 119 of file sound4.x1.cloud-pipe-dyn.cpp.

10.20.2 Constructor & Destructor Documentation

10.20.2.1 MyPresetManager() [1/2]

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

10.20.2.2 ~MyPresetManager() [1/2]

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

10.20.2.3 MyPresetManager() [2/2]

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

10.20.2.4 ~MyPresetManager() [2/2]

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

10.20.3 Member Function Documentation

10.20.3.1 Exists() [1/2]

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

Implements sound4::cloudx1::dyn::CPresetLoader.

Definition at line 137 of file sound4.x1.cloud-pipe-dyn.cpp.

10.20.3.2 Exists() [2/2]

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

Implements sound4::cloudx1::dyn::CPresetLoader.

Definition at line 121 of file sound4cl-pipe.cpp.

10.20.3.3 GetAll() [1/2]

```
virtual std::vector<std::filesystem::path> MyPresetManager::GetAll ( )  [inline], [override],
[virtual]
```

Implements sound4::cloudx1::dyn::CPresetLoader.

Definition at line 153 of file sound4.x1.cloud-pipe-dyn.cpp.

10.20.3.4 GetAll() [2/2]

```
virtual std::vector<std::filesystem::path> MyPresetManager::GetAll ( )  [inline], [override],
[virtual]
```

Implements sound4::cloudx1::dyn::CPresetLoader.

Definition at line 137 of file sound4cl-pipe.cpp.

10.20.3.5 IsReadOnly() [1/2]

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

Implements sound4::cloudx1::dyn::CPresetLoader.

Definition at line 136 of file sound4.x1.cloud-pipe-dyn.cpp.

10.20.3.6 IsReadOnly() [2/2]

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

Implements sound4::cloudx1::dyn::CPresetLoader.

Definition at line 120 of file sound4cl-pipe.cpp.

10.20.3.7 Read() [1/2]

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

Implements sound4::cloudx1::dyn::CPresetLoader.

Definition at line 163 of file sound4.x1.cloud-pipe-dyn.cpp.

10.20.3.8 Read() [2/2]

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

Implements sound4::cloudx1::dyn::CPresetLoader.

Definition at line 147 of file sound4cl-pipe.cpp.

10.20.3.9 Remove() [1/2]

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

Implements sound4::cloudx1::dyn::CPresetLoader.

Definition at line 142 of file sound4.x1.cloud-pipe-dyn.cpp.

10.20.3.10 Remove() [2/2]

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

Implements sound4::cloudx1::dyn::CPresetLoader.

Definition at line 126 of file sound4cl-pipe.cpp.

10.20.3.11 Rename() [1/2]

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

Implements sound4::cloudx1::dyn::CPresetLoader.

Definition at line 147 of file sound4.x1.cloud-pipe-dyn.cpp.

10.20.3.12 Rename() [2/2]

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

Implements sound4::cloudx1::dyn::CPresetLoader.

Definition at line 131 of file sound4cl-pipe.cpp.

10.20.3.13 SetupPath() [1/2]

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

Definition at line 124 of file sound4.x1.cloud-pipe-dyn.cpp.

10.20.3.14 SetupPath() [2/2]

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

Definition at line 108 of file sound4cl-pipe.cpp.

10.20.3.15 Write() [1/2]

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

Implements sound4::cloudx1::dyn::CPresetLoader.

Definition at line 173 of file sound4.x1.cloud-pipe-dyn.cpp.

10.20.3.16 Write() [2/2]

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

Implements sound4::cloudx1::dyn::CPresetLoader.

Definition at line 157 of file sound4cl-pipe.cpp.

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

- sound4.x1.cloud-pipe-dyn.cpp
- sound4cl-pipe.cpp

10.21 sound4::cloudx1::dyn::helper::SampleFormat< T > Struct Template Reference

Helper for Sample format types.

`#include <sound4.x1.cloud_dyn.hpp>`

10.21.1 Detailed Description

template<typename T>
struct sound4::cloudx1::dyn::helper::SampleFormat< T >

Template Parameters

T	the C type for one audio sample

Definition at line 523 of file sound4.x1.cloud_dyn.hpp.

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

- sound4.x1.cloud_dyn.hpp

10.22 sound4::cloudx1::helper::SampleFormat< T > Struct Template Reference

Helper for Sample format types.

`#include <sound4.x1.cloud.hpp>`

10.22.1 Detailed Description

template<typename T>
struct sound4::cloudx1::helper::SampleFormat< T >

Template Parameters

T	the C type for one audio sample

Definition at line 116 of file sound4.x1.cloud.hpp.

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

- sound4.x1.cloud.hpp

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

Helper for Sample format types.

`#include <sound4cl.hpp>`

10.23.1 Detailed Description

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

Template Parameters

T	the C type for one audio sample

Definition at line 85 of file sound4cl.hpp.

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

- sound4cl.hpp

10.24 sound4::cloudx1::dyn::helper::SampleFormat< float > Struct Reference

`#include <sound4.x1.cloud_dyn.hpp>`

Public Attributes

- const cloudx1_SampleFormat format = F32_NATIVE

10.24.1 Detailed Description

Definition at line 534 of file sound4.x1.cloud_dyn.hpp.

10.24.2 Member Data Documentation

10.24.2.1 format

`const cloudx1_SampleFormat sound4::cloudx1::dyn::helper::SampleFormat< float >::format = F32_NATIVE`

Definition at line 535 of file sound4.x1.cloud_dyn.hpp.

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

- sound4.x1.cloud_dyn.hpp

10.25 sound4::cloudx1::helper::SampleFormat< float > Struct Reference

`#include <sound4.x1.cloud.hpp>`

Public Attributes

- const cloudx1_SampleFormat format = F32_NATIVE

10.25.1 Detailed Description

Definition at line 127 of file sound4.x1.cloud.hpp.

10.25.2 Member Data Documentation

10.25.2.1 format

```
const cloudx1_SampleFormat sound4::cloudx1::helper::SampleFormat< float >::format = F32_NATIVE
```

Definition at line 128 of file sound4.x1.cloud.hpp.

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

- sound4.x1.cloud.hpp

10.26 sound4::helper::SampleFormat< float > Struct Reference

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

Public Attributes

- const sound4cl_SampleFormat format = F32_NATIVE

10.26.1 Detailed Description

Definition at line 96 of file sound4cl.hpp.

10.26.2 Member Data Documentation

10.26.2.1 format

```
const sound4cl_SampleFormat sound4::helper::SampleFormat< float >::format = F32_NATIVE
```

Definition at line 97 of file sound4cl.hpp.

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

- sound4cl.hpp

10.27 sound4::cloudx1::dyn::helper::SampleFormat< int16_t > Struct Reference

```
#include <sound4.x1.cloud_dyn.hpp>
```

Public Attributes

- const cloudx1_SampleFormat format = S16_NATIVE

10.27.1 Detailed Description

Definition at line 526 of file sound4.x1.cloud_dyn.hpp.

10.27.2 Member Data Documentation

10.27.2.1 format

```
const cloudx1_SampleFormat sound4::cloudx1::dyn::helper::SampleFormat< int16_t >::format =
S16_NATIVE
```

Definition at line 527 of file sound4.x1.cloud_dyn.hpp.

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

- sound4.x1.cloud_dyn.hpp

10.28 sound4::cloudx1::helper::SampleFormat< int16_t > Struct Reference

```
#include <sound4.x1.cloud.hpp>
```

Public Attributes

- const cloudx1_SampleFormat format = S16_NATIVE

10.28.1 Detailed Description

Definition at line 119 of file sound4.x1.cloud.hpp.

10.28.2 Member Data Documentation

10.28.2.1 format

```
const cloudx1_SampleFormat sound4::cloudx1::helper::SampleFormat< int16_t >::format = S16_NATIVE
```

Definition at line 120 of file sound4.x1.cloud.hpp.

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

- sound4.x1.cloud.hpp

10.29 sound4::helper::SampleFormat< int16_t > Struct Reference

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

Public Attributes

- const sound4cl_SampleFormat format = S16_NATIVE

10.29.1 Detailed Description

Definition at line 88 of file sound4cl.hpp.

10.29.2 Member Data Documentation

10.29.2.1 format

```
const sound4cl_SampleFormat sound4::helper::SampleFormat< int16_t >::format = S16_NATIVE
```

Definition at line 89 of file sound4cl.hpp.

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

- sound4cl.hpp

10.30 sound4::cloudx1::dyn::helper::SampleFormat< int32_t > Struct Reference

```
#include <sound4.x1.cloud_dyn.hpp>
```

Public Attributes

- const cloudx1_SampleFormat format = S32_NATIVE

10.30.1 Detailed Description

Definition at line 530 of file sound4.x1.cloud_dyn.hpp.

10.30.2 Member Data Documentation

10.30.2.1 format

```
const cloudx1_SampleFormat sound4::cloudx1::dyn::helper::SampleFormat< int32_t >::format =
S32_NATIVE
```

Definition at line 531 of file sound4.x1.cloud_dyn.hpp.

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

- sound4.x1.cloud_dyn.hpp

10.31 sound4::cloudx1::helper::SampleFormat< int32_t > Struct Reference

```
#include <sound4.x1.cloud.hpp>
```

Public Attributes

- const cloudx1_SampleFormat format = S32_NATIVE

10.31.1 Detailed Description

Definition at line 123 of file sound4.x1.cloud.hpp.

10.31.2 Member Data Documentation

10.31.2.1 format

```
const cloudx1_SampleFormat sound4::cloudx1::helper::SampleFormat< int32_t >::format = S32_NATIVE
```

Definition at line 124 of file sound4.x1.cloud.hpp.

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

- sound4.x1.cloud.hpp

10.32 sound4::helper::SampleFormat< int32_t > Struct Reference

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

Public Attributes

- const sound4cl_SampleFormat format = S32_NATIVE

10.32.1 Detailed Description

Definition at line 92 of file sound4cl.hpp.

10.32.2 Member Data Documentation

10.32.2.1 format

```
const sound4cl_SampleFormat sound4::helper::SampleFormat< int32_t >::format = S32_NATIVE
```

Definition at line 93 of file sound4cl.hpp.

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

- sound4cl.hpp

10.33 SOUND4_ProcessInfo Struct Reference

```
#include <sound4.x1.cloud.h>
```

Public Attributes

- size_t struct_size

  The size of total this structure, for compatibility checking.
- const char * name

  The name of the processor.
- const char * prefix

  The prefix of this processor functions.
- const char * shortname

  A short name of the processor, single word.
- const char * version

  Library version.

10.33.1 Detailed Description

Information on this processor.

See also

SOUND4_GetProcessInfo

Definition at line 189 of file sound4.x1.cloud.h.

10.33.2 Member Data Documentation

10.33.2.1 name

```
const char* SOUND4_ProcessInfo::name
```

Definition at line 191 of file sound4.x1.cloud.h.

10.33.2.2 prefix

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

Definition at line 192 of file sound4.x1.cloud.h.

10.33.2.3 shortname

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

Definition at line 193 of file sound4.x1.cloud.h.

10.33.2.4 struct_size

```
size_t SOUND4_ProcessInfo::struct_size
```

Definition at line 190 of file sound4.x1.cloud.h.

10.33.2.5 version

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

Definition at line 194 of file sound4.x1.cloud.h.

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

- sound4.x1.cloud.h

10.34 sound4cl_SOUND4_ProcessInfo Struct Reference

```
#include <sound4cl_cdef.h>
```

Public Attributes

- size_t struct_size
  The size of total this structure, for compatibility checking.
- const char * name
  The name of the processor.
- const char * prefix
  The prefix of this processor functions.
- const char * shortname
  A short name of the processor, single word.
- const char * version
  Library version.

10.34.1 Detailed Description

Definition at line 34 of file sound4cl_cdef.h.

10.34.2 Member Data Documentation

10.34.2.1 name

```
const char* sound4cl_SOUND4_ProcessInfo::name
```

Definition at line 36 of file sound4cl_cdef.h.

10.34.2.2 prefix

```
const char* sound4cl_SOUND4_ProcessInfo::prefix
```

Definition at line 37 of file sound4cl_cdef.h.

10.34.2.3 shortname

```
const char* sound4cl_SOUND4_ProcessInfo::shortname
```

Definition at line 38 of file sound4cl_cdef.h.

10.34.2.4 struct_size

```
size_t sound4cl_SOUND4_ProcessInfo::struct_size
```

Definition at line 35 of file sound4cl_cdef.h.

10.34.2.5 version

```
const char* sound4cl_SOUND4_ProcessInfo::version
```

Definition at line 39 of file sound4cl_cdef.h.

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

- sound4cl_cdef.h

Chapter 11

File Documentation

11.1 sound4.x1.cloud-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.x1.cloud.hpp"
```

Functions

- std::string GetEnv (const std::string &name)
- void ProcessLoop (sound4::cloudx1::CInstance &instance)
- void Usage (const char *progname)
- void MyLogger (sound4::cloudx1::LogSeverity severity, const std::string &msg)
- int main (int argc, const char **argv)

11.1.1 Function Documentation

11.1.1.1 GetEnv()

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

Definition at line 39 of file sound4.x1.cloud-pipe-cxx.cpp.

11.1.1.2 main()

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

[Setup and sanity checks]

[Setup and sanity checks]

[Create the processing instance]

[Create the processing instance]

[Communicating with the instance]

[Communicating with the instance]

Definition at line 104 of file sound4.x1.cloud-pipe-cxx.cpp.

11.1.1.3 MyLogger()

```
void MyLogger (
            sound4::cloudx1::LogSeverity severity,
            const std::string & msg )
```

Definition at line 99 of file sound4.x1.cloud-pipe-cxx.cpp.

11.1.1.4 ProcessLoop()

```
void ProcessLoop (
            sound4::cloudx1::CInstance & instance )
```

[Processing Loop]

[Processing Loop]

Definition at line 48 of file sound4.x1.cloud-pipe-cxx.cpp.

11.1.1.5 Usage()

```
void Usage (
            const char * progname )
```

Definition at line 79 of file sound4.x1.cloud-pipe-cxx.cpp.

11.2 sound4.x1.cloud-pipe-cxx.cpp

```
00001
00004 #include <unistd.h>
00005 #include <iostream>
00006 #include <strings.h>
00007 #include <sys/types.h>
00008 #include <filesystem>
00009 #if defined(_WIN32) || defined(WIN32)
00010     #include <io.h>
00011     #include <fcntl.h>
00012 #endif
00013
00014 #include "sound4.x1.cloud.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::cloudx1::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::cloudx1::InputSampleSize,
       stdin) != sound4::cloudx1::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::cloudx1::OutputSampleSize,
       stdout) != sound4::cloudx1::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::cloudx1::GetSampleRate()«" -ac
       "«sound4::cloudx1::GetChannelCount()«" -f f32le - | "«progname«" | ffplay -f f32le -ar
       "«sound4::cloudx1::GetSampleRate()«" -ac "«sound4::cloudx1::GetChannelCount()«" -i -" « std::endl;;
00092     #if defined(_WIN32) || defined(WIN32)
00093         std::cerr « "ffmpeg -f dshow -i audio=\"Microphone\" -ar "«sound4::cloudx1::GetSampleRate()«"
       -ac "«sound4::cloudx1::GetChannelCount()«" -f f32le - | "«progname«" | ffplay -f f32le -ar
       "«sound4::cloudx1::GetSampleRate()«" -ac "«sound4::cloudx1::GetChannelCount()«" -i -" « std::endl;;
00094     #else
00095         std::cerr « "ffmpeg -f pulse -i default -ar "«sound4::cloudx1::GetSampleRate()«" -ac
       "«sound4::cloudx1::GetChannelCount()«" -f f32le - | "«progname«" | ffplay -f f32le -ar
       "«sound4::cloudx1::GetSampleRate()«" -ac "«sound4::cloudx1::GetChannelCount()«" -i -" « std::endl;;
00096     #endif
00097 }
00098
00099 void MyLogger(sound4::cloudx1::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 (CLOUDX1_HAS_WEBSERVER)
00107     int listen_port = 8080;
00108 #endif // (CLOUDX1_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         store_path = std::filesystem::path(store_path_str);
00136         if (!std::filesystem::exists(store_path) || !std::filesystem::is_directory(store_path)) {
00137             std::cerr « "Storage folder '"«store_path.c_str()«"' is not a valid directory" «
      std::endl;
00138             Usage(progname);
00139             return 1;
00140         }
00141     }
00142
00144     sound4::cloudx1::SetLogSeverity(verbose5);
00145     sound4::cloudx1::SetLoggerCallback(MyLogger);
00146
00147     sound4::cloudx1::CInstance instance;
00148     instance.SetParam("ADMIN_USER","admin");
00149     instance.SetParam("ADMIN_SECRET","admin");
00151
00152     std::cerr « "Creating" « std::endl;
00154     if (!instance.Create(LoginKey, RadioName, KeyId, AccessKey, store_path)) {
00155         std::cerr « "Failed to create the processing instance" « std::endl;
00156         return 1;
00157     }
00159
00160 #if (CLOUDX1_HAS_WEBSERVER)
00161     std::cerr « "Starting web server" « std::endl;
00163     if (!instance.StartWebServer(listen_port,0)) {
00164         std::cerr « "Failed to create the web server on port " « listen_port « std::endl;
00165         return 1;
00166     }
00168 #endif // (CLOUDX1_HAS_WEBSERVER)
00169
00170     {
00172         auto client = instance.NewClient();
00173         auto answer = client->ProcessJson(R"JSON({"get":{"processorname":null}})JSON");
00174         std::cerr « "Request for processor name returned "«answer«std::endl;
00176     }
00177
00178     ProcessLoop(instance);
00179
00180     std::cerr « "Stopping now" « std::endl;
00181     instance.Stop();
00182
00183     std::cerr « "End" « std::endl;
00184     return 0;
00185 }
```

11.3 sound4.x1.cloud-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.x1.cloud_dyn.hpp"
```

Classes

- class MyPresetManager

  [Custom preset manager]

Functions

- std::string GetEnv (const std::string &name)
- void ProcessLoop (sound4::cloudx1::dyn::CInstance &instance)
- void Usage (const char *progname, sound4::cloudx1::dyn::CDynLib &dynlib)
- void MyLogger (sound4::cloudx1::dyn::LogSeverity severity, const std::string &msg)
- int main (int argc, const char **argv)

  [Custom preset manager]

11.3.1 Function Documentation

11.3.1.1 GetEnv()

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

Definition at line 51 of file sound4.x1.cloud-pipe-dyn.cpp.

11.3.1.2 main()

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

[Dynamic loading]

[Dynamic loading]

[Setup and sanity checks]

[Setup and sanity checks]

[Custom preset manager loading]

[Custom preset manager loading]

[Create the processing instance]

[Create the processing instance]

[Communicating with the instance]

[Communicating with the instance]

Definition at line 194 of file sound4.x1.cloud-pipe-dyn.cpp.

11.3.1.3 MyLogger()

```
void MyLogger (
            sound4::cloudx1::dyn::LogSeverity severity,
            const std::string & msg )
```

Definition at line 113 of file sound4.x1.cloud-pipe-dyn.cpp.

11.3.1.4 ProcessLoop()

```
void ProcessLoop (
            sound4::cloudx1::dyn::CInstance & instance )
```

[Processing Loop]

[Processing Loop]

Definition at line 60 of file sound4.x1.cloud-pipe-dyn.cpp.

11.3.1.5 Usage()

```
void Usage (
              const char * progname,
              sound4::cloudx1::dyn::CDynLib & dynlib )
```

Definition at line 91 of file sound4.x1.cloud-pipe-dyn.cpp.

11.4 sound4.x1.cloud-pipe-dyn.cpp

```
00001
00004 #include <unistd.h>
00005 #include <iostream>
00006 #include <strings.h>
00007 #include <sys/types.h>
00008 #include <filesystem>
00009 #include <fstream>
00010 #include <sstream>
00011 #if defined(_WIN32) || defined(WIN32)
00012     #include <io.h>
00013     #include <fcntl.h>
00014 #endif
00015
00016 #include "sound4.x1.cloud_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::cloudx1::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::cloudx1::dyn::InputSampleSize, stdin) != sound4::cloudx1::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::cloudx1::dyn::OutputSampleSize, stdout) != sound4::cloudx1::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::cloudx1::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::cloudx1::dyn::LogSeverity severity,const std::string& msg)
00114 {
00115     std::cerr « msg « std::endl;
00116 }
00117
00119 class MyPresetManager : public sound4::cloudx1::dyn::CPresetLoader {
00120 public:
00121     MyPresetManager() = default;
00122     virtual ~MyPresetManager() = default;
00123
00124     bool SetupPath(const std::filesystem::path& a_preset_path)
00125         {
00126             m_preset_path=a_preset_path;
00127             if ( !std::filesystem::exists(m_preset_path ) ) {
00128                 return false;
00129             }
00130             std::error_code ec;
00131             if ( !std::filesystem::is_directory(m_preset_path, ec) ) {
00132                 return false;
00133             }
00134             return true;
00135         }
00136     virtual bool IsReadOnly() override { return false; };
00137     virtual bool Exists(const std::filesystem::path &name) override
00138         {
00139             std::error_code ec;
00140             return std::filesystem::exists(GetRealPath(name), ec);
00141         }
00142     virtual bool Remove(const std::filesystem::path &name) override
00143         {
00144             std::error_code ec;
00145             return std::filesystem::remove(GetRealPath(name), ec);
00146         }
00147     virtual bool Rename(const std::filesystem::path &from, const std::filesystem::path &to) override
00148         {
00149             std::error_code ec;
00150             std::filesystem::rename(GetRealPath(from), GetRealPath(to), ec);
00151             return (!ec);
00152         }
00153     virtual std::vector<std::filesystem::path> GetAll() override
00154         {
00155             std::vector<std::filesystem::path> list;
00156             for (const auto & entry : std::filesystem::directory_iterator(m_preset_path.c_str(),
      std::filesystem::directory_options::skip_permission_denied)) {
00157                 if (entry.is_regular_file()) {
00158                     list.push_back(entry.path().lexically_relative(m_preset_path));
00159                 }
00160             }
00161             return list;
00162         }
00163     virtual std::string Read(const std::filesystem::path &filename) override
00164         {
00165             std::ifstream f(GetRealPath(filename));
00166             if ( f.fail() ) {
00167                 return std::string();
00168             }
00169             std::ostringstream ss;
00170             ss « f.rdbuf();
00171             return ss.str();
00172         }
00173     virtual bool Write(const std::filesystem::path &filename, const std::string &content) override
00174         {
00175             std::ofstream f(GetRealPath(filename));
00176             if ( f.fail() ) {
00177                 return false;
00178             }
00179             f « content;
00180             if ( f.fail() ) {
00181                 return false;
00182             }
00183             return true;
```

```
00184         }
00185 private:
00186     std::filesystem::path m_preset_path;
00187     std::filesystem::path GetRealPath(const std::filesystem::path &name)
00188         {
00189             return m_preset_path / name;
00190         }
00191 };
00193
00194 int main(int argc, const char **argv)
00195 {
00196 #if (CLOUDX1_HAS_WEBSERVER)
00197     int listen_port = 8080;
00198 #endif // (CLOUDX1_HAS_WEBSERVER)
00199     auto store_path_str=GetEnv("S4STORE_PATH");
00200
00201     // Get credentials from environment
00202     auto LoginKey=GetEnv("S4LOGINKEY");
00203     if (LoginKey.empty()) {
00204         LoginKey=GetEnv("S4USERID");
00205     }
00206     auto RadioName=GetEnv("RADIO_NAME");
00207     auto KeyId=GetEnv("ACCESS_KEY_ID");
00208     auto AccessKey=GetEnv("ACCESS_KEY_SECRET");
00209
00210     const char *progname=argv[0];
00211
00212
00214     sound4::cloudx1::dyn::CDynLib dynlib;
00215     if (!dynlib.Load()) {
00216         std::cerr « "Failed to load processing library" « std::endl;
00217         return 1;
00218     }
00220
00221     if (argc>1) {
00222         if ((strcasecmp(argv[1],"-h")==0) || (strcasecmp(argv[1],"--help")==0)) {
00223             Usage(progname,dynlib);
00224             return 0;
00225         }
00226     }
00227     if (LoginKey.empty() || RadioName.empty() || KeyId.empty() || AccessKey.empty()) {
00228         std::cerr « "Missing credentials in environment" « std::endl;
00229         Usage(progname,dynlib);
00230         return 1;
00231     }
00232     std::filesystem::path store_path;
00233     if (!store_path_str.empty()) {
00234         store_path = std::filesystem::path(store_path_str);
00235         if (!std::filesystem::exists(store_path) || !std::filesystem::is_directory(store_path)) {
00236             std::cerr « "Storage folder '"«store_path.c_str()«"' is not a valid directory" «
 std::endl;
00237             Usage(progname,dynlib);
00238             return 1;
00239         }
00240     }
00241
00243     sound4::cloudx1::dyn::SetLogSeverity(dynlib, verbose5);
00244     sound4::cloudx1::dyn::SetLoggerCallback(dynlib, MyLogger);
00245
00246     sound4::cloudx1::dyn::CInstance instance(dynlib);
00247     instance.SetParam("ADMIN_USER","admin");
00248     instance.SetParam("ADMIN_SECRET","admin");
00250
00252     MyPresetManager preset_manager;
00253     if (!store_path.empty()) {
00254         if (!preset_manager.SetupPath(store_path)) {
00255             std::cerr « "Failed to setup the custom preset manager on path " « store_path.c_str() «
 std::endl;
00256             return 1;
00257         }
00258         instance.SetPresetManager(&preset_manager);
00259     }
00261
00262     std::cerr « "Creating" « std::endl;
00264     if (!instance.Create(LoginKey, RadioName, KeyId, AccessKey, {})) {
00265         std::cerr « "Failed to create the processing instance" « std::endl;
00266         return 1;
00267     }
00269
00270 #if (CLOUDX1_HAS_WEBSERVER)
00271     std::cerr « "Starting web server" « std::endl;
00273     if (!instance.StartWebServer(listen_port,0)) {
00274         std::cerr « "Failed to create the web server on port " « listen_port « std::endl;
00275         return 1;
00276     }
00278 #endif // (CLOUDX1_HAS_WEBSERVER)
00279
```

```
00280     {
00282         auto client = instance.NewClient();
00283         auto answer = client->ProcessJson(R"JSON({"get":{"processorname":null}})JSON");
00284         std::cerr « "Request for processor name returned "«answer«std::endl;
00286     }
00287
00288     ProcessLoop(instance);
00289
00290     std::cerr « "Stopping now" « std::endl;
00291     instance.Stop();
00292
00293     std::cerr « "End" « std::endl;
00294     return 0;
00295 }
```

11.5 sound4.x1.cloud-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.x1.cloud.h>
```

Namespaces

- std

Macros

- #define SAMPLE_RATE 48000
- #define CHUNK_SIZE 64
- #define CHANNEL_COUNT 2

Functions

- void ProcessLoop (struct cloudx1_CInstance ∗pInstance)
- void Usage (const char ∗progname)
- void MyLogger (cloudx1_LogSeverity severe[[maybe_unused]], const char ∗str)
- int main (int argc, char ∗const argv[ ])

11.5.1 Macro Definition Documentation

11.5.1.1 CHANNEL_COUNT

```
#define CHANNEL_COUNT 2
```

Definition at line 28 of file sound4.x1.cloud-pipe.cpp.

11.5.1.2 CHUNK_SIZE

```
#define CHUNK_SIZE 64
```

Definition at line 27 of file sound4.x1.cloud-pipe.cpp.

11.5.1.3 SAMPLE_RATE

```
#define SAMPLE_RATE 48000
```

Definition at line 26 of file sound4.x1.cloud-pipe.cpp.

11.5.2 Function Documentation

11.5.2.1 main()

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

[Setup and sanity checks]

[Setup and sanity checks]

[Create the processing instance]

[Create the processing instance]

[Stopping the processing instance]

[Stopping the processing instance]

Definition at line 90 of file sound4.x1.cloud-pipe.cpp.

11.5.2.2 MyLogger()

```
void MyLogger (
            cloudx1_LogSeverity severe] [[maybe_unused],
            const char * str )
```

Definition at line 84 of file sound4.x1.cloud-pipe.cpp.

11.5.2.3 ProcessLoop()

```
void ProcessLoop (
            struct cloudx1_CInstance * pInstance )
```

[Processing Loop]

[Processing Loop]

Definition at line 31 of file sound4.x1.cloud-pipe.cpp.

11.5.2.4 Usage()

```
void Usage (
            const char * progname )
```

Definition at line 64 of file sound4.x1.cloud-pipe.cpp.

11.6 sound4.x1.cloud-pipe.cpp

```
00001
00004 #include <unistd.h>
00005 #include <iostream>
00006 #include <strings.h>
00007 #include <sys/types.h>
00008 #include <thread>           // NOTE: needs pthread link
00009 // Choose good filesystem version depending on compiler.
00010 #if __has_include(<filesystem>)
00011     #include <filesystem>
00012 #else //if __cpp_lib_experimental_filesystem
00013     // We need the alias from std::experimental::filesystem to std::filesystem
00014     #include <experimental/filesystem>
00015     namespace std {
00016         namespace filesystem = experimental::filesystem;
00017     }
00018 #endif
00019 #if defined(_WIN32) || defined(WIN32)
00020     #include <io.h>
00021     #include <fcntl.h>
00022     #include <stringapiset.h>   // UTF-8 conversion
00023 #endif
00024 #include <sound4.x1.cloud.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 cloudx1_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         cloudx1_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(cloudx1_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 cloudx1_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::filesystem::path fs_store_path;
```

```
00117     if (!store_path_str || store_path_str[0]==0) {
00118         fs_store_path=std::filesystem::current_path() / "state";
00119     } else {
00120         fs_store_path = std::filesystem::path(store_path_str);
00121     }
00122     if (!std::filesystem::exists(fs_store_path) || !std::filesystem::is_directory(fs_store_path)) {
00123         std::cerr « "Storage folder '"«fs_store_path.c_str()«"' is not a valid directory" « std::endl;
00124         Usage(progname);
00125         return 1;
00126     }
00127     std::string store_path;
00128     #ifdef _WIN32
00129     {
00130         // Convert path to UTF-8
00131         std::wstring wstr(fs_store_path.wstring());
00132         if (wstr.empty()) return {};
00133         int size = WideCharToMultiByte(CP_UTF8, 0, &wstr[0], (int)wstr.size(), NULL, 0, NULL, NULL);
00134         std::string str(size, 0);
00135         WideCharToMultiByte(CP_UTF8, 0, &wstr[0], (int)wstr.size(), &str[0], size, NULL, NULL);
00136         store_path=str;
00137     }
00138     #else // !_WIN32
00139         store_path=fs_store_path;
00140     #endif // !_WIN32
00141
00143     // Optional: Use our own logging, instead of default color output to stderr.
00144     //cloudx1_SetLoggerCallback(&MyLogger);
00145
00146     //cloudx1_SetLogSeverity(cloudx1_LogSeverity::info);
00147     cloudx1_SetLogSeverity(cloudx1_LogSeverity::verbose5);
00148
00149     std::cerr « "Process version: " « cloudx1_GetVersion() « std::endl;
00150     // Do some sanity checks
00151     if (cloudx1_GetSampleRate()!=SAMPLE_RATE) {
00152       std::cerr « "Bad library sampling rate, expecting '"«SAMPLE_RATE«"' but has "«
      cloudx1_GetSampleRate() « std::endl;
00153       return 1;
00154     }
00155     if (cloudx1_GetChunkSizeInFrames()!=CHUNK_SIZE) {
00156       std::cerr « "Bad library frame size, expecting '"«CHUNK_SIZE«"' but has "«
      cloudx1_GetChunkSizeInFrames() « std::endl;
00157       return 1;
00158     }
00159     if (cloudx1_GetChannelCount()!=CHANNEL_COUNT) {
00160       std::cerr « "Bad library channel count, expecting '"«CHANNEL_COUNT«"' but has "«
      cloudx1_GetChannelCount() « std::endl;
00161       return 1;
00162     }
00164
00166     // Setup parameters
00167     struct cloudx1_CParameters *params = cloudx1_NewParameters();
00168     // Optional: activate web authentication, set admin password
00169     // This creates the authentication only for a new storage path
00170     cloudx1_SetParameter(params,"ADMIN_USER", "admin");
00171     cloudx1_SetParameter(params,"ADMIN_SECRET", "admin");
00172
00173     // Create the processing instance
00174     pInstance=cloudx1_InitProcess2(pLoginKey,pRadioName,pKeyId,pAccessKey,store_path.c_str(), params);
00175     cloudx1_FreeParameters(params);
00176     if (!pInstance) {
00177       std::cerr « "Failed to create the process instance"« std::endl;
00178       return 1;
00179     }
00180
00181     // Start the update thread, without JSON server (low priority)
00182     std::thread updater([pInstance](){
00183        // will returned when stopped or destroyed
00184        std::cerr « "cloudx1_StartUpdateThread (no json port)" « std::endl;
00185        cloudx1_StartUpdateThread(pInstance, 0);
00186        std::cerr « "cloudx1_StartUpdateThread END" « std::endl;
00187      });
00188
00189     // This is not mandatory, but better be sure we are ready right now
00190     std::cerr « "Waiting update thread" « std::endl;
00191     if (cloudx1_WaitUpdateThreadReady(pInstance, 5000)<0) {
00192         std::cerr « "Timeout to create the process instance"« std::endl;
00193     }
00194     std::cerr « "Waiting update thread DONE" « std::endl;
00195
00196 #if (CLOUDX1_HAS_WEBSERVER)
00198     // Starts a web server, listening on port http://localhost:8080, https://localhost:8443
00199     std::cerr « "Starting web server" « std::endl;
00200     int listenport= 8080;
00201     int listenport_secure= 8443;
00202     uint64_t webserver= SOUND4_INVALID_WEBSERVER_ID;
00203     if (listenport || listenport_secure) {
00204         webserver = cloudx1_Webserver(listenport,listenport_secure,pInstance);
```

```
00205         if (webserver==SOUND4_INVALID_WEBSERVER_ID) {
00206             std::cerr « "Starting web server FAILED" « std::endl;
00207         } else {
00208             int web_status=cloudx1_Webserver_Status(webserver);
00209             if (listenport) {
00210                 if (web_status & SOUND4_WEBSERVER_HTTP_OK) {
00211                     std::cerr « "Web server listening on port " « listenport;
00212                 } else {
00213                     std::cerr « "Web server failed to listen on port " « listenport;
00214                 }
00215             }
00216             if (listenport_secure) {
00217                 if (web_status & SOUND4_WEBSERVER_HTTPS_OK) {
00218                     std::cerr « "Web server listening on secure port " « listenport_secure;
00219                 } else {
00220                     std::cerr « "Web server failed to listen on secure port " « listenport_secure;
00221                 }
00222             }
00223             std::cerr « "Starting web server DONE" « std::endl;
00224         }
00225     }
00227 #endif // (CLOUDX1_HAS_WEBSERVER)
00228
00229      // Force load a factory preset now
00230      // Could be used instead of setting a store_path
00231      /*{
00232          cloudx1_CClientInstance *pClient=cloudx1_NewClient(pInstance);
00233          const char *answer=cloudx1_ProcessJson(pClient, "{ \"factoryload\": \"All Programs\" }",
      NULL);
00234          std::cerr « "Loaded factory preset answer: " « answer« std::endl;
00235          cloudx1_FreeJsonAnswer(answer);
00236          cloudx1_DeleteClient(pClient);
00237      }*/
00239
00240     std::cerr « "Estimated delay " « ( cloudx1_GetEstimatedDelay(pInstance)*1000.0/SAMPLE_RATE ) « "
      ms" « std::endl;
00241
00242     ProcessLoop(pInstance);
00243
00245 #if (CLOUDX1_HAS_WEBSERVER)
00246     // Stopping the webserver
00247     std::cerr « "Stopping web server" « std::endl;
00248     cloudx1_Webserver_Stop(webserver, 1000);
00249     std::cerr « "Stopping web server DONE" « std::endl;
00250 #endif // (CLOUDX1_HAS_WEBSERVER)
00251
00252     // Stopping the processing instance
00253     std::cerr « "Stopping Update thread" « std::endl;
00254     cloudx1_StopUpdateThread(pInstance); // this will exit the update thread
00255     std::cerr « "Stopping Update thread DONE" « std::endl;
00256
00257     updater.join();
00258
00259     cloudx1_ExitProcess(pInstance);
00261
00262     return 0;
00263 }
```

11.7 sound4.x1.cloud.h File Reference

C interface for library.

```
#include <stdint.h>
```

Classes

- struct SOUND4_ProcessInfo

Macros

- #define CLOUDX1_DECL __declspec(dllimport)
- #define CLOUDX1_HAS_WEBSERVER 1
- #define CLOUDX1_HAS_CLOUDBUS 0
- #define CLOUDX1_AUDIOFRAME_COUNT 64
- #define CLOUDX1_CHANNEL_COUNT 2
- #define CLOUDX1_AUDIO_INPUT_COUNT 1
- #define CLOUDX1_AUDIO_OUTPUT_COUNT 1
- #define CLOUDX1_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(* cloudx1_loggerfn) (enum cloudx1_LogSeverity, const char *)
- typedef wchar_t fs_char
- typedef char *(* cloudx1_storage_reader) (const fs_char *filename, void *userdata)
- typedef void(* cloudx1_storage_reader_free) (char *content, void *userdata)
- typedef int(* cloudx1_storage_writer) (const fs_char *filename, const char *content, void *userdata)
- typedef int(* cloudx1_storage_exists) (const fs_char *filename, void *userdata)
- typedef fs_char **(* cloudx1_storage_getall) (void *userdata)
- typedef void(* cloudx1_storage_getall_free) (fs_char **all, void *userdata)
- typedef int(* cloudx1_storage_remove) (const fs_char *filename, void *userdata)
- typedef int(* cloudx1_storage_rename) (const fs_char *from, const fs_char *to, void *userdata)

Enumerations

- enum cloudx1_LogSeverity {
  none = 0 , fatal = 1 , error = 2 , warning = 3 ,
  info = 4 , verbose = 5 , verbose2 = 6 , verbose3 = 7 ,
  verbose4 = 8 , verbose5 = 9 }
- enum cloudx1_PresetChange_Kind { change_kind_created = 1 , change_kind_modified , change_kind_deleted }
- enum cloudx1_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 * cloudx1_GetVersion ()
- unsigned int cloudx1_GetChunkSizeInFrames ()
- unsigned int * cloudx1_GetPossibleChunkSizeInFrames ()
- unsigned int cloudx1_GetChannelCount ()
- unsigned int cloudx1_GetAudioInputCount ()
- unsigned int cloudx1_GetAudioOutputCount ()
- unsigned int cloudx1_GetSampleRate ()
- void cloudx1_SetLoggerCallback (cloudx1_loggerfn logger)

- void cloudx1_SetLogSeverity (enum cloudx1_LogSeverity severity)
- struct cloudx1_CParameters * cloudx1_NewParameters ()
- void cloudx1_FreeParameters (struct cloudx1_CParameters *params)
- void cloudx1_SetParameter (struct cloudx1_CParameters *params, const char *name, const char *value)
- const char * cloudx1_GetParameter (struct cloudx1_CParameters *params, const char *name)
- void cloudx1_FreeParameterValue (const char *value)
- struct cloudx1_CInstance * cloudx1_InitProcess (const char *LoginKey, const char *RadioName, const char *Access_Key_ID, const char *Access_Key_Secret, const char *save_path)
- struct cloudx1_CInstance * cloudx1_InitProcess2 (const char *LoginKey, const char *RadioName, const char *Access_Key_ID, const char *Access_Key_Secret, const char *save_path, const struct cloudx1_CParameters *parameters)
- struct cloudx1_CInstance * cloudx1_InitProcess3 (const char *LoginKey, const char *RadioName, const char *Access_Key_ID, const char *Access_Key_Secret, const char *save_path, const struct cloudx1_CParameters *parameters, unsigned int frames_per_chunk)
- unsigned int cloudx1_GetProcessChunkFrames (struct cloudx1_CInstance *instance)
- int cloudx1_TerminateProcess (struct cloudx1_CInstance *instance)
- void cloudx1_ExitProcess (struct cloudx1_CInstance *instance)
- void cloudx1_SetPresetManager (struct cloudx1_CParameters *params, cloudx1_storage_reader reader, cloudx1_storage_reader_free, cloudx1_storage_writer writer, cloudx1_storage_exists exists, cloudx1_storage_getall getall, cloudx1_storage_getall_free getall_free, cloudx1_storage_remove remove, cloudx1_storage_rename rename, int IsReadOnly, void *userdata)
- void cloudx1_PresetManager_InformChange (struct cloudx1_CInstance *instance, const fs_char *relative↩ _path, enum cloudx1_PresetChange_Kind change_kind)
- void cloudx1_SetMetadata (struct cloudx1_CInstance *instance, const char *key, const char *value)
- void cloudx1_SetMetadataMulti (struct cloudx1_CInstance *instance, const char **keyvalue)
- const char ** cloudx1_GetMetadataInfos (struct cloudx1_CInstance *instance)
- void cloudx1_FreeMetadataInfos (struct cloudx1_CInstance *instance, const char **infos)
- void cloudx1_StartUpdateThread (struct cloudx1_CInstance *instance, unsigned int port)
- void cloudx1_StopUpdateThread (struct cloudx1_CInstance *instance)
- int cloudx1_WaitUpdateThreadReady (struct cloudx1_CInstance *instance, int milliseconds)
- void cloudx1_ProcessAudio (struct cloudx1_CInstance *instance, const float *input, float *output)
- void cloudx1_ProcessAudio_Planar (struct cloudx1_CInstance *instance, float const *const *input, float *const *output)
- float * cloudx1_GetBufferIn (struct cloudx1_CInstance *instance)
- float * cloudx1_GetBufferOut (struct cloudx1_CInstance *instance)
- unsigned int cloudx1_GetEstimatedDelay (struct cloudx1_CInstance *instance)
- const char * cloudx1_GetFormatName (const enum cloudx1_SampleFormat fmt)
- enum cloudx1_SampleFormat cloudx1_GetFormatFromName (const char *name)
- unsigned int cloudx1_GetBytesFromFormat (const enum cloudx1_SampleFormat fmt)
- unsigned int cloudx1_GetMaxPacketFrame (struct cloudx1_CInstance *instance)
- unsigned int cloudx1_AddAudio (struct cloudx1_CInstance *instance, const uint8_t *payload, unsigned int nFrame, enum cloudx1_SampleFormat fmt)
- unsigned int cloudx1_AddPadAudio (struct cloudx1_CInstance *instance)
- unsigned int cloudx1_GetOutputCount (struct cloudx1_CInstance *instance)
- unsigned int cloudx1_GetAudio (struct cloudx1_CInstance *instance, uint8_t *payload, unsigned int max_↩ nFrame, enum cloudx1_SampleFormat fmt)
- void cloudx1_AudioConvertFrom (const uint8_t *payload, float *output, size_t nSpl, enum cloudx1_SampleFormat fmt)
- void cloudx1_AudioConvertTo (const float *input, uint8_t *payload, size_t nSpl, enum cloudx1_SampleFormat fmt)
- void cloudx1_StereoToMono (const float *input, float *output, size_t nFrame)
- void cloudx1_MonoToStereo (const float *input, float *output, size_t nFrame)
- void cloudx1_StereoToMono_Planar (const float *inputL, const float *inputR, float *output, size_t nFrame)
- void cloudx1_MonoToStereo_Planar (const float *input, float *outputL, float *outputR, size_t nFrame)
- void cloudx1_AudioMonoFromLiveStereo (const uint8_t *payload, float *output)

- void cloudx1_AudioMonoToLiveStereo (const float *input, uint8_t *payload)
- struct cloudx1_CClientInstance * cloudx1_NewClient (struct cloudx1_CInstance *instance)
- void cloudx1_DeleteClient (struct cloudx1_CClientInstance *client)
- const char * cloudx1_ProcessJson (struct cloudx1_CClientInstance *client, const char *json_str, int *need↩ _save)
- void cloudx1_FreeJsonAnswer (const char *json_str)
- int cloudx1_SaveState (struct cloudx1_CInstance *instance)
- uint64_t cloudx1_Webserver_tcp (unsigned int listenport, unsigned int listenport_secure, const char *socket_ip, unsigned int socket_port)
- uint64_t cloudx1_Webserver_tcp2 (unsigned int listenport, unsigned int listenport_secure, const char *socket_ip, unsigned int socket_port, const struct cloudx1_CParameters *parameters)
- uint64_t cloudx1_Webserver (unsigned int listenport, unsigned int listenport_secure, struct cloudx1_CInstance *instance)
- int cloudx1_Webserver_Stop (uint64_t id, int timeout_ms)
- int cloudx1_Webserver_Status (uint64_t id)
- void cloudx1_Webserver_SetAppHealth (struct cloudx1_CInstance *instance, int httpcode, const char *contenttype, const char *content)
- void cloudx1_Webserver_GetAppHealth (struct cloudx1_CInstance *instance, int *httpcode, char **contenttype, char **content)
- void cloudx1_Webserver_FreeString (char *str)

11.7.1 Macro Definition Documentation

11.7.1.1 CLOUDX1_DECL

```
#define CLOUDX1_DECL __declspec(dllimport)
```

Definition at line 14 of file sound4.x1.cloud.h.

11.7.1.2 CLOUDX1_HAS_CLOUDBUS

```
#define CLOUDX1_HAS_CLOUDBUS 0
```

Definition at line 19 of file sound4.x1.cloud.h.

11.7.1.3 CLOUDX1_HAS_WEBSERVER

```
#define CLOUDX1_HAS_WEBSERVER 1
```

Definition at line 18 of file sound4.x1.cloud.h.

11.8 sound4.x1.cloud.h

```
00001
00004 #pragma once
00005
00006 #include <stdint.h>
00007 #ifdef __unix__
00008     #include <sys/types.h>
00009     #define CLOUDX1_DECL        __attribute__ ((visibility ("default")))
00010 #elif defined(_WIN32) || defined(WIN32)
00011     #ifdef BUILDING_CLOUDX1_DLL
00012         #define CLOUDX1_DECL    __declspec(dllexport)
00013     #else
00014         #define CLOUDX1_DECL    __declspec(dllimport)
00015     #endif
00016 #endif
00017
00018 #define CLOUDX1_HAS_WEBSERVER 1
00019 #define CLOUDX1_HAS_CLOUDBUS 0
00020
00021 #ifdef __cplusplus
00022 extern "C" {
00023 #endif // __cplusplus
00024
00070 // ------------------------------------------------------------
00089 // ------------------------------------------------------------
00144 // ------------------------------------------------------------
00145
00146 // ------------------------------------------------------------
00158 #define CLOUDX1_AUDIOFRAME_COUNT      64
00164 #define CLOUDX1_CHANNEL_COUNT          2
00170 #define CLOUDX1_AUDIO_INPUT_COUNT      1
00176 #define CLOUDX1_AUDIO_OUTPUT_COUNT     1
00182 #define CLOUDX1_SAMPLE_RATE            48000
00183
00189 struct SOUND4_ProcessInfo {
00190     size_t struct_size;
00191     const char *name;
00192     const char *prefix;
00193     const char *shortname;
00194     const char *version;
00195 };
00196
00204 CLOUDX1_DECL const struct SOUND4_ProcessInfo* SOUND4_GetProcessInfo();
00205
00211 CLOUDX1_DECL const char *cloudx1_GetVersion();
00212
00225 CLOUDX1_DECL unsigned int cloudx1_GetChunkSizeInFrames();
00226
00235 CLOUDX1_DECL unsigned int* cloudx1_GetPossibleChunkSizeInFrames();
00236
00244 CLOUDX1_DECL unsigned int cloudx1_GetChannelCount();
00245
00254 CLOUDX1_DECL unsigned int cloudx1_GetAudioInputCount();
00255
00264 CLOUDX1_DECL unsigned int cloudx1_GetAudioOutputCount();
00265
00273 CLOUDX1_DECL unsigned int cloudx1_GetSampleRate();
00274
00278 enum cloudx1_LogSeverity {
00279     none = 0,
00280     fatal = 1,
00281     error = 2,
00282     warning = 3,
00283     info = 4,
00284     verbose = 5,
00285     verbose2 = 6,
00286     verbose3 = 7,
00287     verbose4 = 8,
00288     verbose5 = 9,
00289 };
00290
00297 typedef void (*cloudx1_loggerfn) (enum cloudx1_LogSeverity, const char *);
00298
00308 CLOUDX1_DECL void cloudx1_SetLoggerCallback( cloudx1_loggerfn logger );
00309
00317 CLOUDX1_DECL void cloudx1_SetLogSeverity( enum cloudx1_LogSeverity severity );
00318
00321 // ------------------------------------------------------------
00327 // ------------------------------------------------------------
00331 struct cloudx1_CInstance;
00332
00336 struct cloudx1_CParameters;
00337
00345 CLOUDX1_DECL struct cloudx1_CParameters *cloudx1_NewParameters();
00346
00350 CLOUDX1_DECL void cloudx1_FreeParameters(struct cloudx1_CParameters *params);
```

```
00351
00363 CLOUDX1_DECL void cloudx1_SetParameter(struct cloudx1_CParameters *params, const char *name, const
      char *value);
00364
00370 CLOUDX1_DECL const char *cloudx1_GetParameter(struct cloudx1_CParameters *params, const char *name);
00371
00377 CLOUDX1_DECL void cloudx1_FreeParameterValue(const char *value);
00378
00386 CLOUDX1_DECL struct cloudx1_CInstance *cloudx1_InitProcess(
00387         const char *LoginKey,
00388         const char *RadioName,
00389         const char *Access_Key_ID,
00390         const char *Access_Key_Secret,
00391         const char *save_path
00392     );
00393
00401 CLOUDX1_DECL struct cloudx1_CInstance *cloudx1_InitProcess2(
00402         const char *LoginKey,
00403         const char *RadioName,
00404         const char *Access_Key_ID,
00405         const char *Access_Key_Secret,
00406         const char *save_path,
00407         const struct cloudx1_CParameters *parameters
00408     );
00409
00454 CLOUDX1_DECL struct cloudx1_CInstance *cloudx1_InitProcess3(
00455         const char *LoginKey,
00456         const char *RadioName,
00457         const char *Access_Key_ID,
00458         const char *Access_Key_Secret,
00459         const char *save_path,
00460         const struct cloudx1_CParameters *parameters,
00461         unsigned int frames_per_chunk
00462     );
00463
00475 CLOUDX1_DECL unsigned int cloudx1_GetProcessChunkFrames(struct cloudx1_CInstance *instance);
00476
00486 CLOUDX1_DECL int cloudx1_TerminateProcess(struct cloudx1_CInstance *instance);
00487
00493 CLOUDX1_DECL void cloudx1_ExitProcess(struct cloudx1_CInstance *instance);
00494
00497 #if CLOUDX1_HAS_CLOUDBUS
00498 // ---------------------------------------------------------------
00507 struct cloudx1_CBus;
00508
00514 CLOUDX1_DECL struct cloudx1_CBus* cloudx1_NewBus();
00515
00521 CLOUDX1_DECL void cloudx1_FreeBus(struct cloudx1_CBus *bus);
00522
00532 CLOUDX1_DECL void cloudx1_SetInstanceBus(struct cloudx1_CParameters *params, struct cloudx1_CBus
      *bus);
00533
00534
00536 #endif // CLOUDX1_HAS_CLOUDBUS
00537
00538 // ---------------------------------------------------------------
00561 // ---------------------------------------------------------------
00562
00563 // OS dependent filesystem path char type
00564 #ifdef _WIN32
00565     typedef wchar_t fs_char;
00566 #else // !_WIN32
00567     typedef char fs_char;
00568 #endif // !_WIN32
00569
00577 typedef char * (*cloudx1_storage_reader) (const fs_char *filename, void* userdata);
00578
00585 typedef void (*cloudx1_storage_reader_free) (char *content, void* userdata);
00586
00595 typedef int (*cloudx1_storage_writer) (const fs_char *filename, const char *content, void* userdata);
00596
00604 typedef int (*cloudx1_storage_exists) (const fs_char *filename, void* userdata);
00605
00613 typedef fs_char** (*cloudx1_storage_getall) (void* userdata);
00614
00621 typedef void (*cloudx1_storage_getall_free) (fs_char**all, void* userdata);
00622
00630 typedef int (*cloudx1_storage_remove) (const fs_char *filename, void* userdata);
00631
00640 typedef int (*cloudx1_storage_rename) (const fs_char *from, const fs_char *to, void* userdata);
00641
00659 CLOUDX1_DECL void cloudx1_SetPresetManager(struct cloudx1_CParameters *params
00660     , cloudx1_storage_reader reader, cloudx1_storage_reader_free
00661     , cloudx1_storage_writer writer
00662     , cloudx1_storage_exists exists
00663     , cloudx1_storage_getall getall, cloudx1_storage_getall_free getall_free
00664     , cloudx1_storage_remove remove
```

```
00665     , cloudx1_storage_rename rename
00666     , int IsReadOnly
00667     , void* userdata
00668     );
00669
00673 enum cloudx1_PresetChange_Kind {
00674     change_kind_created = 1,
00675     change_kind_modified   ,
00676     change_kind_deleted    ,
00677 };
00678
00687 CLOUDX1_DECL void cloudx1_PresetManager_InformChange(struct cloudx1_CInstance *instance, const fs_char
       *relative_path, enum cloudx1_PresetChange_Kind change_kind);
00688
00691 // ----------------------------------------------------------------
00699 // ----------------------------------------------------------------
00700
00717 CLOUDX1_DECL void cloudx1_SetMetadata(struct cloudx1_CInstance *instance, const char* key, const char*
       value);
00718
00729 CLOUDX1_DECL void cloudx1_SetMetadataMulti(struct cloudx1_CInstance *instance, const char** keyvalue);
00730
00738 CLOUDX1_DECL const char** cloudx1_GetMetadataInfos(struct cloudx1_CInstance *instance);
00739
00746 CLOUDX1_DECL void cloudx1_FreeMetadataInfos(struct cloudx1_CInstance *instance, const char** infos);
00747
00750 // ----------------------------------------------------------------
00755 // ----------------------------------------------------------------
00756
00769 CLOUDX1_DECL void cloudx1_StartUpdateThread(struct cloudx1_CInstance *instance, unsigned int port);
00770
00776 CLOUDX1_DECL void cloudx1_StopUpdateThread(struct cloudx1_CInstance *instance);
00777
00785 CLOUDX1_DECL int cloudx1_WaitUpdateThreadReady(struct cloudx1_CInstance *instance, int milliseconds);
00786
00789 // ----------------------------------------------------------------
00794 // ----------------------------------------------------------------
00816 CLOUDX1_DECL void cloudx1_ProcessAudio(struct cloudx1_CInstance *instance, const float *input, float
       *output );
00817
00836 CLOUDX1_DECL void cloudx1_ProcessAudio_Planar(struct cloudx1_CInstance *instance
00837     , float const * const *input
00838     , float * const *output);
00839
00852 CLOUDX1_DECL float *cloudx1_GetBufferIn(struct cloudx1_CInstance *instance);
00853
00866 CLOUDX1_DECL float *cloudx1_GetBufferOut(struct cloudx1_CInstance *instance);
00867
00875 CLOUDX1_DECL unsigned int cloudx1_GetEstimatedDelay(struct cloudx1_CInstance *instance);
00876
00879 // ----------------------------------------------------------------
00886 // ----------------------------------------------------------------
00887     #ifdef _MSC_VER
00888         #define __BYTE_ORDER__  REG_DWORD
00889         #define __ORDER_LITTLE_ENDIAN__ REG_DWORD_LITTLE_ENDIAN
00890         #define __ORDER_BIG_ENDIAN__    REG_DWORD_BIG_ENDIAN
00891     #endif
00892
00896 enum cloudx1_SampleFormat {
00897     INVALID_FORMAT=0,
00898     S16_LE,
00899     S16_BE,
00900     S24_LE,
00901     S24_BE,
00902     S32_LE,
00903     S32_BE,
00904     F32_LE,
00905     F32_BE,
00906
00907     // Native definitions, only links to the physical ones
00908     #if __BYTE_ORDER__ == __ORDER_LITTLE_ENDIAN__
00909         S16_NATIVE=S16_LE,
00910         S24_NATIVE=S24_LE,
00911         S32_NATIVE=S32_LE,
00912         F32_NATIVE=F32_LE,
00913     #else
00914         S16_NATIVE=S16_BE,
00915         S24_NATIVE=S24_BE,
00916         S32_NATIVE=S32_BE,
00917         F32_NATIVE=F32_BE,
00918     #endif
00919 };
00920
00927 CLOUDX1_DECL const char *cloudx1_GetFormatName(const enum cloudx1_SampleFormat fmt);
00928
00935 CLOUDX1_DECL enum cloudx1_SampleFormat cloudx1_GetFormatFromName(const char *name);
00936
```

```
00943 CLOUDX1_DECL unsigned int cloudx1_GetBytesFromFormat(const enum cloudx1_SampleFormat fmt);
00944
00945
00952 CLOUDX1_DECL unsigned int cloudx1_GetMaxPacketFrame(struct cloudx1_CInstance *instance);
00953
00964 CLOUDX1_DECL unsigned int cloudx1_AddAudio(struct cloudx1_CInstance *instance, const uint8_t *payload,
       unsigned int nFrame, enum cloudx1_SampleFormat fmt );
00965
00972 CLOUDX1_DECL unsigned int cloudx1_AddPadAudio(struct cloudx1_CInstance *instance);
00973
00980 CLOUDX1_DECL unsigned int cloudx1_GetOutputCount(struct cloudx1_CInstance *instance);
00981
00991 CLOUDX1_DECL unsigned int cloudx1_GetAudio(struct cloudx1_CInstance *instance, uint8_t *payload,
       unsigned int max_nFrame, enum cloudx1_SampleFormat fmt );
00992
00996 // ------------------------------------------------------------
01000 // ------------------------------------------------------------
01011 CLOUDX1_DECL void cloudx1_AudioConvertFrom(const uint8_t *payload, float *output, size_t nSpl, enum
       cloudx1_SampleFormat fmt);
01012
01023 CLOUDX1_DECL void cloudx1_AudioConvertTo(const float *input, uint8_t *payload, size_t nSpl, enum
       cloudx1_SampleFormat fmt);
01024
01034 CLOUDX1_DECL void cloudx1_StereoToMono(const float *input, float *output, size_t nFrame);
01035
01045 CLOUDX1_DECL void cloudx1_MonoToStereo(const float *input, float *output, size_t nFrame);
01046
01057 CLOUDX1_DECL void cloudx1_StereoToMono_Planar(const float *inputL, const float *inputR, float *output,
       size_t nFrame);
01058
01069 CLOUDX1_DECL void cloudx1_MonoToStereo_Planar(const float *input, float *outputL, float *outputR,
       size_t nFrame);
01070
01077 CLOUDX1_DECL void cloudx1_AudioMonoFromLiveStereo(const uint8_t *payload, float *output);
01078
01085 CLOUDX1_DECL void cloudx1_AudioMonoToLiveStereo(const float *input, uint8_t *payload);
01086
01089 // ------------------------------------------------------------
01094 // ------------------------------------------------------------
01098 struct cloudx1_CClientInstance;
01099
01108 CLOUDX1_DECL struct cloudx1_CClientInstance *cloudx1_NewClient(struct cloudx1_CInstance *instance);
01109
01115 CLOUDX1_DECL void cloudx1_DeleteClient(struct cloudx1_CClientInstance *client);
01116
01130 CLOUDX1_DECL const char *cloudx1_ProcessJson(struct cloudx1_CClientInstance *client, const char
       *json_str, int *need_save);
01131
01137 CLOUDX1_DECL void cloudx1_FreeJsonAnswer(const char *json_str);
01138
01150 CLOUDX1_DECL int cloudx1_SaveState(struct cloudx1_CInstance *instance);
01151
01447 #if CLOUDX1_HAS_WEBSERVER
01448 // ------------------------------------------------------------
01456 // ------------------------------------------------------------
01475 CLOUDX1_DECL uint64_t cloudx1_Webserver_tcp(unsigned int listenport, unsigned int listenport_secure,
       const char *socket_ip, unsigned int socket_port);
01476
01477 // ------------------------------------------------------------
01497 CLOUDX1_DECL uint64_t cloudx1_Webserver_tcp2(unsigned int listenport, unsigned int listenport_secure,
       const char *socket_ip, unsigned int socket_port, const struct cloudx1_CParameters *parameters);
01498
01516 CLOUDX1_DECL uint64_t cloudx1_Webserver(unsigned int listenport, unsigned int listenport_secure,
       struct cloudx1_CInstance *instance);
01517
01527 CLOUDX1_DECL int cloudx1_Webserver_Stop(uint64_t id, int timeout_ms);
01528
01537 CLOUDX1_DECL int cloudx1_Webserver_Status(uint64_t id);
01538
01549 CLOUDX1_DECL void cloudx1_Webserver_SetAppHealth(struct cloudx1_CInstance *instance, int httpcode,
       const char *contenttype, const char *content);
01550
01559 CLOUDX1_DECL void cloudx1_Webserver_GetAppHealth(struct cloudx1_CInstance *instance, int* httpcode,
       char** contenttype, char** content);
01560
01566 CLOUDX1_DECL void cloudx1_Webserver_FreeString(char *str);
01567
01572 #define SOUND4_INVALID_WEBSERVER_ID            ((uint64_t)-1)
01573
01577 #define SOUND4_WEBSERVER_HTTP_OK               (1<<0)
01581 #define SOUND4_WEBSERVER_HTTPS_OK              (1<<1)
01582
01584 #endif // CLOUDX1_HAS_WEBSERVER
01585
01586 #ifdef __cplusplus
01587 } // extern "C"
01588 #endif // __cplusplus
```

11.9 sound4.x1.cloud.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.x1.cloud.h"
#include <stringapiset.h>
#include <wchar.h>
```

Classes

- struct sound4::cloudx1::helper::SampleFormat< T >

  Helper for Sample format types.
- struct sound4::cloudx1::helper::SampleFormat< int16_t >
- struct sound4::cloudx1::helper::SampleFormat< int32_t >
- struct sound4::cloudx1::helper::SampleFormat< float >
- class sound4::cloudx1::CPresetLoader
- class sound4::cloudx1::CInstance
- class sound4::cloudx1::CInstance::CClient

Namespaces

- sound4
- sound4::cloudx1
- sound4::cloudx1::helper

Macros

- #define fs_strdup wcsdup

  helper for OS dependant file name copy

Typedefs

- using sound4::cloudx1::SampleFormat = cloudx1_SampleFormat
- using sound4::cloudx1::LogSeverity = cloudx1_LogSeverity
- using sound4::cloudx1::log_cb_t = std::function< void(LogSeverity, const std::string &)>
- using sound4::cloudx1::PresetChange_Kind = cloudx1_PresetChange_Kind

Variables

- const size_t sound4::cloudx1::ChunkFrames =CLOUDX1_AUDIOFRAME_COUNT
- const size_t sound4::cloudx1::InputChannels =CLOUDX1_CHANNEL_COUNT∗CLOUDX1_AUDIO_INPUT_COUNT
- const size_t sound4::cloudx1::OutputChannels =CLOUDX1_CHANNEL_COUNT∗CLOUDX1_AUDIO_OUTPUT_COUNT
- const size_t sound4::cloudx1::InputSampleSize =ChunkFrames∗InputChannels
- const size_t sound4::cloudx1::OutputSampleSize =ChunkFrames∗OutputChannels
- const size_t sound4::cloudx1::SampleRate =CLOUDX1_SAMPLE_RATE

11.9.1 Macro Definition Documentation

11.9.1.1 fs_strdup

```
#define fs_strdup wcsdup
```

Definition at line 25 of file sound4.x1.cloud.hpp.

11.10 sound4.x1.cloud.hpp

```
00001
00004 #pragma once
00005
00006 #include <string>
00007 #include <exception>
00008 #include <thread>
00009 #include <functional>
00010 #include <memory>
00011 #include <array>
00012 #include <assert.h>
00013 #include <stdexcept>
00014 #include <filesystem>
00015 #include <string.h>
00016
00017 #include "sound4.x1.cloud.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 *cloudx1_custom_reader(const fs_char *filename, void* userdata);
00034     static void cloudx1_custom_reader_free(char *content, void* userdata);
00035     static int cloudx1_custom_writer(const fs_char *filename, const char *content, void* userdata);
00036     static int cloudx1_custom_exists(const fs_char *filename, void* userdata);
00037     static fs_char** cloudx1_custom_getall(void* userdata);
00038     static void cloudx1_custom_getall_free(fs_char** all, void* userdata);
00039     static int cloudx1_custom_remove(const fs_char *filename, void* userdata);
00040     static int cloudx1_custom_rename(const fs_char *from, const fs_char *to, void* userdata);
00041 };
00042
00046 namespace sound4 {
00050 namespace cloudx1 {
00051     static constexpr const char *process_name = "SOUND4 x1.CLOUD";
00052     static constexpr const char *process_shortname = "x1";
00053     using SampleFormat = cloudx1_SampleFormat;
00057     const size_t ChunkFrames=CLOUDX1_AUDIOFRAME_COUNT;
00061     const size_t InputChannels=CLOUDX1_CHANNEL_COUNT*CLOUDX1_AUDIO_INPUT_COUNT;
00065     const size_t OutputChannels=CLOUDX1_CHANNEL_COUNT*CLOUDX1_AUDIO_OUTPUT_COUNT;
00069     const size_t InputSampleSize=ChunkFrames*InputChannels;
00073     const size_t OutputSampleSize=ChunkFrames*OutputChannels;
```

```
00077     const size_t SampleRate=CLOUDX1_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 cloudx1_SampleFormat format = S16_NATIVE;
00121         };
00122             template<>
00123         struct SampleFormat<int32_t> {
00124             const cloudx1_SampleFormat format = S32_NATIVE;
00125         };
00126             template<>
00127         struct SampleFormat<float> {
00128             const cloudx1_SampleFormat format = F32_NATIVE;
00129         };
00130
00131
00136         static inline void AudioConvertFrom(const uint8_t *payload, float *output, size_t nSpl,
      cloudx1_SampleFormat fmt) {
00137             cloudx1_AudioConvertFrom(payload, output, nSpl, fmt);
00138         }
00139         static inline void AudioConvertTo(const float *input, uint8_t *payload, size_t nSpl,
      cloudx1_SampleFormat fmt) {
00140             cloudx1_AudioConvertTo(input, payload, nSpl, fmt);
00141         }
00142         static inline void StereoToMono(const float *input, float *output, size_t nSpl) {
00143             cloudx1_StereoToMono(input, output, nSpl);
00144         }
00145         static inline void MonoToStereo(const float *input, float *output, size_t nSpl) {
00146             cloudx1_MonoToStereo(input, output, nSpl);
00147         }
00148         static inline void StereoToMono_Planar(const float *inputL, const float *inputR, float
      *output, size_t nSpl) {
00149             cloudx1_StereoToMono_Planar(inputL, inputR, output, nSpl);
00150         }
00151         static inline void MonoToStereo_Planar(const float *input, float *outputL, float *outputR,
      size_t nSpl) {
00152             cloudx1_MonoToStereo_Planar(input, outputL, outputR, nSpl);
00153         }
00154         static inline void AudioMonoFromLiveStereo(const uint8_t *payload, float *output) {
00155             cloudx1_AudioMonoFromLiveStereo(payload, output);
00156         }
00157         static inline void AudioMonoToLiveStereo(const float *input, uint8_t *payload) {
00158             cloudx1_AudioMonoToLiveStereo(input, payload);
00159         }
00160
00161     }; // namespace helper
00162
00168     static inline std::string GetVersion()
00169         {
00170             return std::string(cloudx1_GetVersion());
00171         }
00172
00183     static inline unsigned int GetChunkSizeInFrames()
00184         {
00185             return cloudx1_GetChunkSizeInFrames();
00186         }
00187
00196     static inline std::vector<unsigned int> GetPossibleChunkSizeInFrames()
00197         {
00198             std::vector<unsigned int> list;
00199             for (unsigned int* src=cloudx1_GetPossibleChunkSizeInFrames();*src;src++) {
00200                 list.push_back(*src);
00201             }
00202             return list;
00203         }
```

```
00204
00212     static inline unsigned int GetChannelCount()
00213         {
00214             return cloudx1_GetChannelCount();
00215         }
00216
00225     static inline unsigned int GetAudioInputCount()
00226         {
00227             return cloudx1_GetAudioInputCount();
00228         }
00229
00238     static inline unsigned int GetAudioOutputCount()
00239         {
00240             return cloudx1_GetAudioOutputCount();
00241         }
00242
00250     static inline unsigned int GetSampleRate()
00251         {
00252             return cloudx1_GetSampleRate();
00253         }
00254
00260     static inline void SanityCheck(bool a_bCheckFrames = true)
00261     {
00262         // Do some sanity checks
00263         if (cloudx1_GetSampleRate()!=CLOUDX1_SAMPLE_RATE) {
00264             throw std::runtime_error("Bad library sampling rate");
00265         }
00266         if (a_bCheckFrames && cloudx1_GetChunkSizeInFrames()!=ChunkFrames) {
00267             throw std::runtime_error("Bad library frame size");
00268         }
00269         if (cloudx1_GetChannelCount()!=CLOUDX1_CHANNEL_COUNT) {
00270             throw std::runtime_error("Bad library channel count");
00271         }
00272         if (cloudx1_GetAudioInputCount()!=CLOUDX1_AUDIO_INPUT_COUNT) {
00273             throw std::runtime_error("Bad library input count");
00274         }
00275         if (cloudx1_GetAudioOutputCount()!=CLOUDX1_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(cloudx1_GetFormatName(fmt));
00289         }
00290
00297     static inline SampleFormat GetFormatFromName(const std::string& name)
00298         {
00299             return cloudx1_GetFormatFromName(name.c_str());
00300         }
00301
00308     static inline unsigned int GetBytesFromFormat(const SampleFormat fmt)
00309         {
00310             return cloudx1_GetBytesFromFormat(fmt);
00311         }
00312
00316     using LogSeverity = cloudx1_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(cloudx1_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             cloudx1_SetLogSeverity(severity);
00337         }
00341     static inline void SetLoggerCallback(log_cb_t cb)
00342         {
00343             _log_cb=cb;
00344             cloudx1_SetLoggerCallback(&_log_cb_c);
00345         }
00346
00347 #if CLOUDX1_HAS_CLOUDBUS
00348     class CBus {
00349         public:
00350             CBus()
00351                 : m_bus(cloudx1_NewBus()) {}
00352             ~CBus()
00353                 {
00354                     cloudx1_FreeBus(m_bus);
00355                     m_bus = nullptr;
00356                 }
00357             cloudx1_CBus *Get() const { return m_bus; }
00358         private:
```

```
00359             cloudx1_CBus *m_bus;
00360     };
00361 #endif // CLOUDX1_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 = cloudx1_PresetChange_Kind;
00381
00385     class CInstance {
00386     public:
00390         CInstance()
00391             {
00392                 params = cloudx1_NewParameters();
00393                 if (!params) throw std::bad_alloc();
00394             }
00395         // non-copyable
00396         CInstance( const CInstance& ) = delete;
00397         CInstance& operator=( const CInstance& ) = delete;
00401         ~CInstance()
00402             {
00403                 Stop();
00404                 if (params) cloudx1_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                 cloudx1_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=cloudx1_GetParameter(params, name.c_str());
00431                 std::string ret(c_value);
00432                 cloudx1_FreeParameterValue(c_value);
00433                 return ret;
00434             }
00435
00445         void SetPresetManager(CPresetLoader *preset_manager)
00446             {
00447                 if (!preset_manager) return;
00448                 assert(params);
00449                 cloudx1_SetPresetManager(params,
00450                     cloudx1_custom_reader,
00451                     cloudx1_custom_reader_free,
00452                     cloudx1_custom_writer,
00453                     cloudx1_custom_exists,
00454                     cloudx1_custom_getall,
00455                     cloudx1_custom_getall_free,
00456                     cloudx1_custom_remove,
00457                     cloudx1_custom_rename,
00458                     preset_manager->IsReadOnly(),
00459                     preset_manager
00460                 );
00461             }
00462
00463 #if CLOUDX1_HAS_CLOUDBUS
00467             void SetBus(const CBus& bus) {
00468                 assert(params);
00469                 cloudx1_SetInstanceBus(params,bus.Get());
00470             }
00471 #endif // CLOUDX1_HAS_CLOUDBUS
00472
00487         bool Create(const std::string& LoginKey, const std::string& RadioName, const std::string&
      AWS_Access_ID, const std::string& AWS_Access_Secret, const std::filesystem::path& save_path, int
      json_port = 0, unsigned int frames_per_chunk=64)
00488             {
00489                 assert(!instance);
00490                 std::string l_save_path;
00491                 if (!save_path.empty()) {
00492                     l_save_path = save_path.u8string();
00493                 }
00494                 instance = cloudx1_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                         cloudx1_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                     cloudx1_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 (cloudx1_WaitUpdateThreadReady(instance, timeout)<0) {
00512                     return false;
00513                 }
00514                 return true;
00515             }
00516
00524         unsigned int GetChunkFrames() {
00525             if (instance) {
00526                 return cloudx1_GetProcessChunkFrames(instance);
00527             }
00528             return 0;
00529         }
00530
00538         void PresetManager_InformChange(const std::filesystem::path& relative_path, PresetChange_Kind
    change_kind)
00539             {
00540                 cloudx1_PresetManager_InformChange(instance, relative_path.c_str(),
    (cloudx1_PresetChange_Kind) change_kind);
00541             }
00542
00554         void SetMetadata(const std::string& key, const char* value)
00555             {
00556                 if (instance) {
00557                     cloudx1_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                     cloudx1_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 = cloudx1_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                     cloudx1_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 CLOUDX1_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 = cloudx1_Webserver(http_port,https_port,instance);
00637                 if (webserver==SOUND4_INVALID_WEBSERVER_ID) {
00638                     return false;
00639                 }
00640                 int web_status=cloudx1_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                     cloudx1_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                 cloudx1_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                 cloudx1_Webserver_GetAppHealth(instance, &httpcode, &c_contenttype, &c_content);
00679                 contenttype=c_contenttype;
00680                 content=c_content;
00681                 cloudx1_Webserver_FreeString(c_contenttype);
00682                 cloudx1_Webserver_FreeString(c_content);
00683         }
00684 #endif // CLOUDX1_HAS_WEBSERVER
00685
00692         int Terminate()
00693             {
00694                 if (!instance) return 0;
00695                 return cloudx1_TerminateProcess(instance);
00696             }
00697
00703         void Stop()
00704             {
00705                 if (!instance) return;
00706 #if CLOUDX1_HAS_WEBSERVER
00707                 StopWebServer();
00708 #endif // CLOUDX1_HAS_WEBSERVER
00709                 if (update_thread.joinable()) {
00710                     cloudx1_StopUpdateThread(instance); // this will exit the update thread
00711                     update_thread.join();
00712                 }
00713                 cloudx1_ExitProcess(instance);
00714                 instance=nullptr;
00715             }
00716
00722         unsigned int GetEstimatedDelay()
00723             {
00724                 assert(instance);
00725                 return cloudx1_GetEstimatedDelay(instance);
00726             }
00727
00738         std::array<float,InputSampleSize>& GetBufferIn()
00739             {
00740                 assert(instance);
00741                 float *buf=cloudx1_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= cloudx1_GetBufferOut(instance);
00759                 return reinterpret_cast< std::array<float,OutputSampleSize>& >(*buf);
00760             }
00766         void ProcessAudio()
00767             {
00768                 assert(instance);
00769                 cloudx1_ProcessAudio(instance, cloudx1_GetBufferIn(instance),
      cloudx1_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                 cloudx1_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 = cloudx1_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(cloudx1_CInstance* instance)
00830                     {
00831                         assert(instance);
00832                         client=cloudx1_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                             cloudx1_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 = cloudx1_ProcessJson(client, request.c_str(),
      &need_save);
00857                         if (!canswer) return {};
00858                         std::string answer(canswer);
00859                         cloudx1_FreeJsonAnswer (canswer);
00860                         if (NeedSave) {
00861                             *NeedSave=(need_save!=0);
00862                         }
00863                         return answer;
00864                     }
00865             private:
00866                 cloudx1_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 cloudx1_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 cloudx1_GetAudio(instance, reinterpret_cast<T*>(payload), max_nFrame,
      helper::SampleFormat<T>::format);
00893             }
00894
00895     private:
00896         cloudx1_CParameters* params = nullptr;
00897         cloudx1_CInstance* instance = nullptr;
00898         std::thread update_thread;
00899         std::vector< std::pair<std::string,std::string> > init_metadata;
00900 #if CLOUDX1_HAS_WEBSERVER
00901         uint64_t webserver = SOUND4_INVALID_WEBSERVER_ID;
00902 #endif // CLOUDX1_HAS_WEBSERVER
00903     };
00904
00905 }; // namespace cloudx1
00906 }; // namespace sound4
00907
00908 // bridge C callbacks to CPresetLoader
00909 extern "C" {
00910     static char *cloudx1_custom_reader(const fs_char *filename, void* userdata) {
00911         sound4::cloudx1::CPresetLoader
      *preset_loader=reinterpret_cast<sound4::cloudx1::CPresetLoader*>(userdata);
00912         std::string content=preset_loader->Read(filename);
00913         return strdup(content.c_str());
00914     }
00915     static void cloudx1_custom_reader_free(char *content, void* userdata) {
00916         if (!content) return;
00917         free(content);
00918     }
00919     static int cloudx1_custom_writer(const fs_char *filename, const char *content, void* userdata) {
00920         sound4::cloudx1::CPresetLoader
      *preset_loader=reinterpret_cast<sound4::cloudx1::CPresetLoader*>(userdata);
00921         auto res=preset_loader->Write(filename,content);
00922         return res?0:-1;
00923     }
00924     static int cloudx1_custom_exists(const fs_char *filename, void* userdata) {
00925         sound4::cloudx1::CPresetLoader
      *preset_loader=reinterpret_cast<sound4::cloudx1::CPresetLoader*>(userdata);
00926         auto res=preset_loader->Exists(filename);
00927         return res?0:-1;
00928     }
00929     static fs_char** cloudx1_custom_getall(void* userdata) {
00930         sound4::cloudx1::CPresetLoader
      *preset_loader=reinterpret_cast<sound4::cloudx1::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 cloudx1_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 cloudx1_custom_remove(const fs_char *filename, void* userdata) {
00947         sound4::cloudx1::CPresetLoader
      *preset_loader=reinterpret_cast<sound4::cloudx1::CPresetLoader*>(userdata);
00948         auto res=preset_loader->Remove(filename);
00949         return res?0:-1;
00950     }
00951     static int cloudx1_custom_rename(const fs_char *from, const fs_char *to, void* userdata) {
00952         sound4::cloudx1::CPresetLoader
      *preset_loader=reinterpret_cast<sound4::cloudx1::CPresetLoader*>(userdata);
00953         auto res=preset_loader->Rename(from,to);
00954         return res?0:-1;
00955     }
00956
00957 };
```

11.11 sound4.x1.cloud_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.x1.cloud.h"
#include <stringapiset.h>
#include <wchar.h>
```

Classes

- class sound4::cloudx1::dyn::helper::DynFuncHolder< T >

  Helper to wrap a function pointer.
- class sound4::cloudx1::dyn::helper::CDynLoader

  Helper to load dynamic library.
- class sound4::cloudx1::dyn::CDynLib

  Dynamic library interface.
- struct sound4::cloudx1::dyn::helper::SampleFormat< T >

  Helper for Sample format types.
- struct sound4::cloudx1::dyn::helper::SampleFormat< int16_t >
- struct sound4::cloudx1::dyn::helper::SampleFormat< int32_t >
- struct sound4::cloudx1::dyn::helper::SampleFormat< float >
- class sound4::cloudx1::dyn::CPresetLoader
- class sound4::cloudx1::dyn::CInstance
- class sound4::cloudx1::dyn::CInstance::CClient

Namespaces

- sound4
- sound4::cloudx1
- dyn
- sound4::cloudx1::dyn
- sound4::cloudx1::dyn::helper

Macros

- #define fs_strdup wcsdup

  helper for OS dependant file name copy

Typedefs

- using sound4::cloudx1::dyn::helper::DynLib_t = HMODULE
- using sound4::cloudx1::dyn::helper::DynFunc_t = FARPROC
- using sound4::cloudx1::dyn::SampleFormat = cloudx1_SampleFormat
- using sound4::cloudx1::dyn::LogSeverity = cloudx1_LogSeverity
- using sound4::cloudx1::dyn::log_cb_t = std::function< void(LogSeverity, const std::string &)>
- using sound4::cloudx1::dyn::PresetChange_Kind = cloudx1_PresetChange_Kind

Variables

- const size_t sound4::cloudx1::dyn::ChunkFrames =CLOUDX1_AUDIOFRAME_COUNT
- const size_t sound4::cloudx1::dyn::InputChannels =CLOUDX1_CHANNEL_COUNT∗CLOUDX1_AUDIO_INPUT_COUNT
- const size_t sound4::cloudx1::dyn::OutputChannels =CLOUDX1_CHANNEL_COUNT∗CLOUDX1_AUDIO_OUTPUT_COUNT
- const size_t sound4::cloudx1::dyn::InputSampleSize =ChunkFrames∗InputChannels
- const size_t sound4::cloudx1::dyn::OutputSampleSize =ChunkFrames∗OutputChannels
- const size_t sound4::cloudx1::dyn::SampleRate =CLOUDX1_SAMPLE_RATE

11.11.1 Detailed Description

You need to load the dynamic library, and if it succeed you can use it.

Definition in file sound4.x1.cloud_dyn.hpp.

11.11.2 Macro Definition Documentation

11.11.2.1 fs_strdup

```
#define fs_strdup wcsdup
```

Definition at line 36 of file sound4.x1.cloud_dyn.hpp.

11.12 sound4.x1.cloud_dyn.hpp

```
00001
00007 #pragma once
00008
00009 #include <string>
00010 #include <exception>
00011 #include <thread>
00012 #include <functional>
00013 #include <memory>
00014 #include <array>
00015 #include <assert.h>
00016 #include <stdexcept>
00017 #include <filesystem>
00018 #include <string.h>
00019
00020 #ifdef __unix__
00021     #include <dlfcn.h>
00022 #elif defined(_WIN32)
00023     #include <libloaderapi.h>
00024 #else
00025     #error "Unsupported OS"
00026 #endif
00027
00028 // Needed to have function declarations
00029 #include "sound4.x1.cloud.h"
00030
00031
00032 #ifdef _WIN32
00033     #include <stringapiset.h>
00034     #include <wchar.h>
00036     #define fs_strdup wcsdup
00037 #else // !_WIN32
00039     #define fs_strdup strdup
00040 #endif // !_WIN32
00041
00042 // bridge C callbacks to CPresetLoader
00043 extern "C" {
```

```
00044     static char *cloudx1_custom_reader(const fs_char *filename, void* userdata);
00045     static void cloudx1_custom_reader_free(char *content, void* userdata);
00046     static int cloudx1_custom_writer(const fs_char *filename, const char *content, void* userdata);
00047     static int cloudx1_custom_exists(const fs_char *filename, void* userdata);
00048     static fs_char** cloudx1_custom_getall(void* userdata);
00049     static void cloudx1_custom_getall_free(fs_char** all, void* userdata);
00050     static int cloudx1_custom_remove(const fs_char *filename, void* userdata);
00051     static int cloudx1_custom_rename(const fs_char *from, const fs_char *to, void* userdata);
00052 };
00053
00057 namespace sound4 {
00061 namespace cloudx1 {
00065 namespace dyn {
00066
00070     namespace helper {
00071         // Storage definition, depending on OS
00072         #ifdef __unix__
00073             using DynLib_t = void *;
00074             using DynFunc_t = void *;
00075         #else
00076             using DynLib_t = HMODULE;
00077             using DynFunc_t = FARPROC;
00078         #endif
00079
00085             template <typename T>
00086         class DynFuncHolder {
00087         private:
00088             DynFunc_t m_ptr = NULL;
00089         public:
00090             DynFuncHolder() = default;
00091             DynFuncHolder(DynFunc_t a_ptr) : m_ptr(a_ptr) {}
00092             operator bool() const { return (m_ptr!=NULL); }
00093             bool IsOk() const { return m_ptr!=NULL; }
00094             operator T* () const { return reinterpret_cast<T*>(m_ptr); }
00095         };
00096
00101         class CDynLoader {
00102             private:
00103                 DynLib_t m_lib {};
00104             public:
00105                 CDynLoader() = default;
00106                 CDynLoader(CDynLoader&&) = default;
00107                 // No copy allowed
00108                 CDynLoader(const CDynLoader&) = delete;
00109                 CDynLoader& operator=(CDynLoader const&) = delete;
00110                 operator bool() const {
00111                     return (m_lib!=NULL);
00112                 }
00113                 bool IsOk() const {
00114                     return (m_lib!=NULL);
00115                 }
00116                 #ifdef __unix__
00117                     void Close() {
00118                         if (IsOk()) {
00119                             #ifndef __SANITIZE_ADDRESS__ // avoid unloading to keep call stack
00120                             dlclose(m_lib);
00121                             #endif // __SANITIZE_ADDRESS__
00122                             m_lib=NULL;
00123                         }
00124                     }
00125                     bool Load(const std::filesystem::path& dynlib)
00126                     {
00127                         Close();
00128                         m_lib=dlopen(dynlib.c_str(), RTLD_NOW|RTLD_LOCAL);
00129                         return IsOk();
00130                     }
00131                         template <typename T>
00132                     DynFuncHolder<T> GetSymbol(const std::string& name) {
00133                         auto ptr=dlsym(m_lib, name.c_str());
00134                         if (!ptr) {
00135                             throw std::runtime_error("Missing function in library");
00136                         }
00137                         return DynFuncHolder<T>(ptr);
00138                     }
00139                         template <typename T>
00140                     DynFuncHolder<T> GetSymbol_safe(const std::string& name) {
00141                         auto ptr=dlsym(m_lib, name.c_str());
00142                         return DynFuncHolder<T>(ptr);
00143                     }
00144                     static std::filesystem::path GetThisLibraryPath(void) {
00145                         static Dl_info info;
00146                         if (! dladdr((void *) GetThisLibraryPath, & info)) return {};
00147                         auto rp=realpath(info.dli_fname, NULL);
00148                         std::filesystem::path p(rp);
00149                         free(rp);
00150                         return p;
00151                     }
```

```
00152                 #else
00153                     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_
00154                     {
00155                         Close();
00156                         m_lib=LoadLibraryExW(dynlib.c_str(), NULL, loadflags);
00157                         return IsOk();
00158                     }
00159                     void Close() {
00160                         if (IsOk()) {
00161                             // NOTE: On Windows, FreeLibrary may create a lot of troubles, should use
    FreeLibraryAndExitThread but...
00162                             // So define SOUND4_CALL_FREELIBRARY before including this file if you
    really want to unload the library dynamically
00163                             #if defined(SOUND4_CALL_FREELIBRARYANDEXITTHREAD)
00164                                 FreeLibraryAndExitThread(m_lib, 0);
00165                             #elif defined(SOUND4_CALL_FREELIBRARY)
00166                                 FreeLibrary(m_lib);
00167                             #endif // SOUND4_CALL_FREELIBRARY
00168                             m_lib=NULL;
00169                         }
00170                     }
00171                         template <typename T>
00172                     DynFuncHolder<T> GetSymbol(const std::string& name) {
00173                         auto ptr=GetProcAddress(m_lib, name.c_str());
00174                         if (!ptr) {
00175                             throw std::runtime_error("Missing function in library");
00176                         }
00177                         return DynFuncHolder<T>(ptr);
00178                     }
00179                         template <typename T>
00180                     DynFuncHolder<T> GetSymbol_safe(const std::string& name) {
00181                         auto ptr=GetProcAddress(m_lib, name.c_str());
00182                         return DynFuncHolder<T>(ptr);
00183                     }
00184                     static std::filesystem::path GetThisLibraryPath(void) {
00185                         static wchar_t path[MAX_PATH]={};
00186                         HMODULE hm = NULL;
00187                         if (GetModuleHandleExW(GET_MODULE_HANDLE_EX_FLAG_FROM_ADDRESS |
00188                                 GET_MODULE_HANDLE_EX_FLAG_UNCHANGED_REFCOUNT,
00189                                 (LPCWSTR) &GetThisLibraryPath, &hm))
00190                         {
00191                             if (GetModuleFileNameW(hm, path, MAX_PATH) > 0) {
00192                                 return std::filesystem::path(path);
00193                             }
00194                         }
00195                         return {};
00196                     }
00197                 #endif
00198                 virtual ~CDynLoader() {
00199                     Close();
00200                 }
00201         }; // class CDynLoader
00202     }; // namespace helper
00203
00210     class CDynLib {
00211     private:
00212         helper::CDynLoader m_lib;
00213         bool m_bOK = false;
00214     public:
00215         CDynLib() = default;
00216
00217         // All those wrapped functions have the same signature as there C cloudx1_XXX
00218         helper::DynFuncHolder< decltype(cloudx1_GetVersion             ) > GetVersion             ;
00219         helper::DynFuncHolder< decltype(cloudx1_GetChunkSizeInFrames   ) > GetChunkSizeInFrames   ;
00220         helper::DynFuncHolder< decltype(cloudx1_GetChannelCount        ) > GetChannelCount        ;
00221         helper::DynFuncHolder< decltype(cloudx1_GetAudioInputCount     ) > GetAudioInputCount     ;
00222         helper::DynFuncHolder< decltype(cloudx1_GetAudioOutputCount    ) > GetAudioOutputCount    ;
00223         helper::DynFuncHolder< decltype(cloudx1_GetSampleRate          ) > GetSampleRate          ;
00224         helper::DynFuncHolder< decltype(cloudx1_SetLoggerCallback      ) > SetLoggerCallback      ;
00225         helper::DynFuncHolder< decltype(cloudx1_SetLogSeverity         ) > SetLogSeverity         ;
00226         helper::DynFuncHolder< decltype(cloudx1_NewParameters          ) > NewParameters          ;
00227         helper::DynFuncHolder< decltype(cloudx1_FreeParameters         ) > FreeParameters         ;
00228         helper::DynFuncHolder< decltype(cloudx1_SetParameter           ) > SetParameter           ;
00229         helper::DynFuncHolder< decltype(cloudx1_GetParameter           ) > GetParameter           ;
00230         helper::DynFuncHolder< decltype(cloudx1_FreeParameterValue     ) > FreeParameterValue     ;
00231         helper::DynFuncHolder< decltype(cloudx1_InitProcess            ) > InitProcess            ;
00232         helper::DynFuncHolder< decltype(cloudx1_InitProcess2           ) > InitProcess2           ;
00233         helper::DynFuncHolder< decltype(cloudx1_TerminateProcess       ) > TerminateProcess       ;
00234         helper::DynFuncHolder< decltype(cloudx1_ExitProcess            ) > ExitProcess            ;
00235         helper::DynFuncHolder< decltype(cloudx1_StartUpdateThread      ) > StartUpdateThread      ;
00236         helper::DynFuncHolder< decltype(cloudx1_StopUpdateThread       ) > StopUpdateThread       ;
00237         helper::DynFuncHolder< decltype(cloudx1_WaitUpdateThreadReady  ) > WaitUpdateThreadReady  ;
00238         helper::DynFuncHolder< decltype(cloudx1_ProcessAudio           ) > ProcessAudio           ;
00239         helper::DynFuncHolder< decltype(cloudx1_ProcessAudio_Planar    ) > ProcessAudio_Planar    ;
00240         helper::DynFuncHolder< decltype(cloudx1_GetBufferIn            ) > GetBufferIn            ;
00241         helper::DynFuncHolder< decltype(cloudx1_GetBufferOut           ) > GetBufferOut           ;
```

```
00242         helper::DynFuncHolder< decltype(cloudx1_GetEstimatedDelay      ) > GetEstimatedDelay      ;
00243         helper::DynFuncHolder< decltype(cloudx1_GetFormatName          ) > GetFormatName          ;
00244         helper::DynFuncHolder< decltype(cloudx1_GetFormatFromName      ) > GetFormatFromName      ;
00245         helper::DynFuncHolder< decltype(cloudx1_GetBytesFromFormat     ) > GetBytesFromFormat     ;
00246         helper::DynFuncHolder< decltype(cloudx1_GetMaxPacketFrame      ) > GetMaxPacketFrame      ;
00247         helper::DynFuncHolder< decltype(cloudx1_AddAudio               ) > AddAudio               ;
00248         helper::DynFuncHolder< decltype(cloudx1_AddPadAudio            ) > AddPadAudio            ;
00249         helper::DynFuncHolder< decltype(cloudx1_GetOutputCount         ) > GetOutputCount         ;
00250         helper::DynFuncHolder< decltype(cloudx1_GetAudio               ) > GetAudio               ;
00251         helper::DynFuncHolder< decltype(cloudx1_AudioConvertFrom       ) > AudioConvertFrom       ;
00252         helper::DynFuncHolder< decltype(cloudx1_AudioConvertTo         ) > AudioConvertTo         ;
00253         helper::DynFuncHolder< decltype(cloudx1_StereoToMono           ) > StereoToMono           ;
00254         helper::DynFuncHolder< decltype(cloudx1_MonoToStereo           ) > MonoToStereo           ;
00255         helper::DynFuncHolder< decltype(cloudx1_AudioMonoFromLiveStereo) > AudioMonoFromLiveStereo;
00256         helper::DynFuncHolder< decltype(cloudx1_AudioMonoToLiveStereo  ) > AudioMonoToLiveStereo  ;
00257         helper::DynFuncHolder< decltype(cloudx1_NewClient              ) > NewClient              ;
00258         helper::DynFuncHolder< decltype(cloudx1_DeleteClient           ) > DeleteClient           ;
00259         helper::DynFuncHolder< decltype(cloudx1_ProcessJson            ) > ProcessJson            ;
00260         helper::DynFuncHolder< decltype(cloudx1_FreeJsonAnswer         ) > FreeJsonAnswer         ;
00261         helper::DynFuncHolder< decltype(cloudx1_SaveState              ) > SaveState              ;
00262 #if CLOUDX1_HAS_WEBSERVER
00263         helper::DynFuncHolder< decltype(cloudx1_Webserver_tcp          ) > Webserver_tcp          ;
00264         helper::DynFuncHolder< decltype(cloudx1_Webserver_tcp2         ) > Webserver_tcp2         ;
00265         helper::DynFuncHolder< decltype(cloudx1_Webserver              ) > Webserver              ;
00266         helper::DynFuncHolder< decltype(cloudx1_Webserver_Stop         ) > Webserver_Stop         ;
00267         helper::DynFuncHolder< decltype(cloudx1_Webserver_Status       ) > Webserver_Status       ;
00268 #endif // CLOUDX1_HAS_WEBSERVER
00269
00270         // added 2023-03-22
00271         helper::DynFuncHolder< decltype(cloudx1_StereoToMono_Planar    ) > StereoToMono_Planar    ;
00272         helper::DynFuncHolder< decltype(cloudx1_MonoToStereo_Planar    ) > MonoToStereo_Planar    ;
00273
00274         // added 2023-05-22
00275         helper::DynFuncHolder< decltype(cloudx1_SetMetadata            ) > SetMetadata            ;
00276         helper::DynFuncHolder< decltype(cloudx1_GetMetadataInfos       ) > GetMetadataInfos       ;
00277         helper::DynFuncHolder< decltype(cloudx1_FreeMetadataInfos      ) > FreeMetadataInfos      ;
00278
00279         // added 2023-06-20
00280         helper::DynFuncHolder< decltype(cloudx1_SetPresetManager               ) > SetPresetManager
     ;
00281         helper::DynFuncHolder< decltype(cloudx1_PresetManager_InformChange     ) >
    PresetManager_InformChange      ;
00282
00283         // added 2023-09-21
00284         helper::DynFuncHolder< decltype(cloudx1_GetPossibleChunkSizeInFrames   ) >
    GetPossibleChunkSizeInFrames   ;
00285         helper::DynFuncHolder< decltype(cloudx1_GetProcessChunkFrames          ) >
    GetProcessChunkFrames    ;
00286         helper::DynFuncHolder< decltype(cloudx1_InitProcess3                   ) > InitProcess3
     ;
00287
00288 #if CLOUDX1_HAS_CLOUDBUS
00289         // added 2023-06-26
00290         helper::DynFuncHolder< decltype(cloudx1_NewBus                 ) > NewBus      ;
00291         helper::DynFuncHolder< decltype(cloudx1_FreeBus                ) > FreeBus      ;
00292         helper::DynFuncHolder< decltype(cloudx1_SetInstanceBus         ) > SetInstanceBus      ;
00293 #endif // CLOUDX1_HAS_CLOUDBUS
00294
00295         // added 2024-02-19
00296         helper::DynFuncHolder< decltype(cloudx1_SetMetadataMulti       ) > SetMetadataMulti        ;
00297
00298 #if CLOUDX1_HAS_WEBSERVER
00299         // added 2024-05-23
00300         helper::DynFuncHolder< decltype(cloudx1_Webserver_SetAppHealth    ) > Webserver_SetAppHealth
     ;
00301         helper::DynFuncHolder< decltype(cloudx1_Webserver_GetAppHealth    ) > Webserver_GetAppHealth
     ;
00302         helper::DynFuncHolder< decltype(cloudx1_Webserver_FreeString      ) > Webserver_FreeString
     ;
00303 #endif // CLOUDX1_HAS_WEBSERVER
00304
00312         bool Load(const std::filesystem::path& path = {}) {
00313             std::string filename;
00314             #ifdef __unix__
00315                 filename="libsound4.x1.cloud.so";
00316             #else
00317                 filename="sound4.x1.cloud.dll";
00318             #endif
00319             m_lib.Close();
00320             // If a path is given, use it directly and do not try other path
00321             if (!path.empty() && !m_lib.Load(path / filename)) {
00322                 return false;
00323             } else if (!m_lib.IsOk()) {
00324                 auto thisdir=helper::CDynLoader::GetThisLibraryPath().parent_path();
00325                 #ifndef _WIN32
00326                 // Linux: if in a bin directory, try ../lib first
00327                 if (!m_lib.IsOk() && thisdir.filename()=="bin") {
```

```
00328                     auto libdir = thisdir.parent_path() / "lib";
00329                     m_lib.Load(libdir / filename);
00330                 }
00331                 #endif // !_WIN32
00332                 if (!m_lib.IsOk()) {
00333                     // Search in the same directory this code is in
00334                     m_lib.Load(thisdir / filename);
00335                 }
00336                 if (!m_lib.IsOk()) {
00337                     // Try current path
00338                     m_lib.Load(std::filesystem::current_path() / filename);
00339                 }
00340                 if (!m_lib.IsOk()) {
00341                     return false;
00342                 }
00343             }
00344             // Load all C functions
00345             try {
00346                 GetVersion              = m_lib.GetSymbol< decltype(cloudx1_GetVersion             )
    >("cloudx1_GetVersion"             );
00347                 GetChunkSizeInFrames    = m_lib.GetSymbol< decltype(cloudx1_GetChunkSizeInFrames   )
    >("cloudx1_GetChunkSizeInFrames"   );
00348                 GetChannelCount         = m_lib.GetSymbol< decltype(cloudx1_GetChannelCount        )
    >("cloudx1_GetChannelCount"        );
00349                 GetAudioInputCount      = m_lib.GetSymbol< decltype(cloudx1_GetAudioInputCount     )
    >("cloudx1_GetAudioInputCount"     );
00350                 GetAudioOutputCount     = m_lib.GetSymbol< decltype(cloudx1_GetAudioOutputCount    )
    >("cloudx1_GetAudioOutputCount"    );
00351                 GetSampleRate           = m_lib.GetSymbol< decltype(cloudx1_GetSampleRate          )
    >("cloudx1_GetSampleRate"          );
00352                 SetLoggerCallback       = m_lib.GetSymbol< decltype(cloudx1_SetLoggerCallback      )
    >("cloudx1_SetLoggerCallback"      );
00353                 SetLogSeverity          = m_lib.GetSymbol< decltype(cloudx1_SetLogSeverity         )
    >("cloudx1_SetLogSeverity"         );
00354                 NewParameters           = m_lib.GetSymbol< decltype(cloudx1_NewParameters          )
    >("cloudx1_NewParameters"          );
00355                 FreeParameters          = m_lib.GetSymbol< decltype(cloudx1_FreeParameters         )
    >("cloudx1_FreeParameters"         );
00356                 SetParameter            = m_lib.GetSymbol< decltype(cloudx1_SetParameter           )
    >("cloudx1_SetParameter"           );
00357                 GetParameter            = m_lib.GetSymbol< decltype(cloudx1_GetParameter           )
    >("cloudx1_GetParameter"           );
00358                 FreeParameterValue      = m_lib.GetSymbol< decltype(cloudx1_FreeParameterValue     )
    >("cloudx1_FreeParameterValue"     );
00359                 InitProcess             = m_lib.GetSymbol< decltype(cloudx1_InitProcess            )
    >("cloudx1_InitProcess"            );
00360                 InitProcess2            = m_lib.GetSymbol< decltype(cloudx1_InitProcess2           )
    >("cloudx1_InitProcess2"           );
00361                 TerminateProcess        = m_lib.GetSymbol< decltype(cloudx1_TerminateProcess       )
    >("cloudx1_TerminateProcess"       );
00362                 ExitProcess             = m_lib.GetSymbol< decltype(cloudx1_ExitProcess            )
    >("cloudx1_ExitProcess"            );
00363                 StartUpdateThread       = m_lib.GetSymbol< decltype(cloudx1_StartUpdateThread      )
    >("cloudx1_StartUpdateThread"      );
00364                 StopUpdateThread        = m_lib.GetSymbol< decltype(cloudx1_StopUpdateThread       )
    >("cloudx1_StopUpdateThread"       );
00365                 WaitUpdateThreadReady   = m_lib.GetSymbol< decltype(cloudx1_WaitUpdateThreadReady  )
    >("cloudx1_WaitUpdateThreadReady"  );
00366                 ProcessAudio            = m_lib.GetSymbol< decltype(cloudx1_ProcessAudio           )
    >("cloudx1_ProcessAudio"           );
00367                 ProcessAudio_Planar     = m_lib.GetSymbol< decltype(cloudx1_ProcessAudio_Planar    )
    >("cloudx1_ProcessAudio_Planar"    );
00368                 GetBufferIn             = m_lib.GetSymbol< decltype(cloudx1_GetBufferIn            )
    >("cloudx1_GetBufferIn"            );
00369                 GetBufferOut            = m_lib.GetSymbol< decltype(cloudx1_GetBufferOut           )
    >("cloudx1_GetBufferOut"           );
00370                 GetEstimatedDelay       = m_lib.GetSymbol< decltype(cloudx1_GetEstimatedDelay      )
    >("cloudx1_GetEstimatedDelay"      );
00371                 GetFormatName           = m_lib.GetSymbol< decltype(cloudx1_GetFormatName          )
    >("cloudx1_GetFormatName"          );
00372                 GetFormatFromName       = m_lib.GetSymbol< decltype(cloudx1_GetFormatFromName      )
    >("cloudx1_GetFormatFromName"      );
00373                 GetBytesFromFormat      = m_lib.GetSymbol< decltype(cloudx1_GetBytesFromFormat     )
    >("cloudx1_GetBytesFromFormat"     );
00374                 GetMaxPacketFrame       = m_lib.GetSymbol< decltype(cloudx1_GetMaxPacketFrame      )
    >("cloudx1_GetMaxPacketFrame"      );
00375                 AddAudio                = m_lib.GetSymbol< decltype(cloudx1_AddAudio               )
    >("cloudx1_AddAudio"               );
00376                 AddPadAudio             = m_lib.GetSymbol< decltype(cloudx1_AddPadAudio            )
    >("cloudx1_AddPadAudio"            );
00377                 GetOutputCount          = m_lib.GetSymbol< decltype(cloudx1_GetOutputCount         )
    >("cloudx1_GetOutputCount"         );
00378                 GetAudio                = m_lib.GetSymbol< decltype(cloudx1_GetAudio               )
    >("cloudx1_GetAudio"               );
00379                 AudioConvertFrom        = m_lib.GetSymbol< decltype(cloudx1_AudioConvertFrom       )
    >("cloudx1_AudioConvertFrom"       );
00380                 AudioConvertTo          = m_lib.GetSymbol< decltype(cloudx1_AudioConvertTo         )
```

```
      >("cloudx1_AudioConvertTo"         );
00381                 StereoToMono            = m_lib.GetSymbol< decltype(cloudx1_StereoToMono           )
      >("cloudx1_StereoToMono"           );
00382                 MonoToStereo            = m_lib.GetSymbol< decltype(cloudx1_MonoToStereo           )
      >("cloudx1_MonoToStereo"           );
00383                 AudioMonoFromLiveStereo = m_lib.GetSymbol< decltype(cloudx1_AudioMonoFromLiveStereo)
      >("cloudx1_AudioMonoFromLiveStereo");
00384                 AudioMonoToLiveStereo   = m_lib.GetSymbol< decltype(cloudx1_AudioMonoToLiveStereo  )
      >("cloudx1_AudioMonoToLiveStereo"  );
00385                 NewClient               = m_lib.GetSymbol< decltype(cloudx1_NewClient              )
      >("cloudx1_NewClient"              );
00386                 DeleteClient            = m_lib.GetSymbol< decltype(cloudx1_DeleteClient           )
      >("cloudx1_DeleteClient"           );
00387                 ProcessJson             = m_lib.GetSymbol< decltype(cloudx1_ProcessJson            )
      >("cloudx1_ProcessJson"            );
00388                 FreeJsonAnswer          = m_lib.GetSymbol< decltype(cloudx1_FreeJsonAnswer         )
      >("cloudx1_FreeJsonAnswer"         );
00389                 SaveState               = m_lib.GetSymbol< decltype(cloudx1_SaveState              )
      >("cloudx1_SaveState"              );
00390 #if CLOUDX1_HAS_WEBSERVER
00391                 Webserver_tcp           = m_lib.GetSymbol< decltype(cloudx1_Webserver_tcp          )
      >("cloudx1_Webserver_tcp"          );
00392                 Webserver_tcp2          = m_lib.GetSymbol< decltype(cloudx1_Webserver_tcp2         )
      >("cloudx1_Webserver_tcp2"         );
00393                 Webserver               = m_lib.GetSymbol< decltype(cloudx1_Webserver              )
      >("cloudx1_Webserver"              );
00394                 Webserver_Stop          = m_lib.GetSymbol< decltype(cloudx1_Webserver_Stop         )
      >("cloudx1_Webserver_Stop"         );
00395                 Webserver_Status        = m_lib.GetSymbol< decltype(cloudx1_Webserver_Status       )
      >("cloudx1_Webserver_Status"       );
00396 #endif // CLOUDX1_HAS_WEBSERVER
00397
00398                 // added 2023-03-22
00399                 StereoToMono_Planar     = m_lib.GetSymbol< decltype(cloudx1_StereoToMono_Planar    )
      >("cloudx1_StereoToMono_Planar"    );
00400                 MonoToStereo_Planar     = m_lib.GetSymbol< decltype(cloudx1_MonoToStereo_Planar    )
      >("cloudx1_MonoToStereo_Planar"    );
00401
00402             } catch (std::runtime_error& ) {
00403                 return false;
00404             }
00405             // C functions allowed to be missing
00406             try {
00407                 // added 2023-05-22
00408                 SetMetadata             = m_lib.GetSymbol< decltype(cloudx1_SetMetadata            )
      >("cloudx1_SetMetadata"            );
00409                 GetMetadataInfos        = m_lib.GetSymbol< decltype(cloudx1_GetMetadataInfos       )
      >("cloudx1_GetMetadataInfos"       );
00410                 FreeMetadataInfos       = m_lib.GetSymbol< decltype(cloudx1_FreeMetadataInfos      )
      >("cloudx1_FreeMetadataInfos"      );
00411                 // added 2023-06-20
00412                 SetPresetManager            = m_lib.GetSymbol< decltype(cloudx1_SetPresetManager
          ) >("cloudx1_SetPresetManager"           );
00413                 PresetManager_InformChange  = m_lib.GetSymbol<
      decltype(cloudx1_PresetManager_InformChange     ) >("cloudx1_PresetManager_InformChange"    );
00414
00415                 // added 2023-09-21
00416                 GetPossibleChunkSizeInFrames = m_lib.GetSymbol<
      decltype(cloudx1_GetPossibleChunkSizeInFrames  ) >("cloudx1_GetPossibleChunkSizeInFrames"  );
00417                 GetProcessChunkFrames        = m_lib.GetSymbol< decltype(cloudx1_GetProcessChunkFrames
          ) >("cloudx1_GetProcessChunkFrames"         );
00418                 InitProcess3                 = m_lib.GetSymbol< decltype(cloudx1_InitProcess3
          ) >("cloudx1_InitProcess3"                  );
00419
00420 #if CLOUDX1_HAS_CLOUDBUS
00421                 // added 2023-06-26
00422                 NewBus                  = m_lib.GetSymbol< decltype(cloudx1_NewBus                 )
      >("cloudx1_NewBus"                 );
00423                 FreeBus                 = m_lib.GetSymbol< decltype(cloudx1_FreeBus                )
      >("cloudx1_FreeBus"                );
00424                 SetInstanceBus          = m_lib.GetSymbol< decltype(cloudx1_SetInstanceBus         )
      >("cloudx1_SetInstanceBus"         );
00425 #endif // CLOUDX1_HAS_CLOUDBUS
00426
00427                 // added 2024-02-19
00428                 SetMetadataMulti        = m_lib.GetSymbol< decltype(cloudx1_SetMetadataMulti       )
      >("cloudx1_SetMetadataMulti"       );
00429
00430 #if CLOUDX1_HAS_WEBSERVER
00431                 // added 2024-05-23
00432                 Webserver_SetAppHealth      = m_lib.GetSymbol< decltype(cloudx1_Webserver_SetAppHealth
        ) >("Webserver_SetAppHealth"     );
00433                 Webserver_GetAppHealth      = m_lib.GetSymbol< decltype(cloudx1_Webserver_GetAppHealth
        ) >("Webserver_GetAppHealth"     );
00434                 Webserver_FreeString        = m_lib.GetSymbol< decltype(cloudx1_Webserver_FreeString
        ) >("Webserver_FreeString"       );
00435 #endif // CLOUDX1_HAS_WEBSERVER
```

```
00436
00437             }  catch (std::runtime_error& ) {
00438                 // Ignored, handler will take care
00439             }
00440
00441             m_bOK=true;
00442             return true;
00443         }
00450         bool IsOk() const {
00451             return m_bOK;
00452         }
00453
00454     };
00455
00456
00457
00458     static constexpr const char *process_name = "SOUND4 x1.CLOUD";
00459     static constexpr const char *process_shortname = "x1";
00460     using SampleFormat = cloudx1_SampleFormat;
00464     const size_t ChunkFrames=CLOUDX1_AUDIOFRAME_COUNT;
00468     const size_t InputChannels=CLOUDX1_CHANNEL_COUNT*CLOUDX1_AUDIO_INPUT_COUNT;
00472     const size_t OutputChannels=CLOUDX1_CHANNEL_COUNT*CLOUDX1_AUDIO_OUTPUT_COUNT;
00476     const size_t InputSampleSize=ChunkFrames*InputChannels;
00480     const size_t OutputSampleSize=ChunkFrames*OutputChannels;
00484     const size_t SampleRate=CLOUDX1_SAMPLE_RATE;
00485
00489     namespace helper {
00490         #ifdef _WIN32
00496         static inline std::string WStringToUTF8(const std::wstring& wstr) {
00497             if (wstr.empty()) return {};
00498             int size = WideCharToMultiByte(CP_UTF8, 0, &wstr[0], (int)wstr.size(), NULL, 0, NULL,
    NULL);
00499             std::string str(size, 0);
00500             WideCharToMultiByte(CP_UTF8, 0, &wstr[0], (int)wstr.size(), &str[0], size, NULL, NULL);
00501             return str;
00502         }
00508         static inline std::wstring UTF8ToWString(const std::string& str) {
00509             if (str.empty()) return std::wstring();
00510             int size = MultiByteToWideChar(CP_UTF8, 0, &str[0], (int)str.size(), NULL, 0);
00511             std::wstring wstr(size, 0);
00512             MultiByteToWideChar(CP_UTF8, 0, &str[0], (int)str.size(), &wstr[0], size);
00513             return wstr;
00514         }
00515         #endif // _WIN32
00516
00522             template<typename T>
00523         struct SampleFormat {
00524         };
00525             template<>
00526         struct SampleFormat<int16_t> {
00527             const cloudx1_SampleFormat format = S16_NATIVE;
00528         };
00529             template<>
00530         struct SampleFormat<int32_t> {
00531             const cloudx1_SampleFormat format = S32_NATIVE;
00532         };
00533             template<>
00534         struct SampleFormat<float> {
00535             const cloudx1_SampleFormat format = F32_NATIVE;
00536         };
00537
00538
00543         static inline void AudioConvertFrom(CDynLib& dynlib, const uint8_t *payload, float *output,
    size_t nSpl, cloudx1_SampleFormat fmt) {
00544             dynlib.AudioConvertFrom(payload, output, nSpl, fmt);
00545         }
00546         static inline void AudioConvertTo(CDynLib& dynlib, const float *input, uint8_t *payload,
    size_t nSpl, cloudx1_SampleFormat fmt) {
00547             dynlib.AudioConvertTo(input, payload, nSpl, fmt);
00548         }
00549         static inline void StereoToMono(CDynLib& dynlib, const float *input, float *output, size_t
    nSpl) {
00550             dynlib.StereoToMono(input, output, nSpl);
00551         }
00552         static inline void MonoToStereo(CDynLib& dynlib, const float *input, float *output, size_t
    nSpl) {
00553             dynlib.MonoToStereo(input, output, nSpl);
00554         }
00555         static inline void StereoToMono_Planar(CDynLib& dynlib, const float *inputL, const float
    *inputR, float *output, size_t nSpl) {
00556             dynlib.StereoToMono_Planar(inputL, inputR, output, nSpl);
00557         }
00558         static inline void MonoToStereo_Planar(CDynLib& dynlib, const float *input, float *outputL,
    float *outputR, size_t nSpl) {
00559             dynlib.MonoToStereo_Planar(input, outputL, outputR, nSpl);
00560         }
00561         static inline void AudioMonoFromLiveStereo(CDynLib& dynlib, const uint8_t *payload, float
```

```
    *output) {
00562             dynlib.AudioMonoFromLiveStereo(payload, output);
00563         }
00564         static inline void AudioMonoToLiveStereo(CDynLib& dynlib, const float *input, uint8_t
    *payload) {
00565             dynlib.AudioMonoToLiveStereo(input, payload);
00566         }
00567
00568     }; // namespace helper
00569
00575     static inline std::string GetVersion(CDynLib& dynlib)
00576         {
00577             return std::string(dynlib.GetVersion());
00578         }
00579
00590     static inline unsigned int GetChunkSizeInFrames(CDynLib& dynlib)
00591         {
00592             return dynlib.GetChunkSizeInFrames();
00593         }
00594
00603     static inline std::vector<unsigned int> GetPossibleChunkSizeInFrames(CDynLib& dynlib)
00604         {
00605             std::vector<unsigned int> list;
00606             if (dynlib.GetPossibleChunkSizeInFrames) {
00607                 for (unsigned int* src=dynlib.GetPossibleChunkSizeInFrames();*src;src++) {
00608                     list.push_back(*src);
00609                 }
00610             } else {
00611                 list.push_back(64);
00612             }
00613             return list;
00614         }
00615
00623     static inline unsigned int GetChannelCount(CDynLib& dynlib)
00624         {
00625             return dynlib.GetChannelCount();
00626         }
00627
00636     static inline unsigned int GetAudioInputCount(CDynLib& dynlib)
00637         {
00638             return dynlib.GetAudioInputCount();
00639         }
00640
00649     static inline unsigned int GetAudioOutputCount(CDynLib& dynlib)
00650         {
00651             return dynlib.GetAudioOutputCount();
00652         }
00653
00661     static inline unsigned int GetSampleRate(CDynLib& dynlib)
00662         {
00663             return dynlib.GetSampleRate();
00664         }
00665
00671     static inline void SanityCheck(CDynLib& dynlib, bool a_bCheckFrames = true)
00672     {
00673         // Do some sanity checks
00674         if (dynlib.GetSampleRate()!=CLOUDX1_SAMPLE_RATE) {
00675             throw std::runtime_error("Bad library sampling rate");
00676         }
00677         if (a_bCheckFrames && dynlib.GetChunkSizeInFrames()!=ChunkFrames) {
00678             throw std::runtime_error("Bad library frame size");
00679         }
00680         if (dynlib.GetChannelCount()!=CLOUDX1_CHANNEL_COUNT) {
00681             throw std::runtime_error("Bad library channel count");
00682         }
00683         if (dynlib.GetAudioInputCount()!=CLOUDX1_AUDIO_INPUT_COUNT) {
00684             throw std::runtime_error("Bad library input count");
00685         }
00686         if (dynlib.GetAudioOutputCount()!=CLOUDX1_AUDIO_OUTPUT_COUNT) {
00687             throw std::runtime_error("Bad library output count");
00688         }
00689     }
00690
00697     static inline std::string GetFormatName(CDynLib& dynlib, const SampleFormat fmt)
00698         {
00699             return std::string(dynlib.GetFormatName(fmt));
00700         }
00701
00708     static inline SampleFormat GetFormatFromName(CDynLib& dynlib, const std::string& name)
00709         {
00710             return dynlib.GetFormatFromName(name.c_str());
00711         }
00712
00719     static inline unsigned int GetBytesFromFormat(CDynLib& dynlib, const SampleFormat fmt)
00720         {
00721             return dynlib.GetBytesFromFormat(fmt);
00722         }
```

```
00723
00727     using LogSeverity = cloudx1_LogSeverity;
00731     using log_cb_t = std::function<void(LogSeverity,const std::string&)>;
00732
00736     static log_cb_t _log_cb;
00737     extern "C" {
00738         static inline void _log_cb_c(cloudx1_LogSeverity severity, const char *c_msg) {
00739             _log_cb(severity, std::string(c_msg));
00740         }
00741     }
00745     static inline void SetLogSeverity(CDynLib& dynlib, LogSeverity severity)
00746         {
00747             dynlib.SetLogSeverity(severity);
00748         }
00752     static inline void SetLoggerCallback(CDynLib& dynlib, log_cb_t cb)
00753         {
00754             _log_cb=cb;
00755             dynlib.SetLoggerCallback(&_log_cb_c);
00756         }
00757
00758 #if CLOUDX1_HAS_CLOUDBUS
00759     class CBus {
00760         public:
00761             CBus(CDynLib& dynlib)
00762                 : m_dynlib(dynlib)
00763                 {
00764                     if (m_dynlib.NewBus) {
00765                         m_bus=m_dynlib.NewBus();
00766                     }
00767                 }
00768             ~CBus()
00769                 {
00770                     if (m_bus && m_dynlib.FreeBus) {
00771                         m_dynlib.FreeBus(m_bus);
00772                     }
00773                     m_bus = nullptr;
00774                 }
00775             cloudx1_CBus *Get() const { return m_bus; }
00776         private:
00777             CDynLib& m_dynlib;
00778             cloudx1_CBus *m_bus;
00779     };
00780 #endif // CLOUDX1_HAS_CLOUDBUS
00781
00786     class CPresetLoader {
00787     public:
00788         CPresetLoader() = default;
00789         virtual ~CPresetLoader() = default;
00790
00791         virtual bool IsReadOnly() = 0;
00792         virtual bool Exists(const std::filesystem::path &name) = 0;
00793         virtual bool Remove(const std::filesystem::path &name) = 0;
00794         virtual bool Rename(const std::filesystem::path &from, const std::filesystem::path &to) = 0;
00795         virtual std::vector<std::filesystem::path> GetAll() = 0;
00796         virtual std::string Read(const std::filesystem::path &filename) = 0;
00797         virtual bool Write(const std::filesystem::path &filename, const std::string &content) =0;
00798     };
00799     using PresetChange_Kind = cloudx1_PresetChange_Kind;
00800
00804     class CInstance {
00805     public:
00809         CInstance(CDynLib& dynlib)
00810         : m_dynlib(dynlib)
00811             {
00812                 params = m_dynlib.NewParameters();
00813                 if (!params) throw std::bad_alloc();
00814             }
00815         // non-copyable
00816         CInstance( const CInstance& ) = delete;
00817         CInstance& operator=( const CInstance& ) = delete;
00821         ~CInstance()
00822             {
00823                 Stop();
00824                 if (params) m_dynlib.FreeParameters(params);
00825             }
00829        bool IsOk() const { return instance!=nullptr; }
00837         void SetParam(const std::string& name, const std::string& value)
00838             {
00839                 assert(params);
00840                 m_dynlib.SetParameter(params, name.c_str(), value.c_str());
00841             }
00847         std::string GetParam(const std::string& name)
00848             {
00849                 assert(params);
00850                 auto c_value=m_dynlib.GetParameter(params, name.c_str());
00851                 std::string ret(c_value);
00852                 m_dynlib.FreeParameterValue(c_value);
```

```
00853                 return ret;
00854             }
00855 #if CLOUDX1_HAS_CLOUDBUS
00859             void SetBus(const CBus& bus) {
00860                 assert(params);
00861                 if (m_dynlib.SetInstanceBus) {
00862                     m_dynlib.SetInstanceBus(params,bus.Get());
00863                 }
00864             }
00865 #endif // CLOUDX1_HAS_CLOUDBUS
00866
00876         void SetPresetManager(CPresetLoader *preset_manager)
00877             {
00878                 if (!preset_manager) return;
00879                 if (!m_dynlib.SetPresetManager) return;
00880                 assert(params);
00881                 m_dynlib.SetPresetManager(params,
00882                     cloudx1_custom_reader,
00883                     cloudx1_custom_reader_free,
00884                     cloudx1_custom_writer,
00885                     cloudx1_custom_exists,
00886                     cloudx1_custom_getall,
00887                     cloudx1_custom_getall_free,
00888                     cloudx1_custom_remove,
00889                     cloudx1_custom_rename,
00890                     preset_manager->IsReadOnly(),
00891                     preset_manager
00892                 );
00893             }
00894
00909         bool Create(const std::string& LoginKey, const std::string& RadioName, const std::string&
      Access_Key_ID, const std::string& Access_Key_Secret, const std::filesystem::path& save_path, int
      json_port = 0, unsigned int frames_per_chunk=64)
00910             {
00911                 assert(!instance);
00912                 std::string l_save_path;
00913                 if (!save_path.empty()) {
00914                     l_save_path = save_path.u8string();
00915                 }
00916                 if (m_dynlib.InitProcess3) {
00917                     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);
00918                 } else if (frames_per_chunk!=64) {
00919                     return false;
00920                 } else {
00921                     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);
00922                 }
00923                 if (!instance) return false;
00924                 if (!init_metadata.empty()) {
00925                     for (auto&& [key,value]: init_metadata) {
00926                         m_dynlib.SetMetadata(instance, key.c_str(), value.c_str());
00927                     }
00928                     init_metadata.clear();
00929                 }
00930                 update_thread = std::thread([this, json_port](){
00931                     OnUpdateThreadStart();
00932                     m_dynlib.StartUpdateThread(instance, json_port);
00933                     OnUpdateThreadStop();
00934                 });
00935                 int timeout=5000;
00936                 #ifdef DEBUG
00937                     timeout=60*1000; // For debugger pauses
00938                 #endif
00939                 if (m_dynlib.WaitUpdateThreadReady(instance, timeout)<0) {
00940                     return false;
00941                 }
00942                 return true;
00943             }
00944
00952         unsigned int GetChunkFrames() {
00953             if (instance && m_dynlib.GetProcessChunkFrames) {
00954                 return m_dynlib.GetProcessChunkFrames(instance);
00955             }
00956             return 64;
00957         }
00958
00966         void PresetManager_InformChange(const std::filesystem::path& relative_path, PresetChange_Kind
      change_kind)
00967             {
00968                 if (m_dynlib.PresetManager_InformChange) {
00969                     m_dynlib.PresetManager_InformChange(instance, relative_path.c_str(),
      (cloudx1_PresetChange_Kind) change_kind);
00970                 }
00971             }
00972
00984         void SetMetadata(const std::string& key, const char* value)
```

```
00985             {
00986                 if (m_dynlib.SetMetadata) {
00987                     if (instance) {
00988                         m_dynlib.SetMetadata(instance, key.c_str(), value);
00989                     } else {
00990                         init_metadata.push_back( {key, value} );
00991                     }
00992                 }
00993             }
00994 
00995 
01005         void SetMetadataMulti(const std::unordered_map<std::string, const char*>& list)
01006             {
01007                 if (instance) {
01008                     if (m_dynlib.SetMetadataMulti) {
01009                         typedef const char* pchar;
01010                         pchar* keyvalue=new pchar[2*list.size()+1];
01011                         size_t n=0;
01012                         for (auto&& [key,value]: list) {
01013                             keyvalue[2*n+0]=key.c_str();
01014                             keyvalue[2*n+1]=value;
01015                             n++;
01016                         }
01017                         keyvalue[2*n+0]=nullptr;
01018 
01019                         cloudx1_SetMetadataMulti(instance, keyvalue);
01020 
01021                         delete[] keyvalue;
01022                     } else {
01023                         for (auto&& [key,value]: list) {
01024                             SetMetadata(key, value);
01025                         }
01026                     }
01027                 } else {
01028                     for (auto&& [key,value]: list) {
01029                         if (value) {
01030                             init_metadata.push_back( {key, value} );
01031                         }
01032                     }
01033                 }
01034             }
01035 
01041         std::vector< std::tuple<std::string,std::string> > GetMetadataInfos()
01042             {
01043                 std::vector< std::tuple<std::string,std::string> > values;
01044                 if (m_dynlib.GetMetadataInfos) {
01045                     const char** c_values = m_dynlib.GetMetadataInfos(instance);
01046                     if (c_values) {
01047                         for (const char** c_value=c_values; *c_value; ) {
01048                             std::string key(*c_value);
01049                             c_value++;
01050                             if (*c_value) {
01051                                 std::string descr(*c_value);
01052                                 c_value++;
01053                                 values.push_back( {key,descr} );
01054                             }
01055                         }
01056                         m_dynlib.FreeMetadataInfos(instance, c_values);
01057                     }
01058                 }
01059                 return values;
01060             }
01061 
01062         // Can be overloaded, called at start of UpdateThread in the thread
01063         virtual void OnUpdateThreadStart() {}
01064         // Can be overloaded to call at end of UpdateThread in the thread
01065         virtual void OnUpdateThreadStop() {}
01066 
01067 #if CLOUDX1_HAS_WEBSERVER
01073         bool StartWebServer(int http_port, int https_port=0)
01074             {
01075                 assert(instance);
01076                 assert(webserver==SOUND4_INVALID_WEBSERVER_ID);
01077                 webserver = m_dynlib.Webserver(http_port,https_port,instance);
01078                 if (webserver==SOUND4_INVALID_WEBSERVER_ID) {
01079                     return false;
01080                 }
01081                 int web_status=m_dynlib.Webserver_Status(webserver);
01082                 if (http_port && (web_status & SOUND4_WEBSERVER_HTTP_OK) == 0) {
01083                     return false;
01084                 }
01085                 if (https_port && (web_status & SOUND4_WEBSERVER_HTTPS_OK) == 0) {
01086                     return false;
01087                 }
01088                 return true;
01089             }
01095         void StopWebServer()
```

```
01096             {
01097                 if (webserver != SOUND4_INVALID_WEBSERVER_ID) {
01098                     m_dynlib.Webserver_Stop(webserver, 1000);
01099                     webserver = SOUND4_INVALID_WEBSERVER_ID;
01100                 }
01101             }
01102
01106         void SetWebServerAppHealth(int httpcode, const std::string& contenttype, const std::string&
      content)
01107             {
01108                 assert(instance);
01109                 if (!m_dynlib.Webserver_SetAppHealth) return;
01110                 m_dynlib.Webserver_SetAppHealth(instance, httpcode, contenttype.c_str(),
      content.c_str());
01111             }
01115         void GetWebServerAppHealth(int &httpcode, std::string& contenttype, std::string& content)
01116             {
01117                 assert(instance);
01118                 if (!m_dynlib.Webserver_GetAppHealth) return;
01119                 char* c_contenttype=nullptr;
01120                 char* c_content=nullptr;
01121                 m_dynlib.Webserver_GetAppHealth(instance, &httpcode, &c_contenttype, &c_content);
01122                 contenttype=c_contenttype;
01123                 content=c_content;
01124                 m_dynlib.Webserver_FreeString(c_contenttype);
01125                 m_dynlib.Webserver_FreeString(c_content);
01126             }
01127 #endif // CLOUDX1_HAS_WEBSERVER
01128
01135         int Terminate()
01136             {
01137                 if (!instance) return 0;
01138                 return m_dynlib.TerminateProcess(instance);
01139             }
01140
01146         void Stop()
01147             {
01148                 if (!instance) return;
01149 #if CLOUDX1_HAS_WEBSERVER
01150                 StopWebServer();
01151 #endif // CLOUDX1_HAS_WEBSERVER
01152                 if (update_thread.joinable()) {
01153                     m_dynlib.StopUpdateThread(instance); // this will exit the update thread
01154                     update_thread.join();
01155                 }
01156                 m_dynlib.ExitProcess(instance);
01157                 instance=nullptr;
01158             }
01159
01165         unsigned int GetEstimatedDelay()
01166             {
01167                 assert(instance);
01168                 return m_dynlib.GetEstimatedDelay(instance);
01169             }
01170
01181         std::array<float,InputSampleSize>& GetBufferIn()
01182             {
01183                 assert(instance);
01184                 float *buf=m_dynlib.GetBufferIn(instance);
01185                 return reinterpret_cast< std::array<float,InputSampleSize>& >(*buf);
01186
01187             }
01198         std::array<float,OutputSampleSize>& GetBufferOut()
01199             {
01200                 assert(instance);
01201                 float *buf= m_dynlib.GetBufferOut(instance);
01202                 return reinterpret_cast< std::array<float,OutputSampleSize>& >(*buf);
01203             }
01209         void ProcessAudio()
01210             {
01211                 assert(instance);
01212                 m_dynlib.ProcessAudio(instance, m_dynlib.GetBufferIn(instance),
      m_dynlib.GetBufferOut(instance));
01213             }
01227         // void ProcessAudio_Planar(const std::array< const std::array<const float,ChunkFrames>,
      InputChannels>& input, std::array< std::array<float,ChunkFrames>, OutputChannels>& output)
01228         void ProcessAudio_Planar(float const * const *input, float * const *output)
01229             {
01230                 assert(instance);
01231                 m_dynlib.ProcessAudio_Planar(instance, input, output);
01232             }
01233
01245             template<typename T>
01246         std::vector<T> ProcessAnyAudio(const std::vector<T> input)
01247             {
01248                 assert(instance);
01249                 std::vector<T> output;
```

```
01250                 unsigned int out_offset=0;
01251                 unsigned int in_offset=0;
01252                 unsigned int todo = input.size();
01253                 while (todo>0) {
01254                     unsigned int left = AddAudio(&input[in_offset], todo);
01255                     unsigned int out_avail = m_dynlib.GetOutputCount(instance);
01256                     output.resize(out_offset + out_avail);
01257                     GetAudio(&output[out_offset], out_avail);
01258                     out_offset+=out_avail;
01259                     in_offset += todo-left;
01260                     todo=left;
01261                 }
01262                 return output;
01263             }
01264 
01270         class CClient {
01271             public:
01272                 CClient(cloudx1_CInstance* instance, CDynLib& dynlib)
01273                 : m_dynlib(dynlib)
01274                     {
01275                         assert(instance);
01276                         client=m_dynlib.NewClient(instance);
01277                         if (!client) throw std::bad_alloc();
01278                     }
01279                 // non-copyable
01280                 CClient( const CInstance& ) = delete;
01281                 CClient& operator=( const CInstance& ) = delete;
01282                 ~CClient()
01283                     {
01284                         if (client) {
01285                             m_dynlib.DeleteClient(client);
01286                             client=nullptr;
01287                         }
01288                     }
01296                 std::string ProcessJson(const std::string &request, bool *NeedSave = nullptr)
01297                     {
01298                         assert(client);
01299                         int need_save=0;
01300                         const char *canswer = m_dynlib.ProcessJson(client, request.c_str(),
    &need_save);
01301                         if (!canswer) return {};
01302                         std::string answer(canswer);
01303                         m_dynlib.FreeJsonAnswer (canswer);
01304                         if (NeedSave) {
01305                             *NeedSave=(need_save!=0);
01306                         }
01307                         return answer;
01308                     }
01309             private:
01310                 CDynLib& m_dynlib;
01311                 cloudx1_CClientInstance *client = nullptr;
01312         };
01320         std::shared_ptr<CClient> NewClient()
01321             {
01322                 assert(instance);
01323                 return std::make_shared<CClient>(instance, m_dynlib);
01324             }
01325 
01326     protected:
01327             template<typename T>
01328         unsigned int AddAudio(const T* payload, unsigned int nFrame)
01329             {
01330                 assert(instance);
01331                 return m_dynlib.AddAudio(instance, reinterpret_cast<const T*>(payload), nFrame,
    helper::SampleFormat<T>::format);
01332             }
01333             template<typename T>
01334         unsigned int GetAudio(T* payload, unsigned int max_nFrame)
01335             {
01336                 assert(instance);
01337                 return m_dynlib.GetAudio(instance, reinterpret_cast<T*>(payload), max_nFrame,
    helper::SampleFormat<T>::format);
01338             }
01339 
01340     private:
01341         CDynLib& m_dynlib;
01342         cloudx1_CParameters* params = nullptr;
01343         cloudx1_CInstance* instance = nullptr;
01344         std::thread update_thread;
01345         std::vector< std::pair<std::string,std::string> > init_metadata;
01346 #if CLOUDX1_HAS_WEBSERVER
01347         uint64_t webserver = SOUND4_INVALID_WEBSERVER_ID;
01348 #endif // CLOUDX1_HAS_WEBSERVER
01349     };
01350 
01351 }; // namespace dyn
01352 }; // namespace cloudx1
```

```
01353 }; // namespace sound4
01354
01355
01356 // bridge C callbacks to CPresetLoader
01357 extern "C" {
01358     static char *cloudx1_custom_reader(const fs_char *filename, void* userdata) {
01359         sound4::cloudx1::dyn::CPresetLoader
      *preset_loader=reinterpret_cast<sound4::cloudx1::dyn::CPresetLoader*>(userdata);
01360         std::string content=preset_loader->Read(filename);
01361         return strdup(content.c_str());
01362     }
01363     static void cloudx1_custom_reader_free(char *content, void* userdata) {
01364         if (!content) return;
01365         free(content);
01366     }
01367     static int cloudx1_custom_writer(const fs_char *filename, const char *content, void* userdata) {
01368         sound4::cloudx1::dyn::CPresetLoader
      *preset_loader=reinterpret_cast<sound4::cloudx1::dyn::CPresetLoader*>(userdata);
01369         auto res=preset_loader->Write(filename,content);
01370         return res?0:-1;
01371     }
01372     static int cloudx1_custom_exists(const fs_char *filename, void* userdata) {
01373         sound4::cloudx1::dyn::CPresetLoader
      *preset_loader=reinterpret_cast<sound4::cloudx1::dyn::CPresetLoader*>(userdata);
01374         auto res=preset_loader->Exists(filename);
01375         return res?0:-1;
01376     }
01377     static fs_char** cloudx1_custom_getall(void* userdata) {
01378         sound4::cloudx1::dyn::CPresetLoader
      *preset_loader=reinterpret_cast<sound4::cloudx1::dyn::CPresetLoader*>(userdata);
01379         auto res=preset_loader->GetAll();
01380         fs_char**all=(fs_char**)malloc((res.size()+1)*sizeof(const fs_char*));
01381         for (size_t n=0;n<res.size();n++) {
01382             all[n]=fs_strdup(res[n].c_str());
01383         }
01384         all[res.size()]=nullptr;
01385         return all;
01386     }
01387     static void cloudx1_custom_getall_free(fs_char** all, void* userdata) {
01388         if (!all) return;
01389         for (fs_char** one=all;*one;one++) {
01390             free(*one);
01391         }
01392         free(all);
01393     }
01394     static int cloudx1_custom_remove(const fs_char *filename, void* userdata) {
01395         sound4::cloudx1::dyn::CPresetLoader
      *preset_loader=reinterpret_cast<sound4::cloudx1::dyn::CPresetLoader*>(userdata);
01396         auto res=preset_loader->Remove(filename);
01397         return res?0:-1;
01398     }
01399     static int cloudx1_custom_rename(const fs_char *from, const fs_char *to, void* userdata) {
01400         sound4::cloudx1::dyn::CPresetLoader
      *preset_loader=reinterpret_cast<sound4::cloudx1::dyn::CPresetLoader*>(userdata);
01401         auto res=preset_loader->Rename(from,to);
01402         return res?0:-1;
01403     }
01404
01405 };
```

11.13 sound4cl-pipe.cpp File Reference

C++ sample using C++ interface wrapper with dynamic library loading Works with any SOUND4 processor provided.

```
#include <unistd.h>
#include <iostream>
#include <strings.h>
#include <sys/types.h>
#include <filesystem>
#include <fstream>
#include <sstream>
#include <io.h>
#include <fcntl.h>
#include "sound4cl.hpp"
```

Classes

- class MyPresetManager

  [Custom preset manager]

Functions

- std::string GetEnv (const std::string &name)
- void ProcessLoop (sound4::CInstance &instance, sound4::CProcessor &processor)
- int main (int argc, const char **argv)

  [Custom preset manager]

11.13.1 Function Documentation

11.13.1.1 GetEnv()

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

Definition at line 60 of file sound4cl-pipe.cpp.

11.13.1.2 main()

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

[Dynamic loading config]

[Dynamic loading config]

[Dynamic loading]

[Dynamic loading]

[Setup and configure]

[Setup and configure]

[Custom preset manager loading]

[Custom preset manager loading]

[Create the processing instance]

[Create the processing instance]

[Starting a web-server]

[Starting a web-server]

[Communicating with the instance]

[Communicating with the instance]

Definition at line 178 of file sound4cl-pipe.cpp.

11.13.1.3 ProcessLoop()

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

[Processing Loop]

[Processing Loop]

Definition at line 69 of file sound4cl-pipe.cpp.

11.14 sound4cl-pipe.cpp

```
00001
00006 #include <unistd.h>
00007 #include <iostream>
00008 #include <strings.h>
00009 #include <sys/types.h>
00010 #include <filesystem>
00011 #include <fstream>
00012 #include <sstream>
00013 #if defined(_WIN32) || defined(WIN32)
00014     #include <io.h>
00015     #include <fcntl.h>
00016 #endif
00017
00018 #include "sound4cl.hpp"
00019
00020 // ------------------------------------------------------------
00060 std::string GetEnv(const std::string& name)
00061 {
00062     char *data=getenv(name.c_str());
00063     if (!data) return {};
00064     return std::string(data);
00065 }
00066
00067
00068 // Reads from stdin, write to stdout
00069 void ProcessLoop(sound4::CInstance &instance, sound4::CProcessor &processor)
00070 {
00071 bool stopthread=false;
00072
00073     #if defined(_WIN32) || defined(WIN32)
00074         // Windows: The stdin, stdout, and stderr streams always open in text mode by default
00075         _setmode(_fileno(stdin), _O_BINARY);
00076         _setmode(_fileno(stdout), _O_BINARY);
00077     #endif
00078
00080     size_t input_sample_count = instance.GetChunkFrames() * processor.GetChannelCount();
00081     size_t output_sample_count = instance.GetChunkFrames() * processor.GetChannelCount();
00082     while (!stopthread)
00083     {
00084         // Read a chunk from stdin
00085         if (std::fread(instance.GetBufferIn(), sizeof(float), input_sample_count, stdin) !=
      input_sample_count) {
00086             std::cerr « "ProcessLoop: input stopped"« std::endl;
00087             stopthread=true;
00088             break;
00089         }
00090         if (stopthread) break;
00091         // Process it
00092         instance.ProcessAudio();
00093         // Write it to stdout
00094         if (fwrite(instance.GetBufferOut(), sizeof(float), output_sample_count, stdout) !=
      output_sample_count) {
00095             std::cerr « "Failed to write audio output" « std::endl;
00096             stopthread=true;
00097         }
00098     }
00100 }
00101
00103 class MyPresetManager : public sound4::CPresetLoader {
00104 public:
00105     MyPresetManager() = default;
00106     virtual ~MyPresetManager() = default;
00107
```

```
00108     bool SetupPath(const std::filesystem::path& a_preset_path)
00109         {
00110             m_preset_path=a_preset_path;
00111             if ( !std::filesystem::exists(m_preset_path ) ) {
00112                 return false;
00113             }
00114             std::error_code ec;
00115             if ( !std::filesystem::is_directory(m_preset_path, ec) ) {
00116                 return false;
00117             }
00118             return true;
00119         }
00120     virtual bool IsReadOnly() override { return false; };
00121     virtual bool Exists(const std::filesystem::path &name) override
00122         {
00123             std::error_code ec;
00124             return std::filesystem::exists(GetRealPath(name), ec);
00125         }
00126     virtual bool Remove(const std::filesystem::path &name) override
00127         {
00128             std::error_code ec;
00129             return std::filesystem::remove(GetRealPath(name), ec);
00130         }
00131     virtual bool Rename(const std::filesystem::path &from, const std::filesystem::path &to) override
00132         {
00133             std::error_code ec;
00134             std::filesystem::rename(GetRealPath(from), GetRealPath(to), ec);
00135             return (!ec);
00136         }
00137     virtual std::vector<std::filesystem::path> GetAll() override
00138         {
00139             std::vector<std::filesystem::path> list;
00140             for (const auto & entry : std::filesystem::directory_iterator(m_preset_path.c_str(),
      std::filesystem::directory_options::skip_permission_denied)) {
00141                 if (entry.is_regular_file()) {
00142                     list.push_back(entry.path().lexically_relative(m_preset_path));
00143                 }
00144             }
00145             return list;
00146         }
00147     virtual std::string Read(const std::filesystem::path &filename) override
00148         {
00149             std::ifstream f(GetRealPath(filename));
00150             if ( f.fail() ) {
00151                 return std::string();
00152             }
00153             std::ostringstream ss;
00154             ss « f.rdbuf();
00155             return ss.str();
00156         }
00157     virtual bool Write(const std::filesystem::path &filename, const std::string &content) override
00158         {
00159             std::ofstream f(GetRealPath(filename));
00160             if ( f.fail() ) {
00161                 return false;
00162             }
00163             f « content;
00164             if ( f.fail() ) {
00165                 return false;
00166             }
00167             return true;
00168         }
00169 private:
00170     std::filesystem::path m_preset_path;
00171     std::filesystem::path GetRealPath(const std::filesystem::path &name)
00172         {
00173             return m_preset_path / name;
00174         }
00175 };
00177
00178 int main(int argc, const char **argv)
00179 {
00180     int listen_port = 8080;
00181     auto store_path_str=GetEnv("S4STORE_PATH");
00183     std::string libname =  "libsound4.x1.cloud.so";
00185
00186
00187     // Get credentials from environment
00188     auto LoginKey=GetEnv("S4LOGINKEY");
00189     if (LoginKey.empty()) {
00190         LoginKey=GetEnv("S4USERID");
00191     }
00192     auto RadioName=GetEnv("RADIO_NAME");
00193     auto KeyId=GetEnv("ACCESS_KEY_ID");
00194     auto AccessKey=GetEnv("ACCESS_KEY_SECRET");
00195
00196     const char *progname=argv[0];
```

```
00197     for (int n=1;n<argc;n++) {
00198         std::string arg = argv[n];
00199         if (arg=="-p" || arg=="--port") {
00200             if (argc<n+1) {
00201                 std::cerr « "--port needs a number"« std::endl;
00202                 return 1;
00203             }
00204             n++;
00205             listen_port=atoi(argv[n]);
00206         } else if (arg=="-l" || arg=="--lib") {
00207             if (argc<n+1) {
00208                 std::cerr « "--port needs a number"« std::endl;
00209                 return 1;
00210             }
00211             n++;
00212             libname=argv[n];
00213         } else if (arg=="-h" || arg=="--help") {
00214             std::cerr « "Usage: " « progname « " [-p port] [-l libname.so]"«std::endl;;
00215             std::cerr « "  Reads audio from stdin and return processed audio in stdout" « std::endl;
00216             std::cerr « "   -l, --lib   : library to load ["«libname«"]" « std::endl;
00217             std::cerr « "   -p, --port  : port for web server ["«listen_port«"]" « std::endl;
00218             std::cerr « std::endl;
00219             std::cerr « "Environment:"« std::endl;;
00220             std::cerr « " - S4STORE_PATH            : path to store config [default: (no storage)]"«
      std::endl;;
00221             std::cerr « " - S4LOGINKEY              : SOUND4 Login key (or S4USERID)"« std::endl;;
00222             std::cerr « " - RADIO_NAME              : SOUND4 Radio Name"« std::endl;;
00223             std::cerr « " - ACCESS_KEY_ID           : SOUND4 Access Key ID"« std::endl;;
00224             std::cerr « " - ACCESS_KEY_SECRET       : SOUND4 Access Key Secret"« std::endl;
00225             std::cerr « std::endl;
00226             std::cerr « "Example:"« std::endl;
00227             std::cerr « "  ffmpeg -re -i file.wav -ar 48000 -ac 2 -f f32le - | "«progname«" | ffplay
      -f f32le -ar 48000 -ac 2 -i -" « std::endl;;
00228             #if defined(_WIN32) || defined(WIN32)
00229             std::cerr « "  ffmpeg -f dshow -i audio=\"Microphone\" -ar 48000 -ac 2 -f f32le - |
      "«progname«" | ffplay -f f32le -ar 48000 -ac 2 -i -" « std::endl;;
00230             #else
00231             std::cerr « "  ffmpeg -f pulse -i default -ar 48000 -ac 2 -f f32le - | "«progname«" |
      ffplay -f f32le -ar 48000 -ac 2 -i -" « std::endl;;
00232             #endif
00233             return 0;
00234         } else {
00235             std::cerr « "unknown argument '"«arg«"'"« std::endl;
00236             return 1;
00237         }
00238     }
00239
00240
00242     sound4::CProcessor processor;
00243     if (!processor.Load(libname)) {
00244         std::cerr « "Failed to load processing library" « std::endl;
00245         return 1;
00246     }
00248
00249     if (LoginKey.empty() || RadioName.empty() || KeyId.empty() || AccessKey.empty()) {
00250         std::cerr « "Missing credentials in environment" « std::endl;
00251         return 1;
00252     }
00253     std::filesystem::path store_path;
00254     if (!store_path_str.empty()) {
00255         store_path = std::filesystem::path(store_path_str);
00256         if (!std::filesystem::exists(store_path) || !std::filesystem::is_directory(store_path)) {
00257             std::cerr « "Storage folder '"«store_path.c_str()«"' is not a valid directory" «
      std::endl;
00258             return 1;
00259         }
00260     }
00261
00263     processor.SetLogSeverity(sound4::verbose5);
00264     processor.SetLoggerCallback([](sound4::LogSeverity severity,const std::string& msg) {
00265             std::cerr « msg « std::endl;
00266         });
00267
00268     sound4::CInstance instance(processor);
00269     instance.SetParam("ADMIN_USER","admin");
00270     instance.SetParam("ADMIN_SECRET","admin");
00272
00274     MyPresetManager preset_manager;
00275     if (!store_path.empty()) {
00276         if (!preset_manager.SetupPath(store_path)) {
00277             std::cerr « "Failed to setup the custom preset manager on path " « store_path.c_str() «
      std::endl;
00278             return 1;
00279         }
00280         instance.SetPresetManager(&preset_manager);
00281     }
00283
```

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

11.15 sound4cl.hpp File Reference

C++ wrapper for dynamic loading library.

```
#include <string>
#include <exception>
#include <thread>
#include <functional>
#include <memory>
#include <array>
#include <assert.h>
#include <stdexcept>
#include <filesystem>
#include <string.h>
#include <libloaderapi.h>
#include "sound4cl_cdef.h"
```

Classes

- struct sound4::helper::SampleFormat< T >
  Helper for Sample format types.
- struct sound4::helper::SampleFormat< int16_t >
- struct sound4::helper::SampleFormat< int32_t >
- struct sound4::helper::SampleFormat< float >
- class sound4::helper::DynFuncHolder< T >
  Helper to wrap a function pointer.
- class sound4::helper::CDynLoader
  Helper to load dynamic library.
- class sound4::CProcessor
  Dynamic library interface.
- class sound4::CBus
  Shared Bus.
- class sound4::CPresetLoader
  Custom preset handler helper.
- class sound4::CInstance
  Instance handling class.
- class sound4::CInstance::CClient

Namespaces

- sound4
- sound4::helper

Typedefs

- using sound4::helper::DynLib_t = HMODULE
- using sound4::helper::DynFunc_t = FARPROC
- using sound4::LogSeverity = sound4cl_LogSeverity
- using sound4::log_cb_t = std::function< void(LogSeverity, const std::string &)>
- using sound4::PresetChange_Kind = sound4cl_PresetChange_Kind

Variables

- const LogSeverity sound4::none = (LogSeverity) sound4cl_none
- const LogSeverity sound4::fatal = (LogSeverity) sound4cl_fatal
- const LogSeverity sound4::error = (LogSeverity) sound4cl_error
- const LogSeverity sound4::warning = (LogSeverity) sound4cl_warning
- const LogSeverity sound4::info = (LogSeverity) sound4cl_info
- const LogSeverity sound4::verbose = (LogSeverity) sound4cl_verbose
- const LogSeverity sound4::verbose2 = (LogSeverity) sound4cl_verbose2
- const LogSeverity sound4::verbose3 = (LogSeverity) sound4cl_verbose3
- const LogSeverity sound4::verbose4 = (LogSeverity) sound4cl_verbose4
- const LogSeverity sound4::verbose5 = (LogSeverity) sound4cl_verbose5

11.15.1 Detailed Description

You need to load the dynamic library, and if it succeed you can use it.

Definition in file sound4cl.hpp.

11.16 sound4cl.hpp

```
00001
00008 #pragma once
00009
00010 #include <string>
00011 #include <exception>
00012 #include <thread>
00013 #include <functional>
00014 #include <memory>
00015 #include <array>
00016 #include <assert.h>
00017 #include <stdexcept>
00018 #include <filesystem>
00019 #include <string.h>
00020
00021 #ifdef __unix__
00022     #include <dlfcn.h>
00023 #elif defined(_WIN32)
00024     #include <libloaderapi.h>
00025 #else
00026     #error "Unsupported OS"
00027 #endif
00028
00029 #include "sound4cl_cdef.h"
00030
00031     // bridge C callbacks to CPresetLoader
00032     extern "C" {
```

```
00033         static char *sound4cl_custom_reader(const fs_char *filename, void* userdata);
00034         static void sound4cl_custom_reader_free(char *content, void* userdata);
00035         static int sound4cl_custom_writer(const fs_char *filename, const char *content, void*
      userdata);
00036         static int sound4cl_custom_exists(const fs_char *filename, void* userdata);
00037         static fs_char** sound4cl_custom_getall(void* userdata);
00038         static void sound4cl_custom_getall_free(fs_char** all, void* userdata);
00039         static int sound4cl_custom_remove(const fs_char *filename, void* userdata);
00040         static int sound4cl_custom_rename(const fs_char *from, const fs_char *to, void* userdata);
00041     };
00042
00046 namespace sound4 {
00047
00051     namespace helper {
00052         #ifdef _WIN32
00058         static inline std::string WStringToUTF8(const std::wstring& wstr) {
00059             if (wstr.empty()) return {};
00060             int size = WideCharToMultiByte(CP_UTF8, 0, &wstr[0], (int)wstr.size(), NULL, 0, NULL,
      NULL);
00061             std::string str(size, 0);
00062             WideCharToMultiByte(CP_UTF8, 0, &wstr[0], (int)wstr.size(), &str[0], size, NULL, NULL);
00063             return str;
00064         }
00070         static inline std::wstring UTF8ToWString(const std::string& str) {
00071             if (str.empty()) return std::wstring();
00072             int size = MultiByteToWideChar(CP_UTF8, 0, &str[0], (int)str.size(), NULL, 0);
00073             std::wstring wstr(size, 0);
00074             MultiByteToWideChar(CP_UTF8, 0, &str[0], (int)str.size(), &wstr[0], size);
00075             return wstr;
00076         }
00077         #endif // _WIN32
00078
00084             template<typename T>
00085         struct SampleFormat {
00086         };
00087             template<>
00088         struct SampleFormat<int16_t> {
00089             const sound4cl_SampleFormat format = S16_NATIVE;
00090         };
00091             template<>
00092         struct SampleFormat<int32_t> {
00093             const sound4cl_SampleFormat format = S32_NATIVE;
00094         };
00095             template<>
00096         struct SampleFormat<float> {
00097             const sound4cl_SampleFormat format = F32_NATIVE;
00098         };
00099
00100         // Storage definition, depending on OS
00101         #ifdef __unix__
00102             using DynLib_t = void *;
00103             using DynFunc_t = void *;
00104         #else
00105             using DynLib_t = HMODULE;
00106             using DynFunc_t = FARPROC;
00107         #endif
00108
00114             template <typename T>
00115         class DynFuncHolder {
00116         private:
00117             DynFunc_t m_ptr = NULL;
00118         public:
00119             DynFuncHolder() = default;
00120             DynFuncHolder(DynFunc_t a_ptr) : m_ptr(a_ptr) {}
00121             operator bool() const { return (m_ptr!=NULL); }
00122             bool IsOk() const { return m_ptr!=NULL; }
00123             operator T* () const { return reinterpret_cast<T*>(m_ptr); }
00124         };
00125
00130         class CDynLoader {
00131             private:
00132                 DynLib_t m_lib {};
00133             public:
00134                 CDynLoader() = default;
00135                 CDynLoader(CDynLoader&&) = default;
00136                 // No copy allowed
00137                 CDynLoader(const CDynLoader&) = delete;
00138                 CDynLoader& operator=(CDynLoader const&) = delete;
00139                 operator bool() const {
00140                     return (m_lib!=NULL);
00141                 }
00142                 bool IsOk() const {
00143                     return (m_lib!=NULL);
00144                 }
00145                 #ifdef __unix__
00146                     void Close() {
00147                         if (IsOk()) {
```

```
00148                           #ifndef __SANITIZE_ADDRESS__ // avoid unloading to keep call stack
00149                           dlclose(m_lib);
00150                           #endif // __SANITIZE_ADDRESS__
00151                           m_lib=NULL;
00152                       }
00153                   }
00154                   bool Load(const std::filesystem::path& dynlib)
00155                   {
00156                       Close();
00157                       m_lib=dlopen(dynlib.c_str(), RTLD_NOW|RTLD_LOCAL);
00158                       return IsOk();
00159                   }
00160                       template <typename T>
00161                   DynFuncHolder<T> GetSymbol(const std::string& name) {
00162                       auto ptr=dlsym(m_lib, name.c_str());
00163                       if (!ptr) {
00164                           throw std::runtime_error("Missing function in library");
00165                       }
00166                       return DynFuncHolder<T>(ptr);
00167                   }
00168                       template <typename T>
00169                   DynFuncHolder<T> GetSymbol_safe(const std::string& name) {
00170                       auto ptr=dlsym(m_lib, name.c_str());
00171                       return DynFuncHolder<T>(ptr);
00172                   }
00173                   static std::filesystem::path GetThisLibraryPath(void) {
00174                       static Dl_info info;
00175                       if (! dladdr((void *) GetThisLibraryPath, & info)) return {};
00176                       auto rp=realpath(info.dli_fname, NULL);
00177                       std::filesystem::path p(rp);
00178                       free(rp);
00179                       return p;
00180                   }
00181               #else
00182                   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_
00183                   {
00184                       Close();
00185                       m_lib=LoadLibraryExW(dynlib.c_str(), NULL, loadflags);
00186                       return IsOk();
00187                   }
00188                   void Close() {
00189                       if (IsOk()) {
00190                           // NOTE: On Windows, FreeLibrary may create a lot of troubles, should use
    FreeLibraryAndExitThread but...
00191                           // So define SOUND4_CALL_FREELIBRARY before including this file if you
    really want to unload the library dynamically
00192                           #if defined(SOUND4_CALL_FREELIBRARYANDEXITTHREAD)
00193                               FreeLibraryAndExitThread(m_lib, 0);
00194                           #elif defined(SOUND4_CALL_FREELIBRARY)
00195                               FreeLibrary(m_lib);
00196                           #endif // SOUND4_CALL_FREELIBRARY
00197                           m_lib=NULL;
00198                       }
00199                   }
00200                       template <typename T>
00201                   DynFuncHolder<T> GetSymbol(const std::string& name) {
00202                       auto ptr=GetProcAddress(m_lib, name.c_str());
00203                       if (!ptr) {
00204                           throw std::runtime_error("Missing function in library");
00205                       }
00206                       return DynFuncHolder<T>(ptr);
00207                   }
00208                       template <typename T>
00209                   DynFuncHolder<T> GetSymbol_safe(const std::string& name) {
00210                       auto ptr=GetProcAddress(m_lib, name.c_str());
00211                       return DynFuncHolder<T>(ptr);
00212                   }
00213                   static std::filesystem::path GetThisLibraryPath(void) {
00214                       static wchar_t path[MAX_PATH]={};
00215                       HMODULE hm = NULL;
00216                       if (GetModuleHandleExW(GET_MODULE_HANDLE_EX_FLAG_FROM_ADDRESS |
00217                               GET_MODULE_HANDLE_EX_FLAG_UNCHANGED_REFCOUNT,
00218                               (LPCWSTR) &GetThisLibraryPath, &hm))
00219                       {
00220                           if (GetModuleFileNameW(hm, path, MAX_PATH) > 0) {
00221                               return std::filesystem::path(path);
00222                           }
00223                       }
00224                       return {};
00225                   }
00226               #endif
00227               virtual ~CDynLoader() {
00228                   Close();
00229               }
00230       }; // class CDynLoader
00231   }; // namespace helper
```

```
00232
00233
00237     using LogSeverity = sound4cl_LogSeverity;
00238     const LogSeverity none     = (LogSeverity) sound4cl_none;
00239     const LogSeverity fatal    = (LogSeverity) sound4cl_fatal;
00240     const LogSeverity error    = (LogSeverity) sound4cl_error;
00241     const LogSeverity warning  = (LogSeverity) sound4cl_warning;
00242     const LogSeverity info     = (LogSeverity) sound4cl_info;
00243     const LogSeverity verbose  = (LogSeverity) sound4cl_verbose;
00244     const LogSeverity verbose2 = (LogSeverity) sound4cl_verbose2;
00245     const LogSeverity verbose3 = (LogSeverity) sound4cl_verbose3;
00246     const LogSeverity verbose4 = (LogSeverity) sound4cl_verbose4;
00247     const LogSeverity verbose5 = (LogSeverity) sound4cl_verbose5;
00248
00252     using log_cb_t = std::function<void(LogSeverity,const std::string&)>;
00253
00257     static log_cb_t _log_cb;
00258     extern "C" {
00259         static inline void _log_cb_c(LogSeverity severity, const char *c_msg) {
00260             _log_cb(severity, std::string(c_msg));
00261         }
00262     }
00263
00270     class CProcessor {
00271     private:
00272         helper::CDynLoader m_lib;
00273         bool m_bOK = false;
00274         std::string m_prefix; // The prefix of functions, depends on the library (cloud_, impact_...)
00275     protected:
00276         friend class CBus;
00277         friend class CInstance;
00278
00279         // All those wrapped functions have the same signature as there C sound4cl_XXX
00280         helper::DynFuncHolder< decltype(sound4cl_GetVersion             ) > fnGetVersion             ;
00281         helper::DynFuncHolder< decltype(sound4cl_GetChunkSizeInFrames   ) > fnGetChunkSizeInFrames   ;
00282         helper::DynFuncHolder< decltype(sound4cl_GetChannelCount        ) > fnGetChannelCount        ;
00283         helper::DynFuncHolder< decltype(sound4cl_GetAudioInputCount     ) > fnGetAudioInputCount     ;
00284         helper::DynFuncHolder< decltype(sound4cl_GetAudioOutputCount    ) > fnGetAudioOutputCount    ;
00285         helper::DynFuncHolder< decltype(sound4cl_GetSampleRate          ) > fnGetSampleRate          ;
00286         helper::DynFuncHolder< decltype(sound4cl_SetLoggerCallback      ) > fnSetLoggerCallback      ;
00287         helper::DynFuncHolder< decltype(sound4cl_SetLogSeverity         ) > fnSetLogSeverity         ;
00288         helper::DynFuncHolder< decltype(sound4cl_NewParameters          ) > fnNewParameters          ;
00289         helper::DynFuncHolder< decltype(sound4cl_FreeParameters         ) > fnFreeParameters         ;
00290         helper::DynFuncHolder< decltype(sound4cl_SetParameter           ) > fnSetParameter           ;
00291         helper::DynFuncHolder< decltype(sound4cl_GetParameter           ) > fnGetParameter           ;
00292         helper::DynFuncHolder< decltype(sound4cl_FreeParameterValue     ) > fnFreeParameterValue     ;
00293         helper::DynFuncHolder< decltype(sound4cl_InitProcess            ) > fnInitProcess            ;
00294         helper::DynFuncHolder< decltype(sound4cl_InitProcess2           ) > fnInitProcess2           ;
00295         helper::DynFuncHolder< decltype(sound4cl_TerminateProcess       ) > fnTerminateProcess       ;
00296         helper::DynFuncHolder< decltype(sound4cl_ExitProcess            ) > fnExitProcess            ;
00297         helper::DynFuncHolder< decltype(sound4cl_StartUpdateThread      ) > fnStartUpdateThread      ;
00298         helper::DynFuncHolder< decltype(sound4cl_StopUpdateThread       ) > fnStopUpdateThread       ;
00299         helper::DynFuncHolder< decltype(sound4cl_WaitUpdateThreadReady  ) > fnWaitUpdateThreadReady  ;
00300         helper::DynFuncHolder< decltype(sound4cl_ProcessAudio           ) > fnProcessAudio           ;
00301         helper::DynFuncHolder< decltype(sound4cl_ProcessAudio_Planar    ) > fnProcessAudio_Planar    ;
00302         helper::DynFuncHolder< decltype(sound4cl_GetBufferIn            ) > fnGetBufferIn            ;
00303         helper::DynFuncHolder< decltype(sound4cl_GetBufferOut           ) > fnGetBufferOut           ;
00304         helper::DynFuncHolder< decltype(sound4cl_GetEstimatedDelay      ) > fnGetEstimatedDelay      ;
00305         helper::DynFuncHolder< decltype(sound4cl_GetFormatName          ) > fnGetFormatName          ;
00306         helper::DynFuncHolder< decltype(sound4cl_GetFormatFromName      ) > fnGetFormatFromName      ;
00307         helper::DynFuncHolder< decltype(sound4cl_GetBytesFromFormat     ) > fnGetBytesFromFormat     ;
00308         helper::DynFuncHolder< decltype(sound4cl_GetMaxPacketFrame      ) > fnGetMaxPacketFrame      ;
00309         helper::DynFuncHolder< decltype(sound4cl_AddAudio               ) > fnAddAudio               ;
00310         helper::DynFuncHolder< decltype(sound4cl_AddPadAudio            ) > fnAddPadAudio            ;
00311         helper::DynFuncHolder< decltype(sound4cl_GetOutputCount         ) > fnGetOutputCount         ;
00312         helper::DynFuncHolder< decltype(sound4cl_GetAudio               ) > fnGetAudio               ;
00313         helper::DynFuncHolder< decltype(sound4cl_AudioConvertFrom       ) > fnAudioConvertFrom       ;
00314         helper::DynFuncHolder< decltype(sound4cl_AudioConvertTo         ) > fnAudioConvertTo         ;
00315         helper::DynFuncHolder< decltype(sound4cl_StereoToMono           ) > fnStereoToMono           ;
00316         helper::DynFuncHolder< decltype(sound4cl_MonoToStereo           ) > fnMonoToStereo           ;
00317         helper::DynFuncHolder< decltype(sound4cl_AudioMonoFromLiveStereo) > fnAudioMonoFromLiveStereo;
00318         helper::DynFuncHolder< decltype(sound4cl_AudioMonoToLiveStereo  ) > fnAudioMonoToLiveStereo  ;
00319         helper::DynFuncHolder< decltype(sound4cl_NewClient              ) > fnNewClient              ;
00320         helper::DynFuncHolder< decltype(sound4cl_DeleteClient           ) > fnDeleteClient           ;
00321         helper::DynFuncHolder< decltype(sound4cl_ProcessJson            ) > fnProcessJson            ;
00322         helper::DynFuncHolder< decltype(sound4cl_FreeJsonAnswer         ) > fnFreeJsonAnswer         ;
00323         helper::DynFuncHolder< decltype(sound4cl_SaveState              ) > fnSaveState              ;
00324
00325         // only for processes supporting webserver
00326         helper::DynFuncHolder< decltype(sound4cl_Webserver_tcp          ) > fnWebserver_tcp          ;
00327         helper::DynFuncHolder< decltype(sound4cl_Webserver_tcp2         ) > fnWebserver_tcp2         ;
00328         helper::DynFuncHolder< decltype(sound4cl_Webserver              ) > fnWebserver              ;
00329         helper::DynFuncHolder< decltype(sound4cl_Webserver_Stop         ) > fnWebserver_Stop         ;
00330         helper::DynFuncHolder< decltype(sound4cl_Webserver_Status       ) > fnWebserver_Status       ;
00331
00332         // added 2023-03-22
00333         helper::DynFuncHolder< decltype(sound4cl_StereoToMono_Planar    ) > fnStereoToMono_Planar    ;
```

```
00334         helper::DynFuncHolder< decltype(sound4cl_MonoToStereo_Planar    ) > fnMonoToStereo_Planar    ;
00335
00336         // added 2023-05-22
00337         helper::DynFuncHolder< decltype(sound4cl_SetMetadata            ) > fnSetMetadata            ;
00338         helper::DynFuncHolder< decltype(sound4cl_GetMetadataInfos       ) > fnGetMetadataInfos       ;
00339         helper::DynFuncHolder< decltype(sound4cl_FreeMetadataInfos      ) > fnFreeMetadataInfos      ;
00340
00341         // added 2023-06-20
00342         helper::DynFuncHolder< decltype(sound4cl_SetPresetManager               ) >
      fnSetPresetManager      ;
00343         helper::DynFuncHolder< decltype(sound4cl_PresetManager_InformChange     ) >
      fnPresetManager_InformChange      ;
00344
00345         // added 2023-09-21
00346         helper::DynFuncHolder< decltype(sound4cl_GetPossibleChunkSizeInFrames   ) >
      fnGetPossibleChunkSizeInFrames   ;
00347         helper::DynFuncHolder< decltype(sound4cl_GetProcessChunkFrames          ) >
      fnGetProcessChunkFrames   ;
00348         helper::DynFuncHolder< decltype(sound4cl_InitProcess3                   ) > fnInitProcess3
        ;
00349
00350         // added 2023-06-26, only for processes supporting bus
00351         helper::DynFuncHolder< decltype(sound4cl_NewBus                 ) > fnNewBus      ;
00352         helper::DynFuncHolder< decltype(sound4cl_FreeBus                ) > fnFreeBus      ;
00353         helper::DynFuncHolder< decltype(sound4cl_SetInstanceBus         ) > fnSetInstanceBus      ;
00354
00355         // added 2024-02-19
00356         helper::DynFuncHolder< decltype(sound4cl_SetMetadataMulti       ) > fnSetMetadataMulti
      ;
00357
00358         // added 2024-05-23
00359         helper::DynFuncHolder< decltype(sound4cl_Webserver_SetAppHealth    ) >
      fnWebserver_SetAppHealth      ;
00360         helper::DynFuncHolder< decltype(sound4cl_Webserver_GetAppHealth    ) >
      fnWebserver_GetAppHealth      ;
00361         helper::DynFuncHolder< decltype(sound4cl_Webserver_FreeString      ) > fnWebserver_FreeString
       ;
00362
00363             template <typename T>
00364         helper::DynFuncHolder<T> GetPrefixSymbol(const std::string& name) {
00365             std::string prefix_name = m_prefix+name;
00366             return m_lib.GetSymbol<T>(prefix_name);
00367         }
00368
00369     public:
00370         CProcessor() = default;
00371
00379         bool Load(const std::filesystem::path& filepath) {
00380             m_lib.Close();
00381             // If a path is given, use it directly and do not try other path
00382             if (filepath.has_parent_path() && !m_lib.Load(filepath)) {
00383                 return false;
00384             } else if (!m_lib.IsOk()) {
00385                 auto thisdir=helper::CDynLoader::GetThisLibraryPath().parent_path();
00386                 auto filename = filepath.filename();
00387                 #ifndef _WIN32
00388                 // Linux: if in a bin directory, try ../lib first
00389                 if (!m_lib.IsOk() && thisdir.filename()=="bin") {
00390                     auto libdir = thisdir.parent_path() / "lib";
00391                     m_lib.Load(libdir / filename);
00392                 }
00393                 #endif // !_WIN32
00394                 if (!m_lib.IsOk()) {
00395                     // Search in the same directory this code is in
00396                     m_lib.Load(thisdir / filename);
00397                 }
00398                 if (!m_lib.IsOk()) {
00399                     // Try current path
00400                     m_lib.Load(std::filesystem::current_path() / filename);
00401                 }
00402                 if (!m_lib.IsOk()) {
00403                     return false;
00404                 }
00405             }
00406             // Get information from the library itself
00407             try {
00408                 auto GetInfo = m_lib.GetSymbol< decltype(sound4cl_SOUND4_GetProcessInfo             )
      >("SOUND4_GetProcessInfo"   );
00409                 auto info = GetInfo();
00410                 m_prefix = info->prefix;
00411             } catch (std::runtime_error& ) {
00412                 // Compatibility: search in known libraries without this function
00413                 #ifdef _WIN32
00414                     std::string filename = helper::WStringToUTF8(filepath.stem());
00415                 #else // !_WIN32
00416                     std::string filename = filepath.stem();
00417                     // remove lib prefix
```

```
00418                   filename=filename.substr(3);
00419               #endif // ! _WIN32
00420               if (filename=="sound4.x1.cloud") {
00421                   m_prefix="cloudx1_";
00422               } else if (filename=="sound4.impact.cl") {
00423                   m_prefix="impact_";
00424               } else if (filename=="sound4.bigvoice.cl") {
00425                   m_prefix="bigvoice_";
00426               } else {
00427                   return false;
00428               }
00429           }
00430
00431           // Load all C functions
00432           try {
00433               fnGetVersion              = GetPrefixSymbol< decltype(sound4cl_GetVersion
      ) >("GetVersion"             );
00434               fnGetChunkSizeInFrames    = GetPrefixSymbol< decltype(sound4cl_GetChunkSizeInFrames
      ) >("GetChunkSizeInFrames"   );
00435               fnGetChannelCount         = GetPrefixSymbol< decltype(sound4cl_GetChannelCount
      ) >("GetChannelCount"        );
00436               fnGetAudioInputCount      = GetPrefixSymbol< decltype(sound4cl_GetAudioInputCount
      ) >("GetAudioInputCount"     );
00437               fnGetAudioOutputCount     = GetPrefixSymbol< decltype(sound4cl_GetAudioOutputCount
      ) >("GetAudioOutputCount"    );
00438               fnGetSampleRate           = GetPrefixSymbol< decltype(sound4cl_GetSampleRate
      ) >("GetSampleRate"          );
00439               fnSetLoggerCallback       = GetPrefixSymbol< decltype(sound4cl_SetLoggerCallback
      ) >("SetLoggerCallback"      );
00440               fnSetLogSeverity          = GetPrefixSymbol< decltype(sound4cl_SetLogSeverity
      ) >("SetLogSeverity"         );
00441               fnNewParameters           = GetPrefixSymbol< decltype(sound4cl_NewParameters
      ) >("NewParameters"          );
00442               fnFreeParameters          = GetPrefixSymbol< decltype(sound4cl_FreeParameters
      ) >("FreeParameters"         );
00443               fnSetParameter            = GetPrefixSymbol< decltype(sound4cl_SetParameter
      ) >("SetParameter"           );
00444               fnGetParameter            = GetPrefixSymbol< decltype(sound4cl_GetParameter
      ) >("GetParameter"           );
00445               fnFreeParameterValue      = GetPrefixSymbol< decltype(sound4cl_FreeParameterValue
      ) >("FreeParameterValue"     );
00446               fnInitProcess             = GetPrefixSymbol< decltype(sound4cl_InitProcess
      ) >("InitProcess"            );
00447               fnInitProcess2            = GetPrefixSymbol< decltype(sound4cl_InitProcess2
      ) >("InitProcess2"           );
00448               fnTerminateProcess        = GetPrefixSymbol< decltype(sound4cl_TerminateProcess
      ) >("TerminateProcess"       );
00449               fnExitProcess             = GetPrefixSymbol< decltype(sound4cl_ExitProcess
      ) >("ExitProcess"            );
00450               fnStartUpdateThread       = GetPrefixSymbol< decltype(sound4cl_StartUpdateThread
      ) >("StartUpdateThread"      );
00451               fnStopUpdateThread        = GetPrefixSymbol< decltype(sound4cl_StopUpdateThread
      ) >("StopUpdateThread"       );
00452               fnWaitUpdateThreadReady   = GetPrefixSymbol< decltype(sound4cl_WaitUpdateThreadReady
      ) >("WaitUpdateThreadReady"  );
00453               fnProcessAudio            = GetPrefixSymbol< decltype(sound4cl_ProcessAudio
      ) >("ProcessAudio"           );
00454               fnProcessAudio_Planar     = GetPrefixSymbol< decltype(sound4cl_ProcessAudio_Planar
      ) >("ProcessAudio_Planar"    );
00455               fnGetBufferIn             = GetPrefixSymbol< decltype(sound4cl_GetBufferIn
      ) >("GetBufferIn"            );
00456               fnGetBufferOut            = GetPrefixSymbol< decltype(sound4cl_GetBufferOut
      ) >("GetBufferOut"           );
00457               fnGetEstimatedDelay       = GetPrefixSymbol< decltype(sound4cl_GetEstimatedDelay
      ) >("GetEstimatedDelay"      );
00458               fnGetFormatName           = GetPrefixSymbol< decltype(sound4cl_GetFormatName
      ) >("GetFormatName"          );
00459               fnGetFormatFromName       = GetPrefixSymbol< decltype(sound4cl_GetFormatFromName
      ) >("GetFormatFromName"      );
00460               fnGetBytesFromFormat      = GetPrefixSymbol< decltype(sound4cl_GetBytesFromFormat
      ) >("GetBytesFromFormat"     );
00461               fnGetMaxPacketFrame       = GetPrefixSymbol< decltype(sound4cl_GetMaxPacketFrame
      ) >("GetMaxPacketFrame"      );
00462               fnAddAudio                = GetPrefixSymbol< decltype(sound4cl_AddAudio
      ) >("AddAudio"               );
00463               fnAddPadAudio             = GetPrefixSymbol< decltype(sound4cl_AddPadAudio
      ) >("AddPadAudio"            );
00464               fnGetOutputCount          = GetPrefixSymbol< decltype(sound4cl_GetOutputCount
      ) >("GetOutputCount"         );
00465               fnGetAudio                = GetPrefixSymbol< decltype(sound4cl_GetAudio
      ) >("GetAudio"               );
00466               fnAudioConvertFrom        = GetPrefixSymbol< decltype(sound4cl_AudioConvertFrom
      ) >("AudioConvertFrom"       );
00467               fnAudioConvertTo          = GetPrefixSymbol< decltype(sound4cl_AudioConvertTo
      ) >("AudioConvertTo"         );
00468               fnStereoToMono            = GetPrefixSymbol< decltype(sound4cl_StereoToMono
      ) >("StereoToMono"           );
```

```
00469                 fnMonoToStereo            = GetPrefixSymbol< decltype(sound4cl_MonoToStereo
      ) >("MonoToStereo"           );
00470                 fnAudioMonoFromLiveStereo = GetPrefixSymbol<
      decltype(sound4cl_AudioMonoFromLiveStereo) >("AudioMonoFromLiveStereo");
00471                 fnAudioMonoToLiveStereo   = GetPrefixSymbol< decltype(sound4cl_AudioMonoToLiveStereo
      ) >("AudioMonoToLiveStereo"  );
00472                 fnNewClient               = GetPrefixSymbol< decltype(sound4cl_NewClient
      ) >("NewClient"              );
00473                 fnDeleteClient            = GetPrefixSymbol< decltype(sound4cl_DeleteClient
      ) >("DeleteClient"           );
00474                 fnProcessJson             = GetPrefixSymbol< decltype(sound4cl_ProcessJson
      ) >("ProcessJson"            );
00475                 fnFreeJsonAnswer          = GetPrefixSymbol< decltype(sound4cl_FreeJsonAnswer
      ) >("FreeJsonAnswer"         );
00476                 fnSaveState               = GetPrefixSymbol< decltype(sound4cl_SaveState
      ) >("SaveState"              );
00477             } catch (std::runtime_error& ) {
00478                 return false;
00479             }
00480             // Webserver : not for all projects
00481             try {
00482                 fnWebserver_tcp           = GetPrefixSymbol< decltype(sound4cl_Webserver_tcp
      ) >("Webserver_tcp"          );
00483                 fnWebserver_tcp2          = GetPrefixSymbol< decltype(sound4cl_Webserver_tcp2
      ) >("Webserver_tcp2"         );
00484                 fnWebserver               = GetPrefixSymbol< decltype(sound4cl_Webserver
      ) >("Webserver"              );
00485                 fnWebserver_Stop          = GetPrefixSymbol< decltype(sound4cl_Webserver_Stop
      ) >("Webserver_Stop"         );
00486                 fnWebserver_Status        = GetPrefixSymbol< decltype(sound4cl_Webserver_Status
      ) >("Webserver_Status"       );
00487             } catch (std::runtime_error& ) {
00488                 // Ignored, handler will take care
00489             }
00490             // Stereo/Mono conversion : must be present
00491             try {
00492                 // added 2023-03-22
00493                 fnStereoToMono_Planar     = GetPrefixSymbol< decltype(sound4cl_StereoToMono_Planar
      ) >("StereoToMono_Planar"    );
00494                 fnMonoToStereo_Planar     = GetPrefixSymbol< decltype(sound4cl_MonoToStereo_Planar
      ) >("MonoToStereo_Planar"    );
00495
00496             } catch (std::runtime_error& ) {
00497                 return false;
00498             }
00499             // C functions allowed to be missing
00500             try {
00501                 // added 2023-05-22
00502                 fnSetMetadata             = GetPrefixSymbol< decltype(sound4cl_SetMetadata
      ) >("SetMetadata"            );
00503                 fnGetMetadataInfos        = GetPrefixSymbol< decltype(sound4cl_GetMetadataInfos
      ) >("GetMetadataInfos"       );
00504                 fnFreeMetadataInfos       = GetPrefixSymbol< decltype(sound4cl_FreeMetadataInfos
      ) >("FreeMetadataInfos"      );
00505                 // added 2023-06-20
00506                 fnSetPresetManager            = GetPrefixSymbol< decltype(sound4cl_SetPresetManager
              ) >("SetPresetManager"             );
00507                 fnPresetManager_InformChange  = GetPrefixSymbol<
      decltype(sound4cl_PresetManager_InformChange     ) >("PresetManager_InformChange"    );
00508
00509                 // added 2023-09-21
00510                 fnGetPossibleChunkSizeInFrames = GetPrefixSymbol<
      decltype(sound4cl_GetPossibleChunkSizeInFrames  ) >("GetPossibleChunkSizeInFrames"  );
00511                 fnGetProcessChunkFrames        = GetPrefixSymbol<
      decltype(sound4cl_GetProcessChunkFrames         ) >("GetProcessChunkFrames"         );
00512                 fnInitProcess3                 = GetPrefixSymbol< decltype(sound4cl_InitProcess3
              ) >("InitProcess3"                 );
00513             }  catch (std::runtime_error& ) {
00514                 // Ignored, handler will take care
00515             }
00516
00517             // bus: not for all projects
00518             try {
00519                 // added 2023-06-26
00520                 fnNewBus                  = GetPrefixSymbol< decltype(sound4cl_NewBus
      ) >("NewBus"                 );
00521                 fnFreeBus                 = GetPrefixSymbol< decltype(sound4cl_FreeBus
      ) >("FreeBus"                );
00522                 fnSetInstanceBus          = GetPrefixSymbol< decltype(sound4cl_SetInstanceBus
      ) >("SetInstanceBus"         );
00523             }  catch (std::runtime_error& ) {
00524                 // Ignored, handler will take care
00525             }
00526
00527             // Other optional
00528             try {
00529                 // added 2024-02-19
```

```
00530                 fnSetMetadataMulti        = GetPrefixSymbol< decltype(sound4cl_SetMetadataMulti
      ) >("SetMetadataMulti"       );
00531
00532             }  catch (std::runtime_error& ) {
00533                 // Ignored, handler will take care
00534             }
00535
00536             // Other optional, web server only
00537             try {
00538                 // added 2024-05-23
00539                 fnWebserver_SetAppHealth    = GetPrefixSymbol<
      decltype(sound4cl_Webserver_SetAppHealth    ) >("Webserver_SetAppHealth"     );
00540                 fnWebserver_GetAppHealth    = GetPrefixSymbol<
      decltype(sound4cl_Webserver_GetAppHealth    ) >("Webserver_GetAppHealth"     );
00541                 fnWebserver_FreeString      = GetPrefixSymbol< decltype(sound4cl_Webserver_FreeString
          ) >("Webserver_FreeString"       );
00542             }  catch (std::runtime_error& ) {
00543                 // Ignored, handler will take care
00544             }
00545
00546             m_bOK=true;
00547             return true;
00548         }
00555         bool IsOk() const {
00556             return m_bOK;
00557         }
00558
00563         void AudioConvertFrom(const uint8_t *payload, float *output, size_t nSpl,
      sound4cl_SampleFormat fmt)
00564             { fnAudioConvertFrom(payload, output, nSpl, fmt); }
00565         void AudioConvertTo(const float *input, uint8_t *payload, size_t nSpl, sound4cl_SampleFormat
      fmt)
00566             { fnAudioConvertTo(input, payload, nSpl, fmt); }
00567         void StereoToMono(const float *input, float *output, size_t nSpl)
00568             { fnStereoToMono(input, output, nSpl); }
00569         void MonoToStereo(const float *input, float *output, size_t nSpl)
00570             { fnMonoToStereo(input, output, nSpl); }
00571         void StereoToMono_Planar(const float *inputL, const float *inputR, float *output, size_t nSpl)
00572             { fnStereoToMono_Planar(inputL, inputR, output, nSpl); }
00573         void MonoToStereo_Planar(const float *input, float *outputL, float *outputR, size_t nSpl)
00574             { fnMonoToStereo_Planar(input, outputL, outputR, nSpl); }
00575         void AudioMonoFromLiveStereo(const uint8_t *payload, float *output)
00576             { fnAudioMonoFromLiveStereo(payload, output); }
00577         void AudioMonoToLiveStereo(const float *input, uint8_t *payload)
00578             { fnAudioMonoToLiveStereo(input, payload); }
00579
00585         std::string GetVersion()
00586             { return std::string(fnGetVersion()); }
00587
00596         unsigned int GetChunkSizeInFrames()
00597             { return fnGetChunkSizeInFrames(); }
00598
00607         std::vector<unsigned int> GetPossibleChunkSizeInFrames()
00608             {
00609                 std::vector<unsigned int> list;
00610                 if (fnGetPossibleChunkSizeInFrames) {
00611                     for (unsigned int* src=fnGetPossibleChunkSizeInFrames();*src;src++) {
00612                         list.push_back(*src);
00613                     }
00614                 } else {
00615                     list.push_back(fnGetChunkSizeInFrames());
00616                 }
00617                 return list;
00618             }
00619
00625         unsigned int GetChannelCount()
00626             { return fnGetChannelCount(); }
00627
00635         unsigned int GetAudioInputCount()
00636             { return fnGetAudioInputCount(); }
00637
00645         unsigned int GetAudioOutputCount()
00646             { return fnGetAudioOutputCount(); }
00647
00655         unsigned int GetSampleRate()
00656             { return fnGetSampleRate(); }
00657
00664         std::string GetFormatName( const sound4cl_SampleFormat fmt)
00665             { return std::string(fnGetFormatName(fmt)); }
00666
00673         sound4cl_SampleFormat GetFormatFromName( const std::string& name)
00674             { return fnGetFormatFromName(name.c_str()); }
00675
00682         unsigned int GetBytesFromFormat( const sound4cl_SampleFormat fmt)
00683             { return fnGetBytesFromFormat(fmt); }
00684
00688         void SetLogSeverity(LogSeverity severity)
```

```
00689             { fnSetLogSeverity(severity); }
00690
00694         void SetLoggerCallback(log_cb_t cb)
00695             { _log_cb=cb; fnSetLoggerCallback(&_log_cb_c); }
00696     };
00697
00703     class CBus {
00704         public:
00705             CBus(CProcessor& dynlib)
00706                 : m_dynlib(dynlib)
00707                 {
00708                     if (m_dynlib.fnNewBus) {
00709                         m_bus=m_dynlib.fnNewBus();
00710                     }
00711                 }
00712             ~CBus()
00713                 {
00714                     if (m_bus && m_dynlib.fnFreeBus) {
00715                         m_dynlib.fnFreeBus(m_bus);
00716                     }
00717                     m_bus = nullptr;
00718                 }
00719             sound4cl_CBus *Get() const { return m_bus; }
00720         private:
00721             CProcessor& m_dynlib;
00722             sound4cl_CBus *m_bus;
00723     };
00724
00730     class CPresetLoader {
00731     public:
00732         CPresetLoader() = default;
00733         virtual ~CPresetLoader() = default;
00734
00735         virtual bool IsReadOnly() = 0;
00736         virtual bool Exists(const std::filesystem::path &name) = 0;
00737         virtual bool Remove(const std::filesystem::path &name) = 0;
00738         virtual bool Rename(const std::filesystem::path &from, const std::filesystem::path &to) = 0;
00739         virtual std::vector<std::filesystem::path> GetAll() = 0;
00740         virtual std::string Read(const std::filesystem::path &filename) = 0;
00741         virtual bool Write(const std::filesystem::path &filename, const std::string &content) =0;
00742     };
00743     using PresetChange_Kind = sound4cl_PresetChange_Kind;
00744
00750     class CInstance {
00751     public:
00755         CInstance(CProcessor& dynlib)
00756         : m_dynlib(dynlib)
00757             {
00758                 params = m_dynlib.fnNewParameters();
00759                 if (!params) throw std::bad_alloc();
00760             }
00761         // non-copyable
00762         CInstance( const CInstance& ) = delete;
00763         CInstance& operator=( const CInstance& ) = delete;
00767         ~CInstance()
00768             {
00769                 Stop();
00770                 if (params) m_dynlib.fnFreeParameters(params);
00771             }
00775        bool IsOk() const { return instance!=nullptr; }
00783         void SetParam(const std::string& name, const std::string& value)
00784             {
00785                 assert(params);
00786                 m_dynlib.fnSetParameter(params, name.c_str(), value.c_str());
00787             }
00793         std::string GetParam(const std::string& name)
00794             {
00795                 assert(params);
00796                 auto c_value=m_dynlib.fnGetParameter(params, name.c_str());
00797                 std::string ret(c_value);
00798                 m_dynlib.fnFreeParameterValue(c_value);
00799                 return ret;
00800             }
00801
00805         void SetBus(const CBus& bus) {
00806             assert(params);
00807             if (m_dynlib.fnSetInstanceBus) {
00808                 m_dynlib.fnSetInstanceBus(params,bus.Get());
00809             }
00810         }
00811
00821         void SetPresetManager(CPresetLoader *preset_manager)
00822             {
00823                 if (!preset_manager) return;
00824                 if (!m_dynlib.fnSetPresetManager) return;
00825                 assert(params);
00826                 m_dynlib.fnSetPresetManager(params,
```

```
00827                   sound4cl_custom_reader,
00828                   sound4cl_custom_reader_free,
00829                   sound4cl_custom_writer,
00830                   sound4cl_custom_exists,
00831                   sound4cl_custom_getall,
00832                   sound4cl_custom_getall_free,
00833                   sound4cl_custom_remove,
00834                   sound4cl_custom_rename,
00835                   preset_manager->IsReadOnly(),
00836                   preset_manager
00837               );
00838           }
00839
00854       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)
00855           {
00856               assert(!instance);
00857               std::string l_save_path;
00858               if (!save_path.empty()) {
00859                   l_save_path = save_path.u8string();
00860               }
00861               if (frames_per_chunk==0) {
00862                   frames_per_chunk=m_dynlib.GetChunkSizeInFrames();
00863               }
00864               if (m_dynlib.fnInitProcess3) {
00865                   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);
00866               } else if (frames_per_chunk!=m_dynlib.GetChunkSizeInFrames()) {
00867                   return false;
00868               } else {
00869                   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);
00870               }
00871               if (!instance) return false;
00872               if (!init_metadata.empty()) {
00873                   for (auto&& [key,value]: init_metadata) {
00874                       m_dynlib.fnSetMetadata(instance, key.c_str(), value.c_str());
00875                   }
00876                   init_metadata.clear();
00877               }
00878               update_thread = std::thread([this, json_port](){
00879                   OnUpdateThreadStart();
00880                   m_dynlib.fnStartUpdateThread(instance, json_port);
00881                   OnUpdateThreadStop();
00882               });
00883               int timeout=5000;
00884               #ifdef DEBUG
00885                   timeout=60*1000; // For debugger pauses
00886               #endif
00887               if (m_dynlib.fnWaitUpdateThreadReady(instance, timeout)<0) {
00888                   return false;
00889               }
00890               return true;
00891           }
00892
00900       unsigned int GetChunkFrames() {
00901           if (instance && m_dynlib.fnGetProcessChunkFrames) {
00902               return m_dynlib.fnGetProcessChunkFrames(instance);
00903           }
00904           return m_dynlib.fnGetChunkSizeInFrames();
00905       }
00906
00914       void PresetManager_InformChange(const std::filesystem::path& relative_path, PresetChange_Kind
      change_kind)
00915           {
00916               if (m_dynlib.fnPresetManager_InformChange) {
00917                   m_dynlib.fnPresetManager_InformChange(instance, relative_path.c_str(),
      (sound4cl_PresetChange_Kind) change_kind);
00918               }
00919           }
00920
00932       void SetMetadata(const std::string& key, const char* value)
00933           {
00934               if (m_dynlib.fnSetMetadata) {
00935                   if (instance) {
00936                       m_dynlib.fnSetMetadata(instance, key.c_str(), value);
00937                   } else {
00938                       init_metadata.push_back( {key, value} );
00939                   }
00940               }
00941           }
00942
00943
00953       void SetMetadataMulti(const std::unordered_map<std::string, const char*>& list)
00954           {
00955               if (instance) {
```

```
00956                     if (m_dynlib.fnSetMetadataMulti) {
00957                         typedef const char* pchar;
00958                         pchar* keyvalue=new pchar[2*list.size()+1];
00959                         size_t n=0;
00960                         for (auto&& [key,value]: list) {
00961                             keyvalue[2*n+0]=key.c_str();
00962                             keyvalue[2*n+1]=value;
00963                             n++;
00964                         }
00965                         keyvalue[2*n+0]=nullptr;
00966 
00967                         sound4cl_SetMetadataMulti(instance, keyvalue);
00968 
00969                         delete[] keyvalue;
00970                     } else {
00971                         for (auto&& [key,value]: list) {
00972                             SetMetadata(key, value);
00973                         }
00974                     }
00975                 } else {
00976                     for (auto&& [key,value]: list) {
00977                         if (value) {
00978                             init_metadata.push_back( {key, value} );
00979                         }
00980                     }
00981                 }
00982             }
00983 
00989         std::vector< std::tuple<std::string,std::string> > GetMetadataInfos()
00990             {
00991                 std::vector< std::tuple<std::string,std::string> > values;
00992                 if (m_dynlib.fnGetMetadataInfos) {
00993                     const char** c_values = m_dynlib.fnGetMetadataInfos(instance);
00994                     if (c_values) {
00995                         for (const char** c_value=c_values; *c_value; ) {
00996                             std::string key(*c_value);
00997                             c_value++;
00998                             if (*c_value) {
00999                                 std::string descr(*c_value);
01000                                 c_value++;
01001                                 values.push_back( {key,descr} );
01002                             }
01003                         }
01004                         m_dynlib.fnFreeMetadataInfos(instance, c_values);
01005                     }
01006                 }
01007                 return values;
01008             }
01009 
01010         // Can be overloaded, called at start of UpdateThread in the thread
01011         virtual void OnUpdateThreadStart() {}
01012         // Can be overloaded to call at end of UpdateThread in the thread
01013         virtual void OnUpdateThreadStop() {}
01014 
01020         bool StartWebServer(int http_port, int https_port=0)
01021             {
01022                 assert(instance);
01023                 assert(webserver==SOUND4_INVALID_WEBSERVER_ID);
01024                 if (!m_dynlib.fnWebserver) {
01025                     return false;
01026                 }
01027                 webserver = m_dynlib.fnWebserver(http_port,https_port,instance);
01028                 if (webserver==SOUND4_INVALID_WEBSERVER_ID) {
01029                     return false;
01030                 }
01031                 int web_status=m_dynlib.fnWebserver_Status(webserver);
01032                 if (http_port && (web_status & SOUND4_WEBSERVER_HTTP_OK) == 0) {
01033                     return false;
01034                 }
01035                 if (https_port && (web_status & SOUND4_WEBSERVER_HTTPS_OK) == 0) {
01036                     return false;
01037                 }
01038                 return true;
01039             }
01045         void StopWebServer()
01046             {
01047                 if (!m_dynlib.fnWebserver_Stop) return;
01048                 if (webserver != SOUND4_INVALID_WEBSERVER_ID) {
01049                     m_dynlib.fnWebserver_Stop(webserver, 1000);
01050                     webserver = SOUND4_INVALID_WEBSERVER_ID;
01051                 }
01052             }
01053 
01057         void SetWebServerAppHealth(int httpcode, const std::string& contenttype, const std::string&
      content)
01058             {
01059                 assert(instance);
```

```
01060                 if (!m_dynlib.fnWebserver_SetAppHealth) return;
01061                 m_dynlib.fnWebserver_SetAppHealth(instance, httpcode, contenttype.c_str(),
      content.c_str());
01062             }
01066         void GetWebServerAppHealth(int &httpcode, std::string& contenttype, std::string& content)
01067             {
01068                 assert(instance);
01069                 if (!m_dynlib.fnWebserver_GetAppHealth) return;
01070                 char* c_contenttype=nullptr;
01071                 char* c_content=nullptr;
01072                 m_dynlib.fnWebserver_GetAppHealth(instance, &httpcode, &c_contenttype, &c_content);
01073                 contenttype=c_contenttype;
01074                 content=c_content;
01075                 m_dynlib.fnWebserver_FreeString(c_contenttype);
01076                 m_dynlib.fnWebserver_FreeString(c_content);
01077             }
01084         int Terminate()
01085             {
01086                 if (!instance) return 0;
01087                 return m_dynlib.fnTerminateProcess(instance);
01088             }
01089
01095         void Stop()
01096             {
01097                 if (!instance) return;
01098                 StopWebServer();
01099                 if (update_thread.joinable()) {
01100                     m_dynlib.fnStopUpdateThread(instance); // this will exit the update thread
01101                     update_thread.join();
01102                 }
01103                 m_dynlib.fnExitProcess(instance);
01104                 instance=nullptr;
01105             }
01106
01112         unsigned int GetEstimatedDelay()
01113             {
01114                 assert(instance);
01115                 return m_dynlib.fnGetEstimatedDelay(instance);
01116             }
01117
01128         float* GetBufferIn()
01129             {
01130                 assert(instance);
01131                 return m_dynlib.fnGetBufferIn(instance);
01132
01133             }
01144         float* GetBufferOut()
01145             {
01146                 assert(instance);
01147                 return m_dynlib.fnGetBufferOut(instance);
01148             }
01154         void ProcessAudio()
01155             {
01156                 assert(instance);
01157                 m_dynlib.fnProcessAudio(instance, m_dynlib.fnGetBufferIn(instance),
      m_dynlib.fnGetBufferOut(instance));
01158             }
01172         // void ProcessAudio_Planar(const std::array< const std::array<const float,ChunkFrames>,
      InputChannels>& input, std::array< std::array<float,ChunkFrames>, OutputChannels>& output)
01173         void ProcessAudio_Planar(float const * const *input, float * const *output)
01174             {
01175                 assert(instance);
01176                 m_dynlib.fnProcessAudio_Planar(instance, input, output);
01177             }
01178
01190             template<typename T>
01191         std::vector<T> ProcessAnyAudio(const std::vector<T> input)
01192             {
01193                 assert(instance);
01194                 std::vector<T> output;
01195                 unsigned int out_offset=0;
01196                 unsigned int in_offset=0;
01197                 unsigned int todo = input.size();
01198                 while (todo>0) {
01199                     unsigned int left = AddAudio(&input[in_offset], todo);
01200                     unsigned int out_avail = m_dynlib.fnGetOutputCount(instance);
01201                     output.resize(out_offset + out_avail);
01202                     GetAudio(&output[out_offset], out_avail);
01203                     out_offset+=out_avail;
01204                     in_offset += todo-left;
01205                     todo=left;
01206                 }
01207                 return output;
01208             }
01209
01215         class CClient {
01216             public:
```

```
01217                 CClient(sound4cl_CInstance* instance, CProcessor& dynlib)
01218                 : m_dynlib(dynlib)
01219                     {
01220                         assert(instance);
01221                         client=m_dynlib.fnNewClient(instance);
01222                         if (!client) throw std::bad_alloc();
01223                     }
01224                 // non-copyable
01225                 CClient( const CInstance& ) = delete;
01226                 CClient& operator=( const CInstance& ) = delete;
01227                 ~CClient()
01228                     {
01229                         if (client) {
01230                             m_dynlib.fnDeleteClient(client);
01231                             client=nullptr;
01232                         }
01233                     }
01241                 std::string ProcessJson(const std::string &request, bool *NeedSave = nullptr)
01242                     {
01243                         assert(client);
01244                         int need_save=0;
01245                         const char *canswer = m_dynlib.fnProcessJson(client, request.c_str(),
    &need_save);
01246                         if (!canswer) return {};
01247                         std::string answer(canswer);
01248                         m_dynlib.fnFreeJsonAnswer (canswer);
01249                         if (NeedSave) {
01250                             *NeedSave=(need_save!=0);
01251                         }
01252                         return answer;
01253                     }
01254             private:
01255                 CProcessor& m_dynlib;
01256                 sound4cl_CClientInstance *client = nullptr;
01257         };
01265         std::shared_ptr<CClient> NewClient()
01266             {
01267                 assert(instance);
01268                 return std::make_shared<CClient>(instance, m_dynlib);
01269             }
01270
01271     protected:
01272             template<typename T>
01273         unsigned int AddAudio(const T* payload, unsigned int nFrame)
01274             {
01275                 assert(instance);
01276                 return m_dynlib.fnAddAudio(instance, reinterpret_cast<const T*>(payload), nFrame,
    helper::SampleFormat<T>::format);
01277             }
01278             template<typename T>
01279         unsigned int GetAudio(T* payload, unsigned int max_nFrame)
01280             {
01281                 assert(instance);
01282                 return m_dynlib.fnGetAudio(instance, reinterpret_cast<T*>(payload), max_nFrame,
    helper::SampleFormat<T>::format);
01283             }
01284
01285     private:
01286         CProcessor& m_dynlib;
01287         sound4cl_CParameters* params = nullptr;
01288         sound4cl_CInstance* instance = nullptr;
01289         std::thread update_thread;
01290         std::vector< std::pair<std::string,std::string> > init_metadata;
01291         uint64_t webserver = SOUND4_INVALID_WEBSERVER_ID;
01292     };
01293
01294 }; // namespace sound4
01295
01296
01297 // bridge C callbacks to CPresetLoader
01298 extern "C" {
01299     static char *sound4cl_custom_reader(const fs_char *filename, void* userdata) {
01300         sound4::CPresetLoader *preset_loader=reinterpret_cast<sound4::CPresetLoader*>(userdata);
01301         std::string content=preset_loader->Read(filename);
01302         return strdup(content.c_str());
01303     }
01304     static void sound4cl_custom_reader_free(char *content, void* userdata) {
01305         if (!content) return;
01306         free(content);
01307     }
01308     static int sound4cl_custom_writer(const fs_char *filename, const char *content, void* userdata) {
01309         sound4::CPresetLoader *preset_loader=reinterpret_cast<sound4::CPresetLoader*>(userdata);
01310         auto res=preset_loader->Write(filename,content);
01311         return res?0:-1;
01312     }
01313     static int sound4cl_custom_exists(const fs_char *filename, void* userdata) {
01314         sound4::CPresetLoader *preset_loader=reinterpret_cast<sound4::CPresetLoader*>(userdata);
```

```
01315         auto res=preset_loader->Exists(filename);
01316         return res?0:-1;
01317     }
01318     static fs_char** sound4cl_custom_getall(void* userdata) {
01319         sound4::CPresetLoader *preset_loader=reinterpret_cast<sound4::CPresetLoader*>(userdata);
01320         auto res=preset_loader->GetAll();
01321         fs_char**all=(fs_char**)malloc((res.size()+1)*sizeof(const fs_char*));
01322         for (size_t n=0;n<res.size();n++) {
01323             all[n]=fs_strdup(res[n].c_str());
01324         }
01325         all[res.size()]=nullptr;
01326         return all;
01327     }
01328     static void sound4cl_custom_getall_free(fs_char** all, void* userdata) {
01329         if (!all) return;
01330         for (fs_char** one=all;*one;one++) {
01331             free(*one);
01332         }
01333         free(all);
01334     }
01335     static int sound4cl_custom_remove(const fs_char *filename, void* userdata) {
01336         sound4::CPresetLoader *preset_loader=reinterpret_cast<sound4::CPresetLoader*>(userdata);
01337         auto res=preset_loader->Remove(filename);
01338         return res?0:-1;
01339     }
01340     static int sound4cl_custom_rename(const fs_char *from, const fs_char *to, void* userdata) {
01341         sound4::CPresetLoader *preset_loader=reinterpret_cast<sound4::CPresetLoader*>(userdata);
01342         auto res=preset_loader->Rename(from,to);
01343         return res?0:-1;
01344     }
01345
01346 };
```

11.17 sound4cl_cdef.h File Reference

C function signature definition for all sound4.cl libraries.

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

Classes

- struct sound4cl_SOUND4_ProcessInfo

Macros

- #define fs_strdup wcsdup

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

Typedefs

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

Enumerations

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

Functions

- const struct sound4cl_SOUND4_ProcessInfo * sound4cl_SOUND4_GetProcessInfo ()
- const char * sound4cl_GetVersion ()
- unsigned int sound4cl_GetChunkSizeInFrames ()
- unsigned int * sound4cl_GetPossibleChunkSizeInFrames ()
- unsigned int sound4cl_GetChannelCount ()
- unsigned int sound4cl_GetAudioInputCount ()
- unsigned int sound4cl_GetAudioOutputCount ()
- unsigned int sound4cl_GetSampleRate ()
- void sound4cl_SetLoggerCallback (sound4cl_loggerfn logger)
- void sound4cl_SetLogSeverity (enum sound4cl_LogSeverity severity)
- struct sound4cl_CParameters * sound4cl_NewParameters ()
- void sound4cl_FreeParameters (struct sound4cl_CParameters *params)
- void sound4cl_SetParameter (struct sound4cl_CParameters *params, const char *name, const char *value)
- const char * sound4cl_GetParameter (struct sound4cl_CParameters *params, const char *name)
- void sound4cl_FreeParameterValue (const char *value)
- struct sound4cl_CInstance * sound4cl_InitProcess (const char *LoginKey, const char *RadioName, const char *Access_Key_ID, const char *Access_Key_Secret, const char *save_path)
- struct sound4cl_CInstance * sound4cl_InitProcess2 (const char *LoginKey, const char *RadioName, const char *Access_Key_ID, const char *Access_Key_Secret, const char *save_path, const struct sound4cl_↩ CParameters *parameters)
- struct sound4cl_CInstance * sound4cl_InitProcess3 (const char *LoginKey, const char *RadioName, const char *Access_Key_ID, const char *Access_Key_Secret, const char *save_path, const struct sound4cl_↩ CParameters *parameters, unsigned int frames_per_chunk)
- unsigned int sound4cl_GetProcessChunkFrames (struct sound4cl_CInstance *instance)
- int sound4cl_TerminateProcess (struct sound4cl_CInstance *instance)
- void sound4cl_ExitProcess (struct sound4cl_CInstance *instance)
- struct sound4cl_CBus * sound4cl_NewBus ()
- void sound4cl_FreeBus (struct sound4cl_CBus *bus)
- void sound4cl_SetInstanceBus (struct sound4cl_CParameters *params, struct sound4cl_CBus *bus)
- void sound4cl_SetPresetManager (struct sound4cl_CParameters *params, sound4cl_storage_reader reader, sound4cl_storage_reader_free, sound4cl_storage_writer writer, sound4cl_storage_exists exists, sound4cl_storage_getall getall, sound4cl_storage_getall_free getall_free, sound4cl_storage_remove remove, sound4cl_storage_rename rename, int IsReadOnly, void *userdata)
- void sound4cl_PresetManager_InformChange (struct sound4cl_CInstance *instance, const fs_char *relative_path, enum sound4cl_PresetChange_Kind change_kind)
- void sound4cl_SetMetadata (struct sound4cl_CInstance *instance, const char *key, const char *value)
- void sound4cl_SetMetadataMulti (struct sound4cl_CInstance *instance, const char **keyvalue)
- const char ** sound4cl_GetMetadataInfos (struct sound4cl_CInstance *instance)
- void sound4cl_FreeMetadataInfos (struct sound4cl_CInstance *instance, const char **infos)

- void sound4cl_StartUpdateThread (struct sound4cl_CInstance *instance, unsigned int port)
- void sound4cl_StopUpdateThread (struct sound4cl_CInstance *instance)
- int sound4cl_WaitUpdateThreadReady (struct sound4cl_CInstance *instance, int milliseconds)
- void sound4cl_ProcessAudio (struct sound4cl_CInstance *instance, const float *input, float *output)
- void sound4cl_ProcessAudio_Planar (struct sound4cl_CInstance *instance, float const *const *input, float *const *output)
- float * sound4cl_GetBufferIn (struct sound4cl_CInstance *instance)
- float * sound4cl_GetBufferOut (struct sound4cl_CInstance *instance)
- unsigned int sound4cl_GetEstimatedDelay (struct sound4cl_CInstance *instance)
- const char * sound4cl_GetFormatName (const enum sound4cl_SampleFormat fmt)
- enum sound4cl_SampleFormat sound4cl_GetFormatFromName (const char *name)
- unsigned int sound4cl_GetBytesFromFormat (const enum sound4cl_SampleFormat fmt)
- unsigned int sound4cl_GetMaxPacketFrame (struct sound4cl_CInstance *instance)
- unsigned int sound4cl_AddAudio (struct sound4cl_CInstance *instance, const uint8_t *payload, unsigned int nFrame, enum sound4cl_SampleFormat fmt)
- unsigned int sound4cl_AddPadAudio (struct sound4cl_CInstance *instance)
- unsigned int sound4cl_GetOutputCount (struct sound4cl_CInstance *instance)
- unsigned int sound4cl_GetAudio (struct sound4cl_CInstance *instance, uint8_t *payload, unsigned int max↩ _nFrame, enum sound4cl_SampleFormat fmt)
- void sound4cl_AudioConvertFrom (const uint8_t *payload, float *output, size_t nSpl, enum sound4cl_SampleFormat fmt)
- void sound4cl_AudioConvertTo (const float *input, uint8_t *payload, size_t nSpl, enum sound4cl_SampleFormat fmt)
- void sound4cl_StereoToMono (const float *input, float *output, size_t nFrame)
- void sound4cl_MonoToStereo (const float *input, float *output, size_t nFrame)
- void sound4cl_StereoToMono_Planar (const float *inputL, const float *inputR, float *output, size_t nFrame)
- void sound4cl_MonoToStereo_Planar (const float *input, float *outputL, float *outputR, size_t nFrame)
- void sound4cl_AudioMonoFromLiveStereo (const uint8_t *payload, float *output)
- void sound4cl_AudioMonoToLiveStereo (const float *input, uint8_t *payload)
- struct sound4cl_CClientInstance * sound4cl_NewClient (struct sound4cl_CInstance *instance)
- void sound4cl_DeleteClient (struct sound4cl_CClientInstance *client)
- const char * sound4cl_ProcessJson (struct sound4cl_CClientInstance *client, const char *json_str, int *need_save)
- void sound4cl_FreeJsonAnswer (const char *json_str)
- int sound4cl_SaveState (struct sound4cl_CInstance *instance)
- uint64_t sound4cl_Webserver_tcp (unsigned int listenport, unsigned int listenport_secure, const char *socket_ip, unsigned int socket_port)
- uint64_t sound4cl_Webserver_tcp2 (unsigned int listenport, unsigned int listenport_secure, const char *socket_ip, unsigned int socket_port, const struct sound4cl_CParameters *parameters)
- uint64_t sound4cl_Webserver (unsigned int listenport, unsigned int listenport_secure, struct sound4cl_↩ CInstance *instance)
- int sound4cl_Webserver_Stop (uint64_t id, int timeout_ms)
- int sound4cl_Webserver_Status (uint64_t id)
- void sound4cl_Webserver_SetAppHealth (struct sound4cl_CInstance *instance, int httpcode, const char *contenttype, const char *content)
- void sound4cl_Webserver_GetAppHealth (struct sound4cl_CInstance *instance, int *httpcode, char **contenttype, char **content)
- void sound4cl_Webserver_FreeString (char *str)

11.17.1 Macro Definition Documentation

11.17.1.1 fs_strdup

```
#define fs_strdup wcsdup
```

Definition at line 17 of file sound4cl_cdef.h.

11.17.1.2 SOUND4_INVALID_WEBSERVER_ID

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

Definition at line 62 of file sound4cl_cdef.h.

11.17.1.3 SOUND4_WEBSERVER_HTTP_OK

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

Definition at line 63 of file sound4cl_cdef.h.

11.17.1.4 SOUND4_WEBSERVER_HTTPS_OK

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

Definition at line 64 of file sound4cl_cdef.h.

11.17.2 Typedef Documentation

11.17.2.1 fs_char

```
typedef wchar_t fs_char
```

Definition at line 18 of file sound4cl_cdef.h.

11.17.2.2 sound4cl_loggerfn

```
typedef void(* sound4cl_loggerfn) (enum sound4cl_LogSeverity, const char *)
```

Definition at line 102 of file sound4cl_cdef.h.

11.17.2.3 sound4cl_storage_exists

```
typedef int(* sound4cl_storage_exists) (const fs_char *filename, void *userdata)
```

Definition at line 143 of file sound4cl_cdef.h.

11.17.2.4 sound4cl_storage_getall

```
typedef fs_char**(* sound4cl_storage_getall) (void *userdata)
```

Definition at line 144 of file sound4cl_cdef.h.

11.17.2.5 sound4cl_storage_getall_free

```
typedef void(* sound4cl_storage_getall_free) (fs_char **all, void *userdata)
```

Definition at line 145 of file sound4cl_cdef.h.

11.17.2.6 sound4cl_storage_reader

```
typedef char*(* sound4cl_storage_reader) (const fs_char *filename, void *userdata)
```

Definition at line 140 of file sound4cl_cdef.h.

11.17.2.7 sound4cl_storage_reader_free

```
typedef void(* sound4cl_storage_reader_free) (char *content, void *userdata)
```

Definition at line 141 of file sound4cl_cdef.h.

11.17.2.8 sound4cl_storage_remove

```
typedef int(* sound4cl_storage_remove) (const fs_char *filename, void *userdata)
```

Definition at line 146 of file sound4cl_cdef.h.

11.17.2.9 sound4cl_storage_rename

```
typedef int(* sound4cl_storage_rename) (const fs_char *from, const fs_char *to, void *userdata)
```

Definition at line 147 of file sound4cl_cdef.h.

11.17.2.10 sound4cl_storage_writer

```
typedef int(* sound4cl_storage_writer) (const fs_char *filename, const char *content, void
*userdata)
```

Definition at line 142 of file sound4cl_cdef.h.

11.17.3 Enumeration Type Documentation

11.17.3.1 sound4cl_LogSeverity

```
enum sound4cl_LogSeverity
```

Enumerator

sound4cl_none	
sound4cl_fatal	
sound4cl_error	
sound4cl_warning	
sound4cl_info	
sound4cl_verbose	
sound4cl_verbose2	
sound4cl_verbose3	
sound4cl_verbose4	
sound4cl_verbose5	

Definition at line 41 of file sound4cl_cdef.h.

11.17.3.2 sound4cl_PresetChange_Kind

```
enum sound4cl_PresetChange_Kind
```

Enumerator

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

Definition at line 57 of file sound4cl_cdef.h.

11.17.3.3 sound4cl_SampleFormat

```
enum sound4cl_SampleFormat
```

Supported formats for audio I/O

Enumerator

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

Definition at line 69 of file sound4cl_cdef.h.

11.17.4 Function Documentation

11.17.4.1 sound4cl_AddAudio()

```
unsigned int sound4cl_AddAudio (
            struct sound4cl_CInstance * instance,
            const uint8_t * payload,
            unsigned int nFrame,
            enum sound4cl_SampleFormat fmt )
```

11.17.4.2 sound4cl_AddPadAudio()

```
unsigned int sound4cl_AddPadAudio (
            struct sound4cl_CInstance * instance )
```

11.17.4.3 sound4cl_AudioConvertFrom()

```
void sound4cl_AudioConvertFrom (
            const uint8_t * payload,
            float * output,
            size_t nSpl,
            enum sound4cl_SampleFormat fmt )
```

11.17.4.4 sound4cl_AudioConvertTo()

```
void sound4cl_AudioConvertTo (
            const float * input,
            uint8_t * payload,
            size_t nSpl,
            enum sound4cl_SampleFormat fmt )
```

11.17.4.5 sound4cl_AudioMonoFromLiveStereo()

```
void sound4cl_AudioMonoFromLiveStereo (
            const uint8_t * payload,
            float * output )
```

11.17.4.6 sound4cl_AudioMonoToLiveStereo()

```
void sound4cl_AudioMonoToLiveStereo (
            const float * input,
            uint8_t * payload )
```

11.17.4.7 sound4cl_DeleteClient()

```
void sound4cl_DeleteClient (
            struct sound4cl_CClientInstance * client )
```

11.17.4.8 sound4cl_ExitProcess()

```
void sound4cl_ExitProcess (
            struct sound4cl_CInstance * instance )
```

11.17.4.9 sound4cl_FreeBus()

```
void sound4cl_FreeBus (
            struct sound4cl_CBus * bus )
```

11.17.4.10 sound4cl_FreeJsonAnswer()

```
void sound4cl_FreeJsonAnswer (
            const char * json_str )
```

11.17.4.11 sound4cl_FreeMetadataInfos()

```
void sound4cl_FreeMetadataInfos (
            struct sound4cl_CInstance * instance,
            const char ** infos )
```

11.17.4.12 sound4cl_FreeParameters()

```
void sound4cl_FreeParameters (
            struct sound4cl_CParameters * params )
```

11.17.4.13 sound4cl_FreeParameterValue()

```
void sound4cl_FreeParameterValue (
            const char * value )
```

11.17.4.14 sound4cl_GetAudio()

```
unsigned int sound4cl_GetAudio (
            struct sound4cl_CInstance * instance,
            uint8_t * payload,
            unsigned int max_nFrame,
            enum sound4cl_SampleFormat fmt )
```

11.17.4.15 sound4cl_GetAudioInputCount()

```
unsigned int sound4cl_GetAudioInputCount ( )
```

11.17.4.16 sound4cl_GetAudioOutputCount()

```
unsigned int sound4cl_GetAudioOutputCount ( )
```

11.17.4.17 sound4cl_GetBufferIn()

```
float* sound4cl_GetBufferIn (
            struct sound4cl_CInstance * instance )
```

11.17.4.18 sound4cl_GetBufferOut()

```
float* sound4cl_GetBufferOut (
            struct sound4cl_CInstance * instance )
```

11.17.4.19 sound4cl_GetBytesFromFormat()

```
unsigned int sound4cl_GetBytesFromFormat (
            const enum sound4cl_SampleFormat fmt )
```

11.17.4.20 sound4cl_GetChannelCount()

```
unsigned int sound4cl_GetChannelCount ( )
```

11.17.4.21 sound4cl_GetChunkSizeInFrames()

```
unsigned int sound4cl_GetChunkSizeInFrames ( )
```

11.17.4.22 sound4cl_GetEstimatedDelay()

```
unsigned int sound4cl_GetEstimatedDelay (
            struct sound4cl_CInstance * instance )
```

11.17.4.23 sound4cl_GetFormatFromName()

```
enum sound4cl_SampleFormat sound4cl_GetFormatFromName (
            const char * name )
```

11.17.4.24 sound4cl_GetFormatName()

```
const char* sound4cl_GetFormatName (
            const enum sound4cl_SampleFormat fmt )
```

11.17.4.25 sound4cl_GetMaxPacketFrame()

```
unsigned int sound4cl_GetMaxPacketFrame (
            struct sound4cl_CInstance * instance )
```

11.17.4.26 sound4cl_GetMetadataInfos()

```
const char** sound4cl_GetMetadataInfos (
            struct sound4cl_CInstance * instance )
```

11.17.4.27 sound4cl_GetOutputCount()

```
unsigned int sound4cl_GetOutputCount (
            struct sound4cl_CInstance * instance )
```

11.17.4.28 sound4cl_GetParameter()

```
const char* sound4cl_GetParameter (
            struct sound4cl_CParameters * params,
            const char * name )
```

11.17.4.29 sound4cl_GetPossibleChunkSizeInFrames()

```
unsigned int* sound4cl_GetPossibleChunkSizeInFrames ( )
```

11.17.4.30 sound4cl_GetProcessChunkFrames()

```
unsigned int sound4cl_GetProcessChunkFrames (
            struct sound4cl_CInstance * instance )
```

11.17.4.31 sound4cl_GetSampleRate()

```
unsigned int sound4cl_GetSampleRate ( )
```

11.17.4.32 sound4cl_GetVersion()

```
const char* sound4cl_GetVersion ( )
```

11.17.4.33 sound4cl_InitProcess()

```
struct sound4cl_CInstance* sound4cl_InitProcess (
            const char * LoginKey,
            const char * RadioName,
            const char * Access_Key_ID,
            const char * Access_Key_Secret,
            const char * save_path )
```

11.17.4.34 sound4cl_InitProcess2()

```
struct sound4cl_CInstance* sound4cl_InitProcess2 (
            const char * LoginKey,
            const char * RadioName,
            const char * Access_Key_ID,
            const char * Access_Key_Secret,
            const char * save_path,
            const struct sound4cl_CParameters * parameters )
```

11.17.4.35 sound4cl_InitProcess3()

```
struct sound4cl_CInstance* sound4cl_InitProcess3 (
            const char * LoginKey,
            const char * RadioName,
            const char * Access_Key_ID,
            const char * Access_Key_Secret,
            const char * save_path,
            const struct sound4cl_CParameters * parameters,
            unsigned int frames_per_chunk )
```

11.17.4.36 sound4cl_MonoToStereo()

```
void sound4cl_MonoToStereo (
            const float * input,
            float * output,
            size_t nFrame )
```

11.17.4.37 sound4cl_MonoToStereo_Planar()

```
void sound4cl_MonoToStereo_Planar (
            const float * input,
            float * outputL,
            float * outputR,
            size_t nFrame )
```

11.17.4.38 sound4cl_NewBus()

```
struct sound4cl_CBus* sound4cl_NewBus ( )
```

11.17.4.39 sound4cl_NewClient()

```
struct sound4cl_CClientInstance* sound4cl_NewClient (
            struct sound4cl_CInstance * instance )
```

11.17.4.40 sound4cl_NewParameters()

```
struct sound4cl_CParameters* sound4cl_NewParameters ( )
```

11.17.4.41 sound4cl_PresetManager_InformChange()

```
void sound4cl_PresetManager_InformChange (
            struct sound4cl_CInstance * instance,
            const fs_char * relative_path,
            enum sound4cl_PresetChange_Kind change_kind )
```

11.17.4.42 sound4cl_ProcessAudio()

```
void sound4cl_ProcessAudio (
            struct sound4cl_CInstance * instance,
            const float * input,
            float * output )
```

11.17.4.43 sound4cl_ProcessAudio_Planar()

```
void sound4cl_ProcessAudio_Planar (
            struct sound4cl_CInstance * instance,
            float const *const * input,
            float *const * output )
```

11.17.4.44 sound4cl_ProcessJson()

```
const char* sound4cl_ProcessJson (
            struct sound4cl_CClientInstance * client,
            const char * json_str,
            int * need_save )
```

11.17.4.45 sound4cl_SaveState()

```
int sound4cl_SaveState (
            struct sound4cl_CInstance * instance )
```

11.17.4.46 sound4cl_SetInstanceBus()

```
void sound4cl_SetInstanceBus (
            struct sound4cl_CParameters * params,
            struct sound4cl_CBus * bus )
```

11.17.4.47 sound4cl_SetLoggerCallback()

```
void sound4cl_SetLoggerCallback (
            sound4cl_loggerfn logger )
```

11.17.4.48 sound4cl_SetLogSeverity()

```
void sound4cl_SetLogSeverity (
            enum sound4cl_LogSeverity severity )
```

11.17.4.49 sound4cl_SetMetadata()

```
void sound4cl_SetMetadata (
            struct sound4cl_CInstance * instance,
            const char * key,
            const char * value )
```

11.17.4.50 sound4cl_SetMetadataMulti()

```
void sound4cl_SetMetadataMulti (
            struct sound4cl_CInstance * instance,
            const char ** keyvalue )
```

11.17.4.51 sound4cl_SetParameter()

```
void sound4cl_SetParameter (
            struct sound4cl_CParameters * params,
            const char * name,
            const char * value )
```

11.17.4.52 sound4cl_SetPresetManager()

```
void sound4cl_SetPresetManager (
            struct sound4cl_CParameters * params,
            sound4cl_storage_reader reader,
            sound4cl_storage_reader_free ,
            sound4cl_storage_writer writer,
            sound4cl_storage_exists exists,
            sound4cl_storage_getall getall,
            sound4cl_storage_getall_free getall_free,
            sound4cl_storage_remove remove,
            sound4cl_storage_rename rename,
            int IsReadOnly,
            void * userdata )
```

11.17.4.53 sound4cl_SOUND4_GetProcessInfo()

```
const struct sound4cl_SOUND4_ProcessInfo* sound4cl_SOUND4_GetProcessInfo ( )
```

11.17.4.54 sound4cl_StartUpdateThread()

```
void sound4cl_StartUpdateThread (
            struct sound4cl_CInstance * instance,
            unsigned int port )
```

11.17.4.55 sound4cl_StereoToMono()

```
void sound4cl_StereoToMono (
            const float * input,
            float * output,
            size_t nFrame )
```

11.17.4.56 sound4cl_StereoToMono_Planar()

```
void sound4cl_StereoToMono_Planar (
            const float * inputL,
            const float * inputR,
            float * output,
            size_t nFrame )
```

11.17.4.57 sound4cl_StopUpdateThread()

```
void sound4cl_StopUpdateThread (
            struct sound4cl_CInstance * instance )
```

11.17.4.58 sound4cl_TerminateProcess()

```
int sound4cl_TerminateProcess (
            struct sound4cl_CInstance * instance )
```

11.17.4.59 sound4cl_WaitUpdateThreadReady()

```
int sound4cl_WaitUpdateThreadReady (
            struct sound4cl_CInstance * instance,
            int milliseconds )
```

11.17.4.60 sound4cl_Webserver()

```
uint64_t sound4cl_Webserver (
            unsigned int listenport,
            unsigned int listenport_secure,
            struct sound4cl_CInstance * instance )
```

11.17.4.61 sound4cl_Webserver_FreeString()

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

11.17.4.62 sound4cl_Webserver_GetAppHealth()

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

11.17.4.63 sound4cl_Webserver_SetAppHealth()

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

11.17.4.64 sound4cl_Webserver_Status()

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

11.17.4.65 sound4cl_Webserver_Stop()

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

11.17.4.66 sound4cl_Webserver_tcp()

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

11.17.4.67 sound4cl_Webserver_tcp2()

```
uint64_t sound4cl_Webserver_tcp2 (
            unsigned int listenport,
            unsigned int listenport_secure,
            const char * socket_ip,
            unsigned int socket_port,
            const struct sound4cl_CParameters * parameters )
```

11.18 sound4cl_cdef.h

```
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::CBus, 85
~CClient
sound4::CInstance::CClient, 87
sound4::cloudx1::CInstance::CClient, 89
sound4::cloudx1::dyn::CInstance::CClient, 91
~CDynLoader
sound4::cloudx1::dyn::helper::CDynLoader, 106
sound4::helper::CDynLoader, 109
~CInstance
sound4::CInstance, 113
sound4::cloudx1::CInstance, 124
sound4::cloudx1::dyn::CInstance, 135
~CPresetLoader
sound4::cloudx1::CPresetLoader, 146
sound4::cloudx1::dyn::CPresetLoader, 148
sound4::CPresetLoader, 150
~MyPresetManager
MyPresetManager, 176

AddAudio
sound4::CInstance, 113
sound4::cloudx1::CInstance, 124
sound4::cloudx1::dyn::CDynLib, 94
sound4::cloudx1::dyn::CInstance, 135
AddPadAudio
sound4::cloudx1::dyn::CDynLib, 95
Audio convert helpers, 48
cloudx1_AudioConvertFrom, 48
cloudx1_AudioConvertTo, 49
cloudx1_AudioMonoFromLiveStereo, 49
cloudx1_AudioMonoToLiveStereo, 49
cloudx1_MonoToStereo, 50
cloudx1_MonoToStereo_Planar, 50
cloudx1_StereoToMono, 51
cloudx1_StereoToMono_Planar, 51
Audio managing (any audio), 43
cloudx1_AddAudio, 45
cloudx1_AddPadAudio, 45
cloudx1_GetAudio, 45
cloudx1_GetBytesFromFormat, 46
cloudx1_GetFormatFromName, 46
cloudx1_GetFormatName, 47
cloudx1_GetMaxPacketFrame, 47
cloudx1_GetOutputCount, 47
cloudx1_SampleFormat, 44
F32_BE, 44
F32_LE, 44
F32_NATIVE, 44
INVALID_FORMAT, 44
S16_BE, 44
S16_LE, 44
S16_NATIVE, 44
S24_BE, 44
S24_LE, 44
S24_NATIVE, 44
S32_BE, 44
S32_LE, 44
S32_NATIVE, 44
AudioConvertFrom
sound4::cloudx1::dyn::CDynLib, 95
sound4::CProcessor, 154
AudioConvertTo
sound4::cloudx1::dyn::CDynLib, 95
sound4::CProcessor, 154
AudioMonoFromLiveStereo
sound4::cloudx1::dyn::CDynLib, 95
sound4::CProcessor, 155
AudioMonoToLiveStereo
sound4::cloudx1::dyn::CDynLib, 95
sound4::CProcessor, 155

CBus
sound4::CBus, 85
sound4::CProcessor, 161
CClient
sound4::CInstance::CClient, 87
sound4::cloudx1::CInstance::CClient, 89
sound4::cloudx1::dyn::CInstance::CClient, 90, 91
CDynLib
sound4::cloudx1::dyn::CDynLib, 94
CDynLoader
sound4::cloudx1::dyn::helper::CDynLoader, 106
sound4::helper::CDynLoader, 109
change_kind_created
Custom preset managagement, 36
sound4cl_cdef.h, 260
change_kind_deleted
Custom preset managagement, 36
sound4cl_cdef.h, 260
change_kind_modified
Custom preset managagement, 36
sound4cl_cdef.h, 260
CHANNEL_COUNT
sound4.x1.cloud-pipe.cpp, 200
CHUNK_SIZE
sound4.x1.cloud-pipe.cpp, 201
ChunkFrames
sound4::cloudx1, 78
sound4::cloudx1::dyn, 81

CInstance
sound4::CInstance, 113
sound4::cloudx1::CInstance, 124
sound4::cloudx1::dyn::CInstance, 134, 135
sound4::CProcessor, 161
Client's JSon config interfacing, 52
cloudx1_DeleteClient, 52
cloudx1_FreeJsonAnswer, 52
cloudx1_NewClient, 53
cloudx1_ProcessJson, 53
cloudx1_SaveState, 54
Close
sound4::cloudx1::dyn::helper::CDynLoader, 106
sound4::helper::CDynLoader, 109
cloudx1_AddAudio
Audio managing (any audio), 45
cloudx1_AddPadAudio
Audio managing (any audio), 45
CLOUDX1_AUDIO_INPUT_COUNT
Global setup, 21
CLOUDX1_AUDIO_OUTPUT_COUNT
Global setup, 21
cloudx1_AudioConvertFrom
Audio convert helpers, 48
cloudx1_AudioConvertTo
Audio convert helpers, 49
CLOUDX1_AUDIOFRAME_COUNT
Global setup, 21
cloudx1_AudioMonoFromLiveStereo
Audio convert helpers, 49
cloudx1_AudioMonoToLiveStereo
Audio convert helpers, 49
cloudx1_CClientInstance, 144
CLOUDX1_CHANNEL_COUNT
Global setup, 21
cloudx1_CInstance, 144
cloudx1_CParameters, 145
CLOUDX1_DECL
sound4.x1.cloud.h, 208
cloudx1_DeleteClient
Client's JSon config interfacing, 52
cloudx1_ExitProcess
Process creation/destruction, 27
cloudx1_FreeJsonAnswer
Client's JSon config interfacing, 52
cloudx1_FreeMetadataInfos
Metadata Management, 37
cloudx1_FreeParameters
Process creation/destruction, 27
cloudx1_FreeParameterValue
Process creation/destruction, 27
cloudx1_GetAudio
Audio managing (any audio), 45
cloudx1_GetAudioInputCount
Global setup, 23
cloudx1_GetAudioOutputCount
Global setup, 23
cloudx1_GetBufferIn
Direct processing, 41
cloudx1_GetBufferOut
Direct processing, 41
cloudx1_GetBytesFromFormat
Audio managing (any audio), 46
cloudx1_GetChannelCount
Global setup, 23
cloudx1_GetChunkSizeInFrames
Global setup, 24
cloudx1_GetEstimatedDelay
Direct processing, 42
cloudx1_GetFormatFromName
Audio managing (any audio), 46
cloudx1_GetFormatName
Audio managing (any audio), 47
cloudx1_GetMaxPacketFrame
Audio managing (any audio), 47
cloudx1_GetMetadataInfos
Metadata Management, 38
cloudx1_GetOutputCount
Audio managing (any audio), 47
cloudx1_GetParameter
Process creation/destruction, 27
cloudx1_GetPossibleChunkSizeInFrames
Global setup, 24
cloudx1_GetProcessChunkFrames
Process creation/destruction, 28
cloudx1_GetSampleRate
Global setup, 24
cloudx1_GetVersion
Global setup, 25
CLOUDX1_HAS_CLOUDBUS
sound4.x1.cloud.h, 208
CLOUDX1_HAS_WEBSERVER
sound4.x1.cloud.h, 208
cloudx1_InitProcess
Process creation/destruction, 28
cloudx1_InitProcess2
Process creation/destruction, 28
cloudx1_InitProcess3
Process creation/destruction, 29
cloudx1_loggerfn
Global setup, 22
cloudx1_LogSeverity
Global setup, 22
cloudx1_MonoToStereo
Audio convert helpers, 50
cloudx1_MonoToStereo_Planar
Audio convert helpers, 50
cloudx1_NewClient
Client's JSon config interfacing, 53
cloudx1_NewParameters
Process creation/destruction, 30
cloudx1_PresetChange_Kind
Custom preset managagement, 35
cloudx1_PresetManager_InformChange
Custom preset managagement, 36
cloudx1_ProcessAudio

Direct processing, 42
cloudx1_ProcessAudio_Planar
Direct processing, 43
cloudx1_ProcessJson
Client's JSon config interfacing, 53
CLOUDX1_SAMPLE_RATE
Global setup, 22
cloudx1_SampleFormat
Audio managing (any audio), 44
cloudx1_SaveState
Client's JSon config interfacing, 54
cloudx1_SetLoggerCallback
Global setup, 25
cloudx1_SetLogSeverity
Global setup, 25
cloudx1_SetMetadata
Metadata Management, 38
cloudx1_SetMetadataMulti
Metadata Management, 39
cloudx1_SetParameter
Process creation/destruction, 30
cloudx1_SetPresetManager
Custom preset managagement, 36
cloudx1_StartUpdateThread
Update Management, 39
cloudx1_StereoToMono
Audio convert helpers, 51
cloudx1_StereoToMono_Planar
Audio convert helpers, 51
cloudx1_StopUpdateThread
Update Management, 40
cloudx1_storage_exists
Custom preset managagement, 32
cloudx1_storage_getall
Custom preset managagement, 33
cloudx1_storage_getall_free
Custom preset managagement, 33
cloudx1_storage_reader
Custom preset managagement, 33
cloudx1_storage_reader_free
Custom preset managagement, 34
cloudx1_storage_remove
Custom preset managagement, 34
cloudx1_storage_rename
Custom preset managagement, 34
cloudx1_storage_writer
Custom preset managagement, 35
cloudx1_TerminateProcess
Process creation/destruction, 31
cloudx1_WaitUpdateThreadReady
Update Management, 40
cloudx1_Webserver
Web server, 62
cloudx1_Webserver_FreeString
Web server, 63
cloudx1_Webserver_GetAppHealth
Web server, 63
cloudx1_Webserver_SetAppHealth
Web server, 63
cloudx1_Webserver_Status
Web server, 64
cloudx1_Webserver_Stop
Web server, 64
cloudx1_Webserver_tcp
Web server, 65
cloudx1_Webserver_tcp2
Web server, 65
CPresetLoader
sound4::cloudx1::CPresetLoader, 146
sound4::cloudx1::dyn::CPresetLoader, 148
sound4::CPresetLoader, 150
CProcessor
sound4::CProcessor, 154
Create
sound4::CInstance, 113
sound4::cloudx1::CInstance, 124
sound4::cloudx1::dyn::CInstance, 135
Custom preset managagement, 31
change_kind_created, 36
change_kind_deleted, 36
change_kind_modified, 36
cloudx1_PresetChange_Kind, 35
cloudx1_PresetManager_InformChange, 36
cloudx1_SetPresetManager, 36
cloudx1_storage_exists, 32
cloudx1_storage_getall, 33
cloudx1_storage_getall_free, 33
cloudx1_storage_reader, 33
cloudx1_storage_reader_free, 34
cloudx1_storage_remove, 34
cloudx1_storage_rename, 34
cloudx1_storage_writer, 35
fs_char, 35

DeleteClient
sound4::cloudx1::dyn::CDynLib, 95
Direct processing, 40
cloudx1_GetBufferIn, 41
cloudx1_GetBufferOut, 41
cloudx1_GetEstimatedDelay, 42
cloudx1_ProcessAudio, 42
cloudx1_ProcessAudio_Planar, 43
dyn, 73
DynFunc_t
sound4::cloudx1::dyn::helper, 82
sound4::helper, 84
DynFuncHolder
sound4::cloudx1::dyn::helper::DynFuncHolder< T >, 173
sound4::helper::DynFuncHolder< T >, 174
DynLib_t
sound4::cloudx1::dyn::helper, 82
sound4::helper, 84

Environment variables and parameters, 19
error
Global setup, 23

sound4, 75
Exists
MyPresetManager, 177
sound4::cloudx1::CPresetLoader, 146
sound4::cloudx1::dyn::CPresetLoader, 148
sound4::CPresetLoader, 150
ExitProcess
sound4::cloudx1::dyn::CDynLib, 96

F32_BE
Audio managing (any audio), 44
sound4cl_cdef.h, 261
F32_LE
Audio managing (any audio), 44
sound4cl_cdef.h, 261
F32_NATIVE
Audio managing (any audio), 44
sound4cl_cdef.h, 261
fatal
Global setup, 23
sound4, 75
fnAddAudio
sound4::CProcessor, 161
fnAddPadAudio
sound4::CProcessor, 161
fnAudioConvertFrom
sound4::CProcessor, 161
fnAudioConvertTo
sound4::CProcessor, 162
fnAudioMonoFromLiveStereo
sound4::CProcessor, 162
fnAudioMonoToLiveStereo
sound4::CProcessor, 162
fnDeleteClient
sound4::CProcessor, 162
fnExitProcess
sound4::CProcessor, 162
fnFreeBus
sound4::CProcessor, 162
fnFreeJsonAnswer
sound4::CProcessor, 163
fnFreeMetadataInfos
sound4::CProcessor, 163
fnFreeParameters
sound4::CProcessor, 163
fnFreeParameterValue
sound4::CProcessor, 163
fnGetAudio
sound4::CProcessor, 163
fnGetAudioInputCount
sound4::CProcessor, 163
fnGetAudioOutputCount
sound4::CProcessor, 164
fnGetBufferIn
sound4::CProcessor, 164
fnGetBufferOut
sound4::CProcessor, 164
fnGetBytesFromFormat
sound4::CProcessor, 164
fnGetChannelCount
sound4::CProcessor, 164
fnGetChunkSizeInFrames
sound4::CProcessor, 164
fnGetEstimatedDelay
sound4::CProcessor, 165
fnGetFormatFromName
sound4::CProcessor, 165
fnGetFormatName
sound4::CProcessor, 165
fnGetMaxPacketFrame
sound4::CProcessor, 165
fnGetMetadataInfos
sound4::CProcessor, 165
fnGetOutputCount
sound4::CProcessor, 165
fnGetParameter
sound4::CProcessor, 166
fnGetPossibleChunkSizeInFrames
sound4::CProcessor, 166
fnGetProcessChunkFrames
sound4::CProcessor, 166
fnGetSampleRate
sound4::CProcessor, 166
fnGetVersion
sound4::CProcessor, 166
fnInitProcess
sound4::CProcessor, 166
fnInitProcess2
sound4::CProcessor, 167
fnInitProcess3
sound4::CProcessor, 167
fnMonoToStereo
sound4::CProcessor, 167
fnMonoToStereo_Planar
sound4::CProcessor, 167
fnNewBus
sound4::CProcessor, 167
fnNewClient
sound4::CProcessor, 167
fnNewParameters
sound4::CProcessor, 168
fnPresetManager_InformChange
sound4::CProcessor, 168
fnProcessAudio
sound4::CProcessor, 168
fnProcessAudio_Planar
sound4::CProcessor, 168
fnProcessJson
sound4::CProcessor, 168
fnSaveState
sound4::CProcessor, 168
fnSetInstanceBus
sound4::CProcessor, 169
fnSetLoggerCallback
sound4::CProcessor, 169
fnSetLogSeverity
sound4::CProcessor, 169

fnSetMetadata
sound4::CProcessor, 169
fnSetMetadataMulti
sound4::CProcessor, 169
fnSetParameter
sound4::CProcessor, 169
fnSetPresetManager
sound4::CProcessor, 170
fnStartUpdateThread
sound4::CProcessor, 170
fnStereoToMono
sound4::CProcessor, 170
fnStereoToMono_Planar
sound4::CProcessor, 170
fnStopUpdateThread
sound4::CProcessor, 170
fnTerminateProcess
sound4::CProcessor, 170
fnWaitUpdateThreadReady
sound4::CProcessor, 171
fnWebserver
sound4::CProcessor, 171
fnWebserver_FreeString
sound4::CProcessor, 171
fnWebserver_GetAppHealth
sound4::CProcessor, 171
fnWebserver_SetAppHealth
sound4::CProcessor, 171
fnWebserver_Status
sound4::CProcessor, 171
fnWebserver_Stop
sound4::CProcessor, 172
fnWebserver_tcp
sound4::CProcessor, 172
fnWebserver_tcp2
sound4::CProcessor, 172
format
sound4::cloudx1::dyn::helper::SampleFormat< float >, 182
sound4::cloudx1::dyn::helper::SampleFormat< int16_t >, 184
sound4::cloudx1::dyn::helper::SampleFormat< int32_t >, 186
sound4::cloudx1::helper::SampleFormat< float >, 183
sound4::cloudx1::helper::SampleFormat< int16_t >, 184
sound4::cloudx1::helper::SampleFormat< int32_t >, 186
sound4::helper::SampleFormat< float >, 183
sound4::helper::SampleFormat< int16_t >, 185
sound4::helper::SampleFormat< int32_t >, 187
FreeJsonAnswer
sound4::cloudx1::dyn::CDynLib, 96
FreeMetadataInfos
sound4::cloudx1::dyn::CDynLib, 96
FreeParameters
sound4::cloudx1::dyn::CDynLib, 96
FreeParameterValue
sound4::cloudx1::dyn::CDynLib, 96
fs_char
Custom preset managagement, 35
sound4cl_cdef.h, 258
fs_strdup
sound4.x1.cloud.hpp, 214
sound4.x1.cloud_dyn.hpp, 223
sound4cl_cdef.h, 257

Get
sound4::CBus, 86
GetAll
MyPresetManager, 177
sound4::cloudx1::CPresetLoader, 146
sound4::cloudx1::dyn::CPresetLoader, 148
sound4::CPresetLoader, 150
GetAudio
sound4::CInstance, 114
sound4::cloudx1::CInstance, 125
sound4::cloudx1::dyn::CDynLib, 96
sound4::cloudx1::dyn::CInstance, 136
GetAudioInputCount
sound4::cloudx1::dyn::CDynLib, 97
sound4::CProcessor, 155
GetAudioOutputCount
sound4::cloudx1::dyn::CDynLib, 97
sound4::CProcessor, 155
GetBufferIn
sound4::CInstance, 114
sound4::cloudx1::CInstance, 125
sound4::cloudx1::dyn::CDynLib, 97
sound4::cloudx1::dyn::CInstance, 136
GetBufferOut
sound4::CInstance, 114
sound4::cloudx1::CInstance, 125
sound4::cloudx1::dyn::CDynLib, 97
sound4::cloudx1::dyn::CInstance, 136
GetBytesFromFormat
sound4::cloudx1::dyn::CDynLib, 97
sound4::CProcessor, 156
GetChannelCount
sound4::cloudx1::dyn::CDynLib, 97
sound4::CProcessor, 156
GetChunkFrames
sound4::CInstance, 115
sound4::cloudx1::CInstance, 126
sound4::cloudx1::dyn::CInstance, 137
GetChunkSizeInFrames
sound4::cloudx1::dyn::CDynLib, 98
sound4::CProcessor, 156
GetEnv
sound4.x1.cloud-pipe-cxx.cpp, 191
sound4.x1.cloud-pipe-dyn.cpp, 195
sound4cl-pipe.cpp, 237
GetEstimatedDelay
sound4::CInstance, 115
sound4::cloudx1::CInstance, 126
sound4::cloudx1::dyn::CDynLib, 98

sound4::cloudx1::dyn::CInstance, 137
GetFormatFromName
sound4::cloudx1::dyn::CDynLib, 98
sound4::CProcessor, 157
GetFormatName
sound4::cloudx1::dyn::CDynLib, 98
sound4::CProcessor, 157
GetMaxPacketFrame
sound4::cloudx1::dyn::CDynLib, 98
GetMetadataInfos
sound4::CInstance, 115
sound4::cloudx1::CInstance, 126
sound4::cloudx1::dyn::CDynLib, 98
sound4::cloudx1::dyn::CInstance, 137
GetOutputCount
sound4::cloudx1::dyn::CDynLib, 99
GetParam
sound4::CInstance, 116
sound4::cloudx1::CInstance, 127
sound4::cloudx1::dyn::CInstance, 138
GetParameter
sound4::cloudx1::dyn::CDynLib, 99
GetPossibleChunkSizeInFrames
sound4::cloudx1::dyn::CDynLib, 99
sound4::CProcessor, 158
GetPrefixSymbol
sound4::CProcessor, 158
GetProcessChunkFrames
sound4::cloudx1::dyn::CDynLib, 99
GetSampleRate
sound4::cloudx1::dyn::CDynLib, 99
sound4::CProcessor, 158
GetSymbol
sound4::cloudx1::dyn::helper::CDynLoader, 107
sound4::helper::CDynLoader, 110
GetSymbol_safe
sound4::cloudx1::dyn::helper::CDynLoader, 107
sound4::helper::CDynLoader, 110
GetThisLibraryPath
sound4::cloudx1::dyn::helper::CDynLoader, 107
sound4::helper::CDynLoader, 110
GetVersion
sound4::cloudx1::dyn::CDynLib, 99
sound4::CProcessor, 158
GetWebServerAppHealth
sound4::CInstance, 116
sound4::cloudx1::CInstance, 127
sound4::cloudx1::dyn::CInstance, 138
Global setup, 20
CLOUDX1_AUDIO_INPUT_COUNT, 21
CLOUDX1_AUDIO_OUTPUT_COUNT, 21
CLOUDX1_AUDIOFRAME_COUNT, 21
CLOUDX1_CHANNEL_COUNT, 21
cloudx1_GetAudioInputCount, 23
cloudx1_GetAudioOutputCount, 23
cloudx1_GetChannelCount, 23
cloudx1_GetChunkSizeInFrames, 24
cloudx1_GetPossibleChunkSizeInFrames, 24
cloudx1_GetSampleRate, 24
cloudx1_GetVersion, 25
cloudx1_loggerfn, 22
cloudx1_LogSeverity, 22
CLOUDX1_SAMPLE_RATE, 22
cloudx1_SetLoggerCallback, 25
cloudx1_SetLogSeverity, 25
error, 23
fatal, 23
info, 23
none, 23
SOUND4_GetProcessInfo, 26
verbose, 23
verbose2, 23
verbose3, 23
verbose4, 23
verbose5, 23
warning, 23

info
Global setup, 23
sound4, 75
InitProcess
sound4::cloudx1::dyn::CDynLib, 100
InitProcess2
sound4::cloudx1::dyn::CDynLib, 100
InitProcess3
sound4::cloudx1::dyn::CDynLib, 100
InputChannels
sound4::cloudx1, 78
sound4::cloudx1::dyn, 81
InputSampleSize
sound4::cloudx1, 78
sound4::cloudx1::dyn, 81
INVALID_FORMAT
Audio managing (any audio), 44
sound4cl_cdef.h, 261
IsOk
sound4::CInstance, 116
sound4::cloudx1::CInstance, 127
sound4::cloudx1::dyn::CDynLib, 94
sound4::cloudx1::dyn::CInstance, 138
sound4::cloudx1::dyn::helper::CDynLoader, 107
sound4::cloudx1::dyn::helper::DynFuncHolder< T >, 173
sound4::CProcessor, 159
sound4::helper::CDynLoader, 110
sound4::helper::DynFuncHolder< T >, 175
IsReadOnly
MyPresetManager, 177, 178
sound4::cloudx1::CPresetLoader, 146
sound4::cloudx1::dyn::CPresetLoader, 148
sound4::CPresetLoader, 151

JSON syntax, 54
JSON syntax for internal setup, 57

Load
sound4::cloudx1::dyn::CDynLib, 94

sound4::cloudx1::dyn::helper::CDynLoader, 107
sound4::CProcessor, 159
sound4::helper::CDynLoader, 110
log_cb_t
sound4, 74
sound4::cloudx1, 77
sound4::cloudx1::dyn, 80
LogSeverity
sound4, 74
sound4::cloudx1, 77
sound4::cloudx1::dyn, 80

main
sound4.x1.cloud-pipe-cxx.cpp, 191
sound4.x1.cloud-pipe-dyn.cpp, 195
sound4.x1.cloud-pipe.cpp, 201
sound4cl-pipe.cpp, 237
Metadata Management, 37
cloudx1_FreeMetadataInfos, 37
cloudx1_GetMetadataInfos, 38
cloudx1_SetMetadata, 38
cloudx1_SetMetadataMulti, 39
MonoToStereo
sound4::cloudx1::dyn::CDynLib, 100
sound4::CProcessor, 159
MonoToStereo_Planar
sound4::cloudx1::dyn::CDynLib, 100
sound4::CProcessor, 160
MyLogger
sound4.x1.cloud-pipe-cxx.cpp, 192
sound4.x1.cloud-pipe-dyn.cpp, 196
sound4.x1.cloud-pipe.cpp, 201
MyPresetManager, 175
∼MyPresetManager, 176
Exists, 177
GetAll, 177
IsReadOnly, 177, 178
MyPresetManager, 176
Read, 178
Remove, 178
Rename, 179
SetupPath, 179
Write, 179, 180

name
SOUND4_ProcessInfo, 188
sound4cl_SOUND4_ProcessInfo, 189
NewClient
sound4::CInstance, 116
sound4::cloudx1::CInstance, 127
sound4::cloudx1::dyn::CDynLib, 100
sound4::cloudx1::dyn::CInstance, 138
NewParameters
sound4::cloudx1::dyn::CDynLib, 101
none
Global setup, 23
sound4, 75

OnUpdateThreadStart
sound4::CInstance, 117
sound4::cloudx1::CInstance, 128
sound4::cloudx1::dyn::CInstance, 139
OnUpdateThreadStop
sound4::CInstance, 117
sound4::cloudx1::CInstance, 128
sound4::cloudx1::dyn::CInstance, 139
operator bool
sound4::cloudx1::dyn::helper::CDynLoader, 108
sound4::cloudx1::dyn::helper::DynFuncHolder< T >, 173
sound4::helper::CDynLoader, 111
sound4::helper::DynFuncHolder< T >, 175
operator T*
sound4::cloudx1::dyn::helper::DynFuncHolder< T >, 173
sound4::helper::DynFuncHolder< T >, 175
operator=
sound4::CInstance, 117
sound4::CInstance::CClient, 87
sound4::cloudx1::CInstance, 128
sound4::cloudx1::CInstance::CClient, 89
sound4::cloudx1::dyn::CInstance, 139
sound4::cloudx1::dyn::CInstance::CClient, 91
sound4::cloudx1::dyn::helper::CDynLoader, 108
sound4::helper::CDynLoader, 111
OutputChannels
sound4::cloudx1, 78
sound4::cloudx1::dyn, 81
OutputSampleSize
sound4::cloudx1, 79
sound4::cloudx1::dyn, 81

prefix
SOUND4_ProcessInfo, 188
sound4cl_SOUND4_ProcessInfo, 190
PresetChange_Kind
sound4, 74
sound4::cloudx1, 77
sound4::cloudx1::dyn, 80
PresetManager_InformChange
sound4::CInstance, 117
sound4::cloudx1::CInstance, 128
sound4::cloudx1::dyn::CDynLib, 101
sound4::cloudx1::dyn::CInstance, 139
Process creation/destruction, 26
cloudx1_ExitProcess, 27
cloudx1_FreeParameters, 27
cloudx1_FreeParameterValue, 27
cloudx1_GetParameter, 27
cloudx1_GetProcessChunkFrames, 28
cloudx1_InitProcess, 28
cloudx1_InitProcess2, 28
cloudx1_InitProcess3, 29
cloudx1_NewParameters, 30
cloudx1_SetParameter, 30
cloudx1_TerminateProcess, 31
ProcessAnyAudio
sound4::CInstance, 118

sound4::cloudx1::CInstance, 129
sound4::cloudx1::dyn::CInstance, 140
ProcessAudio
sound4::CInstance, 118
sound4::cloudx1::CInstance, 129
sound4::cloudx1::dyn::CDynLib, 101
sound4::cloudx1::dyn::CInstance, 140
ProcessAudio_Planar
sound4::CInstance, 118
sound4::cloudx1::CInstance, 129
sound4::cloudx1::dyn::CDynLib, 101
sound4::cloudx1::dyn::CInstance, 140
ProcessJson
sound4::CInstance::CClient, 87
sound4::cloudx1::CInstance::CClient, 89
sound4::cloudx1::dyn::CDynLib, 101
sound4::cloudx1::dyn::CInstance::CClient, 91
ProcessLoop
sound4.x1.cloud-pipe-cxx.cpp, 192
sound4.x1.cloud-pipe-dyn.cpp, 196
sound4.x1.cloud-pipe.cpp, 202
sound4cl-pipe.cpp, 237

Read
MyPresetManager, 178
sound4::cloudx1::CPresetLoader, 146
sound4::cloudx1::dyn::CPresetLoader, 148
sound4::CPresetLoader, 151
Remove
MyPresetManager, 178
sound4::cloudx1::CPresetLoader, 146
sound4::cloudx1::dyn::CPresetLoader, 149
sound4::CPresetLoader, 151
Rename
MyPresetManager, 179
sound4::cloudx1::CPresetLoader, 147
sound4::cloudx1::dyn::CPresetLoader, 149
sound4::CPresetLoader, 151

S16_BE
Audio managing (any audio), 44
sound4cl_cdef.h, 261
S16_LE
Audio managing (any audio), 44
sound4cl_cdef.h, 261
S16_NATIVE
Audio managing (any audio), 44
sound4cl_cdef.h, 261
S24_BE
Audio managing (any audio), 44
sound4cl_cdef.h, 261
S24_LE
Audio managing (any audio), 44
sound4cl_cdef.h, 261
S24_NATIVE
Audio managing (any audio), 44
sound4cl_cdef.h, 261
S32_BE
Audio managing (any audio), 44
sound4cl_cdef.h, 261
S32_LE
Audio managing (any audio), 44
sound4cl_cdef.h, 261
S32_NATIVE
Audio managing (any audio), 44
sound4cl_cdef.h, 261
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.x1.cloud-pipe.cpp, 201
SampleFormat
sound4::cloudx1, 78
sound4::cloudx1::dyn, 80
SampleRate
sound4::cloudx1, 79
sound4::cloudx1::dyn, 81
SaveState
sound4::cloudx1::dyn::CDynLib, 101
SetBus
sound4::CInstance, 119
SetLoggerCallback
sound4::cloudx1::dyn::CDynLib, 102
sound4::CProcessor, 160
SetLogSeverity
sound4::cloudx1::dyn::CDynLib, 102
sound4::CProcessor, 160
SetMetadata
sound4::CInstance, 119
sound4::cloudx1::CInstance, 130
sound4::cloudx1::dyn::CDynLib, 102
sound4::cloudx1::dyn::CInstance, 141
SetMetadataMulti
sound4::CInstance, 120
sound4::cloudx1::CInstance, 130
sound4::cloudx1::dyn::CDynLib, 102
sound4::cloudx1::dyn::CInstance, 141
SetParam
sound4::CInstance, 120
sound4::cloudx1::CInstance, 131
sound4::cloudx1::dyn::CInstance, 142
SetParameter
sound4::cloudx1::dyn::CDynLib, 102
SetPresetManager
sound4::CInstance, 120
sound4::cloudx1::CInstance, 131
sound4::cloudx1::dyn::CDynLib, 102
sound4::cloudx1::dyn::CInstance, 142
SetupPath
MyPresetManager, 179
SetWebServerAppHealth
sound4::CInstance, 121
sound4::cloudx1::CInstance, 132
sound4::cloudx1::dyn::CInstance, 143

shortname
SOUND4_ProcessInfo, 188
sound4cl_SOUND4_ProcessInfo, 190
sound4, 73
error, 75
fatal, 75
info, 75
log_cb_t, 74
LogSeverity, 74
none, 75
PresetChange_Kind, 74
verbose, 75
verbose2, 75
verbose3, 76
verbose4, 76
verbose5, 76
warning, 76
sound4.x1.cloud-pipe-cxx.cpp, 191, 193
GetEnv, 191
main, 191
MyLogger, 192
ProcessLoop, 192
Usage, 192
sound4.x1.cloud-pipe-dyn.cpp, 194, 197
GetEnv, 195
main, 195
MyLogger, 196
ProcessLoop, 196
Usage, 196
sound4.x1.cloud-pipe.cpp, 200, 202
CHANNEL_COUNT, 200
CHUNK_SIZE, 201
main, 201
MyLogger, 201
ProcessLoop, 202
SAMPLE_RATE, 201
Usage, 202
sound4.x1.cloud.h, 205, 209
CLOUDX1_DECL, 208
CLOUDX1_HAS_CLOUDBUS, 208
CLOUDX1_HAS_WEBSERVER, 208
sound4.x1.cloud.hpp, 213, 214
fs_strdup, 214
sound4.x1.cloud_dyn.hpp, 221, 223
fs_strdup, 223
sound4::CBus, 85
~CBus, 85
CBus, 85
Get, 86
sound4::CInstance, 111
~CInstance, 113
AddAudio, 113
CInstance, 113
Create, 113
GetAudio, 114
GetBufferIn, 114
GetBufferOut, 114
GetChunkFrames, 115
GetEstimatedDelay, 115
GetMetadataInfos, 115
GetParam, 116
GetWebServerAppHealth, 116
IsOk, 116
NewClient, 116
OnUpdateThreadStart, 117
OnUpdateThreadStop, 117
operator=, 117
PresetManager_InformChange, 117
ProcessAnyAudio, 118
ProcessAudio, 118
ProcessAudio_Planar, 118
SetBus, 119
SetMetadata, 119
SetMetadataMulti, 120
SetParam, 120
SetPresetManager, 120
SetWebServerAppHealth, 121
StartWebServer, 121
Stop, 121
StopWebServer, 122
Terminate, 122
sound4::CInstance::CClient, 86
~CClient, 87
CClient, 87
operator=, 87
ProcessJson, 87
sound4::cloudx1, 76
ChunkFrames, 78
InputChannels, 78
InputSampleSize, 78
log_cb_t, 77
LogSeverity, 77
OutputChannels, 78
OutputSampleSize, 79
PresetChange_Kind, 77
SampleFormat, 78
SampleRate, 79
sound4::cloudx1::CInstance, 122
~CInstance, 124
AddAudio, 124
CInstance, 124
Create, 124
GetAudio, 125
GetBufferIn, 125
GetBufferOut, 125
GetChunkFrames, 126
GetEstimatedDelay, 126
GetMetadataInfos, 126
GetParam, 127
GetWebServerAppHealth, 127
IsOk, 127
NewClient, 127
OnUpdateThreadStart, 128
OnUpdateThreadStop, 128
operator=, 128
PresetManager_InformChange, 128

ProcessAnyAudio, 129
ProcessAudio, 129
ProcessAudio_Planar, 129
SetMetadata, 130
SetMetadataMulti, 130
SetParam, 131
SetPresetManager, 131
SetWebServerAppHealth, 132
StartWebServer, 132
Stop, 132
StopWebServer, 132
Terminate, 133
sound4::cloudx1::CInstance::CClient, 88
~CClient, 89
CClient, 89
operator=, 89
ProcessJson, 89
sound4::cloudx1::CPresetLoader, 145
~CPresetLoader, 146
CPresetLoader, 146
Exists, 146
GetAll, 146
IsReadOnly, 146
Read, 146
Remove, 146
Rename, 147
Write, 147
sound4::cloudx1::dyn, 79
ChunkFrames, 81
InputChannels, 81
InputSampleSize, 81
log_cb_t, 80
LogSeverity, 80
OutputChannels, 81
OutputSampleSize, 81
PresetChange_Kind, 80
SampleFormat, 80
SampleRate, 81
sound4::cloudx1::dyn::CDynLib, 92
AddAudio, 94
AddPadAudio, 95
AudioConvertFrom, 95
AudioConvertTo, 95
AudioMonoFromLiveStereo, 95
AudioMonoToLiveStereo, 95
CDynLib, 94
DeleteClient, 95
ExitProcess, 96
FreeJsonAnswer, 96
FreeMetadataInfos, 96
FreeParameters, 96
FreeParameterValue, 96
GetAudio, 96
GetAudioInputCount, 97
GetAudioOutputCount, 97
GetBufferIn, 97
GetBufferOut, 97
GetBytesFromFormat, 97
GetChannelCount, 97
GetChunkSizeInFrames, 98
GetEstimatedDelay, 98
GetFormatFromName, 98
GetFormatName, 98
GetMaxPacketFrame, 98
GetMetadataInfos, 98
GetOutputCount, 99
GetParameter, 99
GetPossibleChunkSizeInFrames, 99
GetProcessChunkFrames, 99
GetSampleRate, 99
GetVersion, 99
InitProcess, 100
InitProcess2, 100
InitProcess3, 100
IsOk, 94
Load, 94
MonoToStereo, 100
MonoToStereo_Planar, 100
NewClient, 100
NewParameters, 101
PresetManager_InformChange, 101
ProcessAudio, 101
ProcessAudio_Planar, 101
ProcessJson, 101
SaveState, 101
SetLoggerCallback, 102
SetLogSeverity, 102
SetMetadata, 102
SetMetadataMulti, 102
SetParameter, 102
SetPresetManager, 102
StartUpdateThread, 103
StereoToMono, 103
StereoToMono_Planar, 103
StopUpdateThread, 103
TerminateProcess, 103
WaitUpdateThreadReady, 103
Webserver, 104
Webserver_FreeString, 104
Webserver_GetAppHealth, 104
Webserver_SetAppHealth, 104
Webserver_Status, 104
Webserver_Stop, 104
Webserver_tcp, 105
Webserver_tcp2, 105
sound4::cloudx1::dyn::CInstance, 133
~CInstance, 135
AddAudio, 135
CInstance, 134, 135
Create, 135
GetAudio, 136
GetBufferIn, 136
GetBufferOut, 136
GetChunkFrames, 137
GetEstimatedDelay, 137
GetMetadataInfos, 137

GetParam, 138
GetWebServerAppHealth, 138
IsOk, 138
NewClient, 138
OnUpdateThreadStart, 139
OnUpdateThreadStop, 139
operator=, 139
PresetManager_InformChange, 139
ProcessAnyAudio, 140
ProcessAudio, 140
ProcessAudio_Planar, 140
SetMetadata, 141
SetMetadataMulti, 141
SetParam, 142
SetPresetManager, 142
SetWebServerAppHealth, 143
StartWebServer, 143
Stop, 143
StopWebServer, 143
Terminate, 144
sound4::cloudx1::dyn::CInstance::CClient, 90
~CClient, 91
CClient, 90, 91
operator=, 91
ProcessJson, 91
sound4::cloudx1::dyn::CPresetLoader, 147
~CPresetLoader, 148
CPresetLoader, 148
Exists, 148
GetAll, 148
IsReadOnly, 148
Read, 148
Remove, 149
Rename, 149
Write, 149
sound4::cloudx1::dyn::helper, 82
DynFunc_t, 82
DynLib_t, 82
sound4::cloudx1::dyn::helper::CDynLoader, 105
~CDynLoader, 106
CDynLoader, 106
Close, 106
GetSymbol, 107
GetSymbol_safe, 107
GetThisLibraryPath, 107
IsOk, 107
Load, 107
operator bool, 108
operator=, 108
sound4::cloudx1::dyn::helper::DynFuncHolder< T >, 172
DynFuncHolder, 173
IsOk, 173
operator bool, 173
operator T*, 173
sound4::cloudx1::dyn::helper::SampleFormat< float >, 182
format, 182
sound4::cloudx1::dyn::helper::SampleFormat< int16_t >, 183
format, 184
sound4::cloudx1::dyn::helper::SampleFormat< int32_t >, 185
format, 186
sound4::cloudx1::dyn::helper::SampleFormat< T >, 180
sound4::cloudx1::helper, 83
sound4::cloudx1::helper::SampleFormat< float >, 182
format, 183
sound4::cloudx1::helper::SampleFormat< int16_t >, 184
format, 184
sound4::cloudx1::helper::SampleFormat< int32_t >, 186
format, 186
sound4::cloudx1::helper::SampleFormat< T >, 181
sound4::CPresetLoader, 149
~CPresetLoader, 150
CPresetLoader, 150
Exists, 150
GetAll, 150
IsReadOnly, 151
Read, 151
Remove, 151
Rename, 151
Write, 151
sound4::CProcessor, 152
AudioConvertFrom, 154
AudioConvertTo, 154
AudioMonoFromLiveStereo, 155
AudioMonoToLiveStereo, 155
CBus, 161
CInstance, 161
CProcessor, 154
fnAddAudio, 161
fnAddPadAudio, 161
fnAudioConvertFrom, 161
fnAudioConvertTo, 162
fnAudioMonoFromLiveStereo, 162
fnAudioMonoToLiveStereo, 162
fnDeleteClient, 162
fnExitProcess, 162
fnFreeBus, 162
fnFreeJsonAnswer, 163
fnFreeMetadataInfos, 163
fnFreeParameters, 163
fnFreeParameterValue, 163
fnGetAudio, 163
fnGetAudioInputCount, 163
fnGetAudioOutputCount, 164
fnGetBufferIn, 164
fnGetBufferOut, 164
fnGetBytesFromFormat, 164
fnGetChannelCount, 164
fnGetChunkSizeInFrames, 164
fnGetEstimatedDelay, 165

fnGetFormatFromName, 165
fnGetFormatName, 165
fnGetMaxPacketFrame, 165
fnGetMetadataInfos, 165
fnGetOutputCount, 165
fnGetParameter, 166
fnGetPossibleChunkSizeInFrames, 166
fnGetProcessChunkFrames, 166
fnGetSampleRate, 166
fnGetVersion, 166
fnInitProcess, 166
fnInitProcess2, 167
fnInitProcess3, 167
fnMonoToStereo, 167
fnMonoToStereo_Planar, 167
fnNewBus, 167
fnNewClient, 167
fnNewParameters, 168
fnPresetManager_InformChange, 168
fnProcessAudio, 168
fnProcessAudio_Planar, 168
fnProcessJson, 168
fnSaveState, 168
fnSetInstanceBus, 169
fnSetLoggerCallback, 169
fnSetLogSeverity, 169
fnSetMetadata, 169
fnSetMetadataMulti, 169
fnSetParameter, 169
fnSetPresetManager, 170
fnStartUpdateThread, 170
fnStereoToMono, 170
fnStereoToMono_Planar, 170
fnStopUpdateThread, 170
fnTerminateProcess, 170
fnWaitUpdateThreadReady, 171
fnWebserver, 171
fnWebserver_FreeString, 171
fnWebserver_GetAppHealth, 171
fnWebserver_SetAppHealth, 171
fnWebserver_Status, 171
fnWebserver_Stop, 172
fnWebserver_tcp, 172
fnWebserver_tcp2, 172
GetAudioInputCount, 155
GetAudioOutputCount, 155
GetBytesFromFormat, 156
GetChannelCount, 156
GetChunkSizeInFrames, 156
GetFormatFromName, 157
GetFormatName, 157
GetPossibleChunkSizeInFrames, 158
GetPrefixSymbol, 158
GetSampleRate, 158
GetVersion, 158
IsOk, 159
Load, 159
MonoToStereo, 159
MonoToStereo_Planar, 160
SetLoggerCallback, 160
SetLogSeverity, 160
StereoToMono, 160
StereoToMono_Planar, 160
sound4::helper, 83
DynFunc_t, 84
DynLib_t, 84
sound4::helper::CDynLoader, 108
~CDynLoader, 109
CDynLoader, 109
Close, 109
GetSymbol, 110
GetSymbol_safe, 110
GetThisLibraryPath, 110
IsOk, 110
Load, 110
operator bool, 111
operator=, 111
sound4::helper::DynFuncHolder< T >, 174
DynFuncHolder, 174
IsOk, 175
operator bool, 175
operator T*, 175
sound4::helper::SampleFormat< float >, 183
format, 183
sound4::helper::SampleFormat< int16_t >, 185
format, 185
sound4::helper::SampleFormat< int32_t >, 187
format, 187
sound4::helper::SampleFormat< T >, 181
SOUND4_GetProcessInfo
Global setup, 26
SOUND4_INVALID_WEBSERVER_ID
sound4cl_cdef.h, 258
Web server, 61
SOUND4_ProcessInfo, 187
name, 188
prefix, 188
shortname, 188
struct_size, 188
version, 189
SOUND4_WEBSERVER_HTTP_OK
sound4cl_cdef.h, 258
Web server, 61
SOUND4_WEBSERVER_HTTPS_OK
sound4cl_cdef.h, 258
Web server, 62
sound4cl-pipe.cpp, 236, 238
GetEnv, 237
main, 237
ProcessLoop, 237
sound4cl.hpp, 241, 242
sound4cl_AddAudio
sound4cl_cdef.h, 261
sound4cl_AddPadAudio
sound4cl_cdef.h, 261
sound4cl_AudioConvertFrom

sound4cl_cdef.h, 261
sound4cl_AudioConvertTo
sound4cl_cdef.h, 262
sound4cl_AudioMonoFromLiveStereo
sound4cl_cdef.h, 262
sound4cl_AudioMonoToLiveStereo
sound4cl_cdef.h, 262
sound4cl_cdef.h, 255, 272
change_kind_created, 260
change_kind_deleted, 260
change_kind_modified, 260
F32_BE, 261
F32_LE, 261
F32_NATIVE, 261
fs_char, 258
fs_strdup, 257
INVALID_FORMAT, 261
S16_BE, 261
S16_LE, 261
S16_NATIVE, 261
S24_BE, 261
S24_LE, 261
S24_NATIVE, 261
S32_BE, 261
S32_LE, 261
S32_NATIVE, 261
SOUND4_INVALID_WEBSERVER_ID, 258
SOUND4_WEBSERVER_HTTP_OK, 258
SOUND4_WEBSERVER_HTTPS_OK, 258
sound4cl_AddAudio, 261
sound4cl_AddPadAudio, 261
sound4cl_AudioConvertFrom, 261
sound4cl_AudioConvertTo, 262
sound4cl_AudioMonoFromLiveStereo, 262
sound4cl_AudioMonoToLiveStereo, 262
sound4cl_DeleteClient, 262
sound4cl_error, 260
sound4cl_ExitProcess, 262
sound4cl_fatal, 260
sound4cl_FreeBus, 262
sound4cl_FreeJsonAnswer, 263
sound4cl_FreeMetadataInfos, 263
sound4cl_FreeParameters, 263
sound4cl_FreeParameterValue, 263
sound4cl_GetAudio, 263
sound4cl_GetAudioInputCount, 263
sound4cl_GetAudioOutputCount, 264
sound4cl_GetBufferIn, 264
sound4cl_GetBufferOut, 264
sound4cl_GetBytesFromFormat, 264
sound4cl_GetChannelCount, 264
sound4cl_GetChunkSizeInFrames, 264
sound4cl_GetEstimatedDelay, 264
sound4cl_GetFormatFromName, 265
sound4cl_GetFormatName, 265
sound4cl_GetMaxPacketFrame, 265
sound4cl_GetMetadataInfos, 265
sound4cl_GetOutputCount, 265
sound4cl_GetParameter, 265
sound4cl_GetPossibleChunkSizeInFrames, 265
sound4cl_GetProcessChunkFrames, 266
sound4cl_GetSampleRate, 266
sound4cl_GetVersion, 266
sound4cl_info, 260
sound4cl_InitProcess, 266
sound4cl_InitProcess2, 266
sound4cl_InitProcess3, 266
sound4cl_loggerfn, 258
sound4cl_LogSeverity, 260
sound4cl_MonoToStereo, 267
sound4cl_MonoToStereo_Planar, 267
sound4cl_NewBus, 267
sound4cl_NewClient, 267
sound4cl_NewParameters, 267
sound4cl_none, 260
sound4cl_PresetChange_Kind, 260
sound4cl_PresetManager_InformChange, 267
sound4cl_ProcessAudio, 268
sound4cl_ProcessAudio_Planar, 268
sound4cl_ProcessJson, 268
sound4cl_SampleFormat, 261
sound4cl_SaveState, 268
sound4cl_SetInstanceBus, 268
sound4cl_SetLoggerCallback, 268
sound4cl_SetLogSeverity, 269
sound4cl_SetMetadata, 269
sound4cl_SetMetadataMulti, 269
sound4cl_SetParameter, 269
sound4cl_SetPresetManager, 269
sound4cl_SOUND4_GetProcessInfo, 269
sound4cl_StartUpdateThread, 270
sound4cl_StereoToMono, 270
sound4cl_StereoToMono_Planar, 270
sound4cl_StopUpdateThread, 270
sound4cl_storage_exists, 258
sound4cl_storage_getall, 259
sound4cl_storage_getall_free, 259
sound4cl_storage_reader, 259
sound4cl_storage_reader_free, 259
sound4cl_storage_remove, 259
sound4cl_storage_rename, 259
sound4cl_storage_writer, 260
sound4cl_TerminateProcess, 270
sound4cl_verbose, 260
sound4cl_verbose2, 260
sound4cl_verbose3, 260
sound4cl_verbose4, 260
sound4cl_verbose5, 260
sound4cl_WaitUpdateThreadReady, 270
sound4cl_warning, 260
sound4cl_Webserver, 271
sound4cl_Webserver_FreeString, 271
sound4cl_Webserver_GetAppHealth, 271
sound4cl_Webserver_SetAppHealth, 271
sound4cl_Webserver_Status, 271
sound4cl_Webserver_Stop, 271

sound4cl_Webserver_tcp, 272
sound4cl_Webserver_tcp2, 272
sound4cl_DeleteClient
sound4cl_cdef.h, 262
sound4cl_error
sound4cl_cdef.h, 260
sound4cl_ExitProcess
sound4cl_cdef.h, 262
sound4cl_fatal
sound4cl_cdef.h, 260
sound4cl_FreeBus
sound4cl_cdef.h, 262
sound4cl_FreeJsonAnswer
sound4cl_cdef.h, 263
sound4cl_FreeMetadataInfos
sound4cl_cdef.h, 263
sound4cl_FreeParameters
sound4cl_cdef.h, 263
sound4cl_FreeParameterValue
sound4cl_cdef.h, 263
sound4cl_GetAudio
sound4cl_cdef.h, 263
sound4cl_GetAudioInputCount
sound4cl_cdef.h, 263
sound4cl_GetAudioOutputCount
sound4cl_cdef.h, 264
sound4cl_GetBufferIn
sound4cl_cdef.h, 264
sound4cl_GetBufferOut
sound4cl_cdef.h, 264
sound4cl_GetBytesFromFormat
sound4cl_cdef.h, 264
sound4cl_GetChannelCount
sound4cl_cdef.h, 264
sound4cl_GetChunkSizeInFrames
sound4cl_cdef.h, 264
sound4cl_GetEstimatedDelay
sound4cl_cdef.h, 264
sound4cl_GetFormatFromName
sound4cl_cdef.h, 265
sound4cl_GetFormatName
sound4cl_cdef.h, 265
sound4cl_GetMaxPacketFrame
sound4cl_cdef.h, 265
sound4cl_GetMetadataInfos
sound4cl_cdef.h, 265
sound4cl_GetOutputCount
sound4cl_cdef.h, 265
sound4cl_GetParameter
sound4cl_cdef.h, 265
sound4cl_GetPossibleChunkSizeInFrames
sound4cl_cdef.h, 265
sound4cl_GetProcessChunkFrames
sound4cl_cdef.h, 266
sound4cl_GetSampleRate
sound4cl_cdef.h, 266
sound4cl_GetVersion
sound4cl_cdef.h, 266
sound4cl_info
sound4cl_cdef.h, 260
sound4cl_InitProcess
sound4cl_cdef.h, 266
sound4cl_InitProcess2
sound4cl_cdef.h, 266
sound4cl_InitProcess3
sound4cl_cdef.h, 266
sound4cl_loggerfn
sound4cl_cdef.h, 258
sound4cl_LogSeverity
sound4cl_cdef.h, 260
sound4cl_MonoToStereo
sound4cl_cdef.h, 267
sound4cl_MonoToStereo_Planar
sound4cl_cdef.h, 267
sound4cl_NewBus
sound4cl_cdef.h, 267
sound4cl_NewClient
sound4cl_cdef.h, 267
sound4cl_NewParameters
sound4cl_cdef.h, 267
sound4cl_none
sound4cl_cdef.h, 260
sound4cl_PresetChange_Kind
sound4cl_cdef.h, 260
sound4cl_PresetManager_InformChange
sound4cl_cdef.h, 267
sound4cl_ProcessAudio
sound4cl_cdef.h, 268
sound4cl_ProcessAudio_Planar
sound4cl_cdef.h, 268
sound4cl_ProcessJson
sound4cl_cdef.h, 268
sound4cl_SampleFormat
sound4cl_cdef.h, 261
sound4cl_SaveState
sound4cl_cdef.h, 268
sound4cl_SetInstanceBus
sound4cl_cdef.h, 268
sound4cl_SetLoggerCallback
sound4cl_cdef.h, 268
sound4cl_SetLogSeverity
sound4cl_cdef.h, 269
sound4cl_SetMetadata
sound4cl_cdef.h, 269
sound4cl_SetMetadataMulti
sound4cl_cdef.h, 269
sound4cl_SetParameter
sound4cl_cdef.h, 269
sound4cl_SetPresetManager
sound4cl_cdef.h, 269
sound4cl_SOUND4_GetProcessInfo
sound4cl_cdef.h, 269
sound4cl_SOUND4_ProcessInfo, 189
name, 189
prefix, 190
shortname, 190

struct_size, 190
version, 190
sound4cl_StartUpdateThread
sound4cl_cdef.h, 270
sound4cl_StereoToMono
sound4cl_cdef.h, 270
sound4cl_StereoToMono_Planar
sound4cl_cdef.h, 270
sound4cl_StopUpdateThread
sound4cl_cdef.h, 270
sound4cl_storage_exists
sound4cl_cdef.h, 258
sound4cl_storage_getall
sound4cl_cdef.h, 259
sound4cl_storage_getall_free
sound4cl_cdef.h, 259
sound4cl_storage_reader
sound4cl_cdef.h, 259
sound4cl_storage_reader_free
sound4cl_cdef.h, 259
sound4cl_storage_remove
sound4cl_cdef.h, 259
sound4cl_storage_rename
sound4cl_cdef.h, 259
sound4cl_storage_writer
sound4cl_cdef.h, 260
sound4cl_TerminateProcess
sound4cl_cdef.h, 270
sound4cl_verbose
sound4cl_cdef.h, 260
sound4cl_verbose2
sound4cl_cdef.h, 260
sound4cl_verbose3
sound4cl_cdef.h, 260
sound4cl_verbose4
sound4cl_cdef.h, 260
sound4cl_verbose5
sound4cl_cdef.h, 260
sound4cl_WaitUpdateThreadReady
sound4cl_cdef.h, 270
sound4cl_warning
sound4cl_cdef.h, 260
sound4cl_Webserver
sound4cl_cdef.h, 271
sound4cl_Webserver_FreeString
sound4cl_cdef.h, 271
sound4cl_Webserver_GetAppHealth
sound4cl_cdef.h, 271
sound4cl_Webserver_SetAppHealth
sound4cl_cdef.h, 271
sound4cl_Webserver_Status
sound4cl_cdef.h, 271
sound4cl_Webserver_Stop
sound4cl_cdef.h, 271
sound4cl_Webserver_tcp
sound4cl_cdef.h, 272
sound4cl_Webserver_tcp2
sound4cl_cdef.h, 272
StartUpdateThread
sound4::cloudx1::dyn::CDynLib, 103
StartWebServer
sound4::CInstance, 121
sound4::cloudx1::CInstance, 132
sound4::cloudx1::dyn::CInstance, 143
std, 84
StereoToMono
sound4::cloudx1::dyn::CDynLib, 103
sound4::CProcessor, 160
StereoToMono_Planar
sound4::cloudx1::dyn::CDynLib, 103
sound4::CProcessor, 160
Stop
sound4::CInstance, 121
sound4::cloudx1::CInstance, 132
sound4::cloudx1::dyn::CInstance, 143
StopUpdateThread
sound4::cloudx1::dyn::CDynLib, 103
StopWebServer
sound4::CInstance, 122
sound4::cloudx1::CInstance, 132
sound4::cloudx1::dyn::CInstance, 143
struct_size
SOUND4_ProcessInfo, 188
sound4cl_SOUND4_ProcessInfo, 190

Terminate
sound4::CInstance, 122
sound4::cloudx1::CInstance, 133
sound4::cloudx1::dyn::CInstance, 144
TerminateProcess
sound4::cloudx1::dyn::CDynLib, 103

Update Management, 39
cloudx1_StartUpdateThread, 39
cloudx1_StopUpdateThread, 40
cloudx1_WaitUpdateThreadReady, 40
Usage
sound4.x1.cloud-pipe-cxx.cpp, 192
sound4.x1.cloud-pipe-dyn.cpp, 196
sound4.x1.cloud-pipe.cpp, 202

verbose
Global setup, 23
sound4, 75
verbose2
Global setup, 23
sound4, 75
verbose3
Global setup, 23
sound4, 76
verbose4
Global setup, 23
sound4, 76
verbose5
Global setup, 23
sound4, 76
version

SOUND4_ProcessInfo, 189
sound4cl_SOUND4_ProcessInfo, 190

WaitUpdateThreadReady
sound4::cloudx1::dyn::CDynLib, 103
warning
Global setup, 23
sound4, 76
Web server, 61
cloudx1_Webserver, 62
cloudx1_Webserver_FreeString, 63
cloudx1_Webserver_GetAppHealth, 63
cloudx1_Webserver_SetAppHealth, 63
cloudx1_Webserver_Status, 64
cloudx1_Webserver_Stop, 64
cloudx1_Webserver_tcp, 65
cloudx1_Webserver_tcp2, 65
SOUND4_INVALID_WEBSERVER_ID, 61
SOUND4_WEBSERVER_HTTP_OK, 61
SOUND4_WEBSERVER_HTTPS_OK, 62
Webserver
sound4::cloudx1::dyn::CDynLib, 104
Webserver_FreeString
sound4::cloudx1::dyn::CDynLib, 104
Webserver_GetAppHealth
sound4::cloudx1::dyn::CDynLib, 104
Webserver_SetAppHealth
sound4::cloudx1::dyn::CDynLib, 104
Webserver_Status
sound4::cloudx1::dyn::CDynLib, 104
Webserver_Stop
sound4::cloudx1::dyn::CDynLib, 104
Webserver_tcp
sound4::cloudx1::dyn::CDynLib, 105
Webserver_tcp2
sound4::cloudx1::dyn::CDynLib, 105
Write
MyPresetManager, 179, 180
sound4::cloudx1::CPresetLoader, 147
sound4::cloudx1::dyn::CPresetLoader, 149
sound4::CPresetLoader, 151