GLOBAL IP SOLUTIONS

GIPS VoiceEngine:
Porting from 2.x to 3.x

Porting Guide

Headquarters
Global IP Solutions, Inc.
301 Brannan Street, 6th floor
San Francisco, CA 94107
USA

Phone: +1 415 397 2555
Fax: +1 415 397 2577

For additional contact information, please visit the GIPS URL: http://www.gipscorp.com

For information on how to contact Customer Support, please visit the following URL:
https://www.gipssolutions.com/customer/login.asp

Contents

1 About This Guide

This document provides guidelines and describes how to port GIPS VoiceEngine version 2.x to the new version 3.x.

The main changes between 2.x and 3.x are in:

- how to allocate resources;
- how the API has been split up into several smaller interfaces, or sub-APIs; and
- how to release resources.

In addition, a number of minor changes exist:

- modified API names to make their function more clear;
- parameter changes to make the VoiceEngine API more consistent; and
- enumerators used instead of integer parameters.

The document has been written based on a generic example application that supports both versions. Code snippets from the example are shown in Chapter 2: Application Changes, and the complete example can be found in Appendix A: Generic Code Example.

Product Version

This GIPS VoiceEngine Porting Guide corresponds to GIPS VoiceEngine product versions 2.6 and 3.0.

In This Guide

- Chapter 2, Application Changes describes what kind of changes must be done to port an application from VoiceEngine version 2.x to version 3.x.
- Appendix A, Generic Code Example lists the generic example application on which this porting guide has been based.
- Appendix B, Modified Interfaces summarizes all API changes (names, parameters, enumerators etc.) in a table.

Document Change History

Version	Date	Change Summary
1.0	2008-04-11	Initial document release.
1.1	2008-08-15	Changes given latest VoiceEngine 3.0.1 release.

Obtaining GIPS Documentation

White papers, case studies, test tools, guides, and other documents can be viewed or downloaded from the Global IP Solutions Developer Community Forum at developer.gipscorp.com.

Related Documents

The following guides are related to the GIPS VoiceEngine:

- GIPS VoiceEngine API Guide

2 Application Changes

This chapter describes what kind of changes must be made in application source to port from GIPS VoiceEngine version 2.x to version 3.x.

Resource Allocation

VoiceEngine 2.x and 3.x use different factory methods.

The old way to create a VoiceEngine object:

```
GipsVoiceEngineLib* ve = GetNewVoiceEngineLib();
```

must now be changed to:

```
GIPSVoiceEngine* ve = GIPSVoiceEngine::Create();
```

Both methods return a pointer to a new VoiceEngine object with one important difference: the new VoiceEngine pointer in 3.x does not give access to any API functions as is. It must first be provided as input to a static function that returns the specified interface, or sub-API:

```
GIPSVEBase* base = GIPSVEBase::GIPSVE_GetInterface(ve);
GIPSVEVQE* vqe = GIPSVEVQE::GetInterface(ve);
```

See the Acquiring Sub-APIs section for more details.

Note that, VoiceEngine 2.x also supports an alternative allocation method

```
GipsVoiceEngineLib &GetGipsVoiceEngineLib();
```

where the returned object is a singleton instance that does not need to be created or destroyed. Instead, the user simply acquires a reference, or alias, to an already existing (global) VoiceEngine instance. To exemplify:

```
GipsVoiceEngineLib& veSingelton = GetGipsVoiceEngineLib();
```

Where `veSingleton` is destroyed as soon as its owner is destroyed.

Acquiring Sub-APIs

The VoiceEngine pointer in 2.x gives immediate access to all API functions in the complete API. Version 3.x requires an additional step to get access to the new, so called, sub-APIs. The rationale for dividing the API into smaller sub-APIs, or interfaces, is to simplify the usage. Continuing the resource allocation example above, the following steps are now required:

```
#ifdef GIPS_VE_2_X
    #include "GipsVoiceEngineLib.h"
```

```
#else
    #include "GIPSVEErrors.h"
    #include "GIPSVEBase.h"
    #include "GIPSVEVQE.h"
#endif
```

```
#ifndef GIPS_VE_2_X
    GIPSVEBase* base = GIPSVEBase::GIPSVE_GetInterface(ve);
    GIPSVEVQE* vqe = GIPSVEVQE::GetInterface(ve);
#endif
```

Each sub-API has its own internal reference counter which is increased when the sub-API is acquired and decreased when it is released. See Resource Deallocation for more details.

NOTE: The `GIPSVEBase` instance is mandatory and its `GetInterface()` method is therefore prefixed by `GIPSVE_` to make its name unique. All other sub-APIs are optional and may be used when needed.

These interface pointers give access to new sub-APIs, where each API is declared in individual header files. The code snippet below illustrates the main differences in how to use the 2.x API and the 3.x API:

```
#ifdef GIPS_VE_2_X
    ve->GIPSVE_SetTrace(2);
    ve->GIPSVE_SetNRStatus(1);
    ve->GIPSVE_SetNRpolicy(2);
#else
    base->GIPSVE_SetTraceStatus(true, TRACE_FILE);
    vqe->GIPSVE_SetNSStatus(true, NS_HIGH_SUPPRESSION);
#endif
```

The short example above illustrates some of the main differences between 2.x and 3.x. Each sub-API in 3.x needs it own unique interface pointer. Enumerators are used more in 3.x instead of integers, and some old APIs have been merged from two APIs into one combined.

Resource Deallocation

VoiceEngine 2.x and 3.x use different ways to deallocate resources.

From 3.x and onwards, the acquired interfaces must first be released.

```
#ifndef GIPS_VE_2_X
    vqe->Release();
    base->GIPSVE_Release();
#endif
```

Releasing the interfaces is mandatory in 3.x and the destruction will not be completed unless all `GetInterface()` calls are matched with a corresponding `Release()` call. Each sub-API has its own internal reference counter which is increased when the sub-API is acquired and decreased when it is released. The return value of all `Release()` methods equals the value of the internal reference count. When the reference count of all sub-APIs reaches zero, `GIPSVoiceEngine::Delete()` can safely be performed to release the allocated resources.

Finally, the actual memory deallocation can be performed:

```
#ifdef GIPS_VE_2_X
    DeleteVoiceEngineLib(ve);
#else
    bool ok = GIPSVoiceEngine::Delete(ve);
    if (!ok)
    {
        printf("ERROR: all interfaces must be released first\n");
    }
#endif
```

It is essential to verify that all sub-APIs are properly released (reference counter is zero) before attempting to delete the VoiceEngine instance. If this was not the case in the example above, `GIPSVoiceEngine::Delete(ve)` would return `false` and memory would leak. The user must always ensure that `GIPSVoiceEngine::Delete(ve)` returns `true`.

The `DeleteVoiceEngineLib()` call is not required for VE 2.x if the singleton allocation method (see Resource Allocation above) is used instead.

Header Files

Each sub-API (or interface) in VE 3.x is declared in its own header file, contrary to VE 2.x, where all API methods are declared in one large header file. This change has been done mainly to improve the readability and to make it easier to get an overview of the API and its functions.

The code snippet below illustrates the difference between VE 2.x and 3.x:

```
#ifdef GIPS_VE_2_X
    #include "GipsVoiceEngineLib.h"
#else
    #include "GIPSVEErrors.h"
    #include "GIPSVEBase.h"
    #include "GIPSVECodec.h"
    #include "GIPSVEVQE.h"
    // add more sub-APIs here if needed
#endif
```

Error codes are declared in a separate file called GIPSVEErrors.h in VE 3.x, while GipsVoiceEngineLib.h contains both the API and the error codes for VE 2.x.

Merged APIs

Some old API methods have been merged, or combined, into one API method in the new VE 3.x.

The example below enables GIPS Noise Suppression and selects the high-suppression mode.

```
#ifdef GIPS_VE_2_X
    ve->GIPSVE_SetNRStatus(1);
    ve->GIPSVE_SetNRpolicy(2);
#else
    vqe->GIPSVE_SetNSStatus(true, NS_HIGH_SUPPRESSION);
```

```
#endif
```

The old declarations are given by

```
int GIPSVE_SetNRStatus(int mode);
int GIPSVE_SetNRpolicy(int mode);
```

are now combined into one new API

```
int GIPSVE_SetNSStatus(bool enable, GIPS_NSmodes mode = NS_LOW_SUPPRESSION);
```

where the `GIPS_NSmodes` enumerator is defined as

```
enum GIPS_NSmodes
{
    NS_LOW_SUPPRESSION = 0,
    NS_MODERATE_SUPPRESSION,
    NS_HIGH_SUPPRESSION,
    NS_VERY_HIGH_SUPPRESSION
};
```

See the Enumerators section for more information regarding enumerators in VE 3.x.

API Name Changes

The names of some APIs are modified in VE 3.x. The intention is to make the function of the API clearer and also to make the API more consistent.

The old 2.x signature

```
int GIPSVE_PlayPCM();
```

has been modified to

```
GIPSVE_StartPlayingFileLocally();
```

in VE 3.x to clarify that other file types than PCM can be played out. Version 3.x also includes a corresponding Stop-method.

In addition, the old 2.x API called

```
GIPSVE_GetNoOfChannels();
```

has been changed to

```
GIPSVE_MaxNumOfChannels();
```

The `Get`-prefix is removed from all API-functions in VE 3.x, which returns a positive integer as output instead of the default error notifications `0` or `-1`. Instead, `Get`-functions in 3.x return their results via output reference parameters, and the actual return value signals only if the call was successful or not.

Examples of new Get-functions in VE 3.x are:

```
int GIPSVE_GetTraceStatus(bool& enabled, GIPS_TraceModes& mode);
int GIPSVE_GetCodec(int index, GIPS_CodecInst& codec);
int GIPSVE_GetAGCStatus(bool& enabled, GIPS_AGCmodes& mode);
```

Enumerators

Enumerators are introduced to a higher degree in VE 3.x, with the intention to make the API less error-prone. This approach also improves readability of the final application code.

This old API in VE 2.x

```
GIPSVE_SetTrace(int mode);
```

has been modified to

```
GIPSVE_SetTraceStatus(bool enable, GIPS_TraceModes mode = TRACE_FILE);
```

in VE 3.x, which explains the API function better.

To compare:

```
#ifdef GIPS_VE_2_X
    ve->GIPSVE_SetTrace(2);
#else
    base->GIPSVE_SetTrace(true, TRACE_FILE);
#endif
```

Appendix A Generic Code Example

```
// VoiceEngine_GenericApp.cpp
//
//      This console-based test application exemplifies the main differences
//      between using VoiceEngine version 2.x and 3.x (default).
//      Note that error handling has been excluded from this example.
//
//      Copyright (c) 1999-2008 Global IP Solutions. All rights reserved.

#include <stdio.h>
#include <conio.h>

// Define the GIPS_VE_2_X flag to use VoiceEngine 2.x
//
// #define GIPS_VE_2_X

#ifdef GIPS_VE_2_X
#include "GipsVoiceEngineLib.h"
#else
#include "GIPSVEErrors.h"
#include "GIPSVEBase.h"
#include "GIPSVECodec.h"
#include "GIPSVEVQE.h"
#endif

#define PAUSE()                             \
{                                           \
    printf("\nPress any key to continue");  \
    char c = _getch();                      \
    printf("\n\n");                         \
}

enum MarkType
{
    MARK_NONE = 0,
    MARK_CODEC,
    MARK_NS
};

#ifdef GIPS_VE_2_X
int UpdateState(MarkType mark, GipsVoiceEngineLib* voiceEngine, int& caseNumber, const
char* strMessage);
#else
int UpdateState(MarkType mark, GIPSVoiceEngine* voiceEngine, int& caseNumber, const char*
strMessage);
#endif

// ----------------------------------------------------------------------------
//      Main
// ----------------------------------------------------------------------------
```

```
int main(int argc, char* argv[])
{
    printf("GIPS VoiceEngine Generic Test Application\n");
    printf("========================================\n\n");

#ifdef GIPS_VE_2_X
    printf("Test is based on VE 2.x\n\n");
#else
    printf("Test is based on VE 3.x\n\n");
#endif

    ////////////////
     // Construction

#ifdef GIPS_VE_2_X
    GipsVoiceEngineLib* ve = GetNewVoiceEngineLib();
#else
    GIPSVoiceEngine* ve = GIPSVoiceEngine::Create();
#endif

    /////////////////////////////////////////////////////////////
     // Acquire API interfaces, or sub-APIs (applies to 3.x only)

#ifndef GIPS_VE_2_X
    GIPSVEBase* base = GIPSVEBase::GIPSVE_GetInterface(ve);
    GIPSVECodec* codec = GIPSVECodec::GetInterface(ve);
    GIPSVEVQE* vqe = GIPSVEVQE::GetInterface(ve);
#endif

    ////////////////
    // Preparations

#ifdef GIPS_VE_2_X
    ve->GIPSVE_SetTraceFileName("GIPSVE_2_X_trace.txt");
    ve->GIPSVE_SetTrace(2);
    ve->GIPSVE_SetNetworkStatus(0);
#else
    base->GIPSVE_SetTraceFileName("GIPSVE_3_X_trace.txt");
    base->GIPSVE_SetTraceStatus(true, TRACE_FILE);
#endif

    //////////////////
    // Initialization

#ifdef GIPS_VE_2_X
    ve->GIPSVE_Init();
    ve->GIPSVE_CreateChannel();
#else
    base->GIPSVE_Init();
    base->GIPSVE_CreateChannel();
#endif

    ////////////////////////////////////////
    // Call setup for full-duplex streaming

#ifdef GIPS_VE_2_X
    ve->GIPSVE_SetSendIP(0, "127.0.0.1");
    ve->GIPSVE_SetSendPort(0, 12345);
    ve->GIPSVE_SetRecPort(0, 12345);
#else
    base->GIPSVE_SetLocalReceiver(0 , 12345);
    base->GIPSVE_SetSendDestination(0, 12345, "127.0.0.1");
#endif

    int count(1);

    ///////////////////////////////////////////
    // Case #1
    //
    // Full duplex using default codec (PCMU)

#ifdef GIPS_VE_2_X
    ve->GIPSVE_StartListen(0);
    ve->GIPSVE_StartPlayout(0);
    ve->GIPSVE_StartSend(0);
#else
    base->GIPSVE_StartListen(0);
    base->GIPSVE_StartPlayout(0);
    base->GIPSVE_StartSend(0);
#endif

    UpdateState(MARK_NONE, ve, count, "Full duplex using default codec"); PAUSE();

    ////////////////////////////////////////
    // Case #2
    //
    // Switch to non-default sending codec

    const GIPS_CodecInst cinst = {97,"IPCMWB",16000,320,1,80000};
#ifdef GIPS_VE_2_X
    ve->GIPSVE_SetSendCodec(0, &cinst);
#else
    codec->GIPSVE_SetSendCodec(0, cinst);
#endif
    UpdateState(MARK_CODEC, ve, count, "Switch to non-default sending codec"); PAUSE();

    /////////////////////////////////////////////
    // Case #3
    //
    // Enable non-default Noise Suppression (NS)

#ifdef GIPS_VE_2_X
    ve->GIPSVE_SetNRStatus(1);
    ve->GIPSVE_SetNRpolicy(2);
#else
    vqe->GIPSVE_SetNSStatus(true, NS_HIGH_SUPPRESSION);
#endif
    UpdateState(MARK_NS, ve, count, "Enable non-default Noise Suppression (NS) mode");
PAUSE();

    //////////////////
    // Stop streaming

#ifdef GIPS_VE_2_X
```

```
    ve->GIPSVE_StopListen(0);
    ve->GIPSVE_StopPlayout(0);
    ve->GIPSVE_StopSend(0);
#else
    base->GIPSVE_StopSend(0);
    base->GIPSVE_StopPlayout(0);
    base->GIPSVE_StopListen(0);
#endif

    ///////////////
    // Termination

#ifdef GIPS_VE_2_X
    ve->GIPSVE_DeleteChannel(0);
    ve->GIPSVE_Terminate();
#else
    base->GIPSVE_DeleteChannel(0);
    base->GIPSVE_Terminate();
#endif

    ////////////////////////////////////////////////////////////
    // Release the interfaces, or sub-APIs (applies to 3.x only)

#ifndef GIPS_VE_2_X
    vqe->Release();
    codec->Release();
    base->GIPSVE_Release();
#endif

    ///////////////
    // Destruction

#ifdef GIPS_VE_2_X
     DeleteVoiceEngineLib(ve);
#else
    bool ok = GIPSVoiceEngine::Delete(ve);
    if (!ok)
    {
        printf("ERROR: all interfaces must be released first\n\n");
    }
#endif

    return 0;
}

// ----------------------------------------------------------------------------
//      UpdateState
// ----------------------------------------------------------------------------

#ifdef GIPS_VE_2_X
int UpdateState(MarkType mark, GipsVoiceEngineLib* ve, int& caseNumber, const char*
strMessage)
#else
int UpdateState(MarkType mark, GIPSVoiceEngine* ve, int& caseNumber, const char*
strMessage)
#endif
{
```

```
#ifndef GIPS_VE_2_X
    GIPSVEBase* base = GIPSVEBase::GIPSVE_GetInterface(ve);
    GIPSVECodec* codec = GIPSVECodec::GetInterface(ve);
    GIPSVEVQE* vqe = GIPSVEVQE::GetInterface(ve);
#endif

    printf("Case #%d: %s\n\n", caseNumber, strMessage);

    GIPS_CodecInst cinst;
#ifdef GIPS_VE_2_X
    ve->GIPSVE_GetCurrentSendCodec(0, &cinst);
#else
    codec->GIPSVE_GetSendCodec(0, cinst);
#endif

    int enabled(0);
    int mode(0);
#ifdef GIPS_VE_2_X
    enabled = ve->GIPSVE_GetNRStatus();
    mode = ve->GIPSVE_GetNRpolicy();
#else
    vqe->GIPSVE_GetNSStatus((bool&)enabled, (GIPS_NSmodes&)mode);
#endif

    printf("  cinst   : name=%s, size=%d, fs=%d", cinst.plname, cinst.pacsize,
cinst.plfreq);
    if (MARK_CODEC == mark) printf(" (*)\n"); else printf("\n");

    printf("  NS      : enabled=%d, mode=%d", enabled, mode);
    if (MARK_NS == mark) printf(" (*)\n"); else printf("\n");

    caseNumber++;

#ifndef GIPS_VE_2_X
    vqe->Release();
    codec->Release();
    base->GIPSVE_Release();
#endif
    return 0;
}

// EOF
```

Appendix B Modified Interfaces

This appendix summarizes commonly used interfaces that have been modified in VoiceEngine 3.x. It is not a complete list of all changes but the bulk part is covered and the main new design principles are illustrated.

NOTE: All notes in the table below are focused on VE 3.x; i.e., if nothing else is stated, VE 3.x is implicitly assumed. Default return types are integers (0 or -1).

Interface(s) in VE 2.x	Corresponding interface in VE 3.x	Notes
GipsVoiceEngineLib	**GIPSVEBase**	Declared in GIPSVEBase.h
NA	GIPSVE_GetInterface(GIPSVoiceEngine* voiceEngine)	Get an GIPSVEBase interface pointer.
NA	GIPSVE_Release()	Release the GIPSVEBase interface.
SetObserver (error_callback &observer, bool clear=false)	GIPSVE_SetObserver(GIPSVoiceEngineObserver& observer, bool clear = false)	New observer definition and GIPSVE_ prefix added.
GIPSVE_Init(bool recordAEC =false, bool multiCore = false,int month = 0,int day = 0,int year = 0, int audiolib=1)	GIPSVE_Init(int month = 0, int day = 0, int year = 0, bool recordAEC = false, bool multiCore = false, GIPS_LinuxAudio audiolib = LINUX_AUDIO_ALSA)	Parameters reorganized. Introduced GIPSVE_LinuxAudio enumerator with clearer default value.
GIPSVE_SetTrace(int mode)	GIPSVE_SetTraceStatus(bool enable, GIPS_TraceModes mode = TRACE_FILE)	New API name. Added boolean enable parameter. New enumerator GIPS_TraceModes added to simplify the usage.
NA	GIPSVE_GetTraceStatus(bool& enabled, GIPS_TraceModes& mode)	New API. Does not exist in VE 2.x.
GIPSVE_SetNetworkStatus(int networktype)	NA	Removed API.
GIPSVE_GetNoOfChannels()	GIPSVE_MaxNumOfChannels()	Get-prefix excluded. Method returns value instead of 0 or -1.
GIPSVE_SetRecPort(int channel, int portnr, char * multiCastAddr = NULL, char * ip = NULL, int RTCPport =0)	GIPSVE_SetLocalReceiver(int channel, int port, int RTCPport = GIPS_DEFAULT, const char* ipaddr = NULL, const char* multiCastAddr = NULL)	New API name. Reorganized and renamed parameters. GIPS_DEFAULT used as default-value indicator instead of 0.
GIPSVE_SetSendPort(int channel, int portnr, int RTCPport =0)	GIPSVE_SetSendDestination(int channel, int port, const char* ipaddr, int sourcePort = GIPS_DEFAULT, int RTCPport = GIPS_DEFAULT)	Three old APIs merged into one. GIPS_DEFAULT used as default-value indicator instead of 0.
GIPSVE_SetSendIP(int channel, char *ipadr)		
GIPSVE_SetSrcPort(int channel, int portnr)		
GIPSVE_GetLastError()	GIPSVE_LastError()	Get-prefix excluded. Method returns value instead of 0 or -1.
GIPSVE_AddToConference(int channel,bool enable, bool includeCSRCs = false, bool includeVoiceLevel = false)	GIPSVE_SetConferenceStatus(int channel, bool enable, bool includeCSRCs = false, bool includeVoiceLevel = false)	New API name.
	GIPSVECodec	Declared in GIPSVECodec.h
GIPSVE_GetNofCodecs()	GIPSVE_NumOfCodecs()	Get-prefix excluded. Method returns value instead of 0 or -1.
GIPSVE_GetCodec(short listnr, GIPS_CodecInst *codec_inst)	int GIPSVE_GetCodec(int index, GIPS_CodecInst& codec)	Renamed parameters. Reference parameter introduced.
GIPSVE_SetAMR_enc_format(int channel,int mode)	GIPSVE_SetAMREncFormat(int channel, GIPS_AMRmodes mode = AMR_RFC3267_BWEFFICIENT)	Introduced GIPS_AMRmodes enumerator.
GIPSVE_SetSendCNPayloadType(int	GIPSVE_SetSendCNPayloadType(int channel, int type,	Introduced

channel,short payloadType, int payloadFreq = 8000)	GIPS_PayloadFrequencies frequency = FREQ_8000_HZ)	GIPS_PayloadFrequencies enumerator.
GIPSVE_SetVADStatus(int channel, int enable, int mode = 0, bool disableDTX = false)	GIPSVE_SetVADStatus(int channel, bool enable, GIPS_VADmodes mode = VAD_CONVENTIONAL, bool disableDTX = false)	New GIPS_VADmodes enumerator.
GIPSVE_GetVADStatus(int channel, int *mode = NULL, bool *disableDTX = NULL)	GIPSVE_GetVADStatus(int channel, bool& enabled, GIPS_VADmodes& mode, bool& disabledDTX)	Added boolean reference parameter for state output. GIPSVAD_modes enumerator added for mode output and pointer variable replaced by reference parameter.
	GIPSVEDTMF	Declared in GIPSVEDTMF.h
GIPSVE_GetDTMFFeedbackStatus()	GIPSVE_GetDTMFFeedbackStatus(bool& enabled, bool& directFeedback)	Status information is returned as reference parameter. Added directFeedback output parameter.
	GIPSVEEncryption	Declared in GIPSVEEncryption.h
GIPSVE_EnableSRTPSend(int channel,int cipher_type,int cipher_key_len,int auth_type, int auth_key_len,int auth_tag_len, int security, const unsigned char* key)	GIPSVE_EnableSRTPSend(int channel, GIPS_CipherTypes cipherType, unsigned int cipherKeyLength, GIPS_AuthenticationTypes authType, unsigned int authKeyLength, unsigned int authTagLength, GIPS_SecurityLevels level, const unsigned char* key)	GIPS_CipherTypes, GIPS_AuthenticationTypes and GIPS_SecurityLevels enumerators replace integers.
GIPSVE_EnableSRTPReceive(…)	GIPSVE_EnableSRTPReceive(…)	See GIPSVE_EnableSRTPSend()
GIPSVE_EnableEncryption(int channel)	GIPSVE_SetEncryptionStatus(int channel, bool enable)	Two old APIs merged into one.
GIPSVEDisableEncryption(int channel)		
	GIPSVEFile	Declared in GIPSVEFile.h
GIPSVE_PlayPCM(int channel, char * fileName, bool loop = false ,enum GIPS_FileFormats file_format = FILE_PCM_16KHZ, float volume_scaling = 1.0,int start_point = 0, int stop_point = 0)	GIPSVE_StartPlayingFileLocally(int channel, const char* fileName, bool loop = false, GIPS_FileFormats format = FILE_PCM_16KHZ, float volumeScaling = 1.0,int startPointMs = 0, int stopPointMs = 0)	New API name to show that not only PCM format is supported.
GIPSVE_StopPlayingFile(int channel)	GIPSVE_StopPlayingFileLocally(int channel)	New API name.
GIPSVE_IsPlayingFile(int channel)	GIPSVE_IsPlayingFileLocally(int channel)	New API name.
GIPSVE_SetFilePlayoutScaling(int channel,float scale)	GIPSVE_ScaleLocalFilePlayout(int channel, float scale)	New API name.
GIPSVE_PlayPCMAsMicrophone(int channel, char * fileName, bool loop = false , bool mixWithMic = false,enum GIPS_FileFormats file_format = FILE_PCM_16KHZ, float volume_scaling = 1.0)	GIPSVE_StartPlayingFileAsMicrophone(int channel, const char* fileName, bool loop = false , bool mixWithMicrophone = false, GIPS_FileFormats format = FILE_PCM_16KHZ, float volumeScaling = 1.0)	New API name to show that not only PCM format is supported.
GIPSVE_SetFilePlayoutScalingMic(int channel,float scale)	GIPSVE_ScaleFileAsMicrophonePlayout(int channel, float scale)	New API name.
GIPSVE_StartRecording(int channel,char * fileName,GIPS_CodecInst *gipsve_inst = NULL)	GIPSVE_StartRecordingPlayout(int channel, const char* fileName, GIPS_CodecInst* compression = NULL)	New API name.
GIPSVE_StopRecording(int channel)	GIPSVE_StopRecordingPlayout(int channel)	New API name.
GIPSVE_InitRTPToPCMConversion(const char* fileName, unsigned int conversionDelay, GIPS_CodecInst *gipsve_inst= NULL)	GIPSVE_InitRTPToFileConversion(const char* fileName, unsigned int conversionDelay, GIPS_CodecInst* compression = NULL)	New API name to show that not only PCM format is supported.
GIPSVE_StartRTPToPCMConversion(int channel)	GIPSVE_StartRTPToFileConversion(int channel)	New API name.
GIPSVE_StopRTPToPCMConversion(int channel)	GIPSVE_StopRTPToFileConversion(int channel)	New API name.
GIPSVE_ConvertRTPToPCM(int channel, char *rtpPacketBuffer, int length, unsigned long incomingTimeStamp)	GIPSVE_ConvertRTPToFile(int channel, char* rtpPacketBuffer, unsigned int length, unsigned long incomingTimeStamp)	New API name.
GIPSVE_GetFileDuration(char *filename,	GIPSVE_GetFileDuration(const char* fileName, int&	Added integer reference

enum GIPS_FileFormats file_format = FILE_PCM_16KHZ)	durationMs, GIPS_FileFormats format = FILE_PCM_16KHZ)	parameter for file duration output.
GIPSVE_GetPlaybackPosition(int channel)	GIPSVE_GetPlaybackPosition(int channel, int& positionMs)	Added integer reference parameter for playback position output.
	GIPSVEHardware	Declared in GIPSVEHardware.h
GIPSVE_GetCPULoad()	GIPSVE_GetCPULoad(int& loadPercent)	Added integer reference parameter for load output.
GIPSVE_GetSystemCPULoad()	GIPSVE_GetSystemCPULoad(int& loadPercent)	Added integer reference parameter for load output.

GIPSVE_GetNumDevsRecording()
GIPSVE_GetNumDevsPlayout()
	GIPSVE_GetNumOfSoundDevices(int& playout, int& recording)	Two old APIs merged into one combined.

GIPSVE_GetPlayoutDevName(int index, char *str, int strLen)	GIPSVE_GetPlayoutDeviceName(int index, char* strNameUTF8, int nameLen, char* strGuidUTF8 = NULL, int guidLen = 0)	Minor modification in API name. Added GUID output parameters (Windows Vista).
GIPSVE_GetRecordingDevName(int index, char *str, int strLen)	GIPSVE_GetRecordingDeviceName(int index, char* strNameUTF8, int nameLen, char* strGuidUTF8 = NULL, int guidLen = 0)	Minor modification in API name. Added GUID output parameters (Windows Vista).
GIPSVE_GrabPlayout(bool enable)	GIPSVE_SetGrabPlayout(bool enable)	Added Set to API name.
GIPSVE_GrabRecording(bool enable)	GIPSVE_SetGrabRecording(bool enable)	Added Set to API name.
	GIPSVENetwork	Declared in GIPSVENetwork.h
GIPSVE_SetSendTransport(int channel, GIPS_transport &transport)	GIPSVE_SetExternalTransport(int channel, bool enable, GIPS_transport* transport)	API name changed. Added possibility to turn on and off.

GIPSVE_GetFromIP(int channel, char *ipadr, int bufsize)
GIPSVE_GetFromPort(int channel)
	GIPSVE_GetSourceInfo(int channel, int& port, char* ipaddr, unsigned int ipaddrLength)	Two old APIs merged into one combined.

GIPSVE_SetFilterPort(int channel,unsigned short filter)
GIPSVE_SetFilterIP(int channel,char *IPaddress)
	GIPSVE_SetSourceFilter(int channel, int port, const char* ipaddr)	Two old APIs merged into one combined.

GIPSVE_GetSendTOS(int channel)	GIPSVE_GetSendTOS(int channel, int& TOS)	Added integer reference parameter for TOS output.
GIPSVE_GetSendGQOS(int channel)	GIPSVE_GetSendGQOS(int channel, bool& enable, int& serviceType, int& overrideTOS)	Added three additional reference parameters for current state output.
GIPSVE_SetPacketTimeout(int channel, bool enable, int time_sec)	GIPSVE_SetPacketTimeoutNotification(int channel, bool enable, int timeoutSeconds)	Modified API name.
sendExtraPacket_RTP(int channel, unsigned char* data, int nbytes)	GIPSVE_SendExtraRTPPacket(int channel, const void* data, unsigned int length, int& transmittedBytes)	Modified API name and added output reference parameter for number of transmitted bytes.
sendExtraPacket_RTCP(int channel, unsigned char* data, int nbytes)	GIPSVE_SendExtraRTCPPacket(int channel, const void* data, unsigned int length, int& transmittedBytes)	Modified API name and added output reference parameter for number of transmitted bytes.
	GIPSVEPTT	Declared in GIPSVEPTT.h
GIPSVE_GetPTTActivity(int channel)	GIPSVE_GetPTTActivity(int channel, bool& activity)	Added boolean reference parameter for activity output.
GIPSVE_GetPTTSession(int channel, GIPSVE_PTTState *state)	GIPSVE_GetPTTSessionInfo(int channel, GIPSVE_PTTState& state)	Changed pointer parameter to reference parameter.
GIPSVE_sendRTCP_APP(int channel,unsigned char *data, int len)	GIPSVE_SendRTCP_APP(int channel, const unsigned char* data, unsigned int length, int& sentBytes)	Minor change in API name. Number of transmitted bytes is now returned via reference parameter.
GIPSVE_subscribeRTCP_APP(int channel, bool enable , RTCP_APP_handler *callback)	GIPSVE_SetRTCP_APPCallback(int channel, bool enable, RTCP_APP_handler* callback)	Modified API name.
	GIPSVERTP_RTCP	Declared in GIPSVERTP_RTCP.h
GIPSVE_GetSendSSRC(int channel)	GIPSVE_GetSendSSRC(int channel, unsigned long& ssrc)	Added unsigned int reference parameter for SSRC output.
GIPSVE_EnableRTCP(int channel, int enable)	GIPSVE_SetRTCPStatus(int channel, bool enable)	Modified API name.
GIPSVE_SetRTCPCNAME(int channel, char *	GIPSVE_SetRTCP_CNAME(int channel, const char*	Modified API name.

str)	cname)	
GIPSVE_getRemoteRTCPCNAME(int channel, char * str)	GIPSVE_GetRemoteRTCP_CNAME(int channel, char* cname)	Modified API name.
GIPSVE_getRemoteRTCPData(int channel, unsigned long * NTP_high, unsigned long * NTP_low, unsigned long * timeStamp, unsigned long * playoutTimeStamp, unsigned long * jitter = NULL, unsigned short * fraction_lost=NULL)	GIPSVE_GetRemoteRTCPData(int channel, unsigned long& NTPHigh, unsigned long& NTPLow, unsigned long& timestamp, unsigned long& playoutTimestamp, unsigned long* jitter = NULL, unsigned short* fractionLost = NULL)	Modified API name and replaced pointer parameters with reference parameters.
GIPSVE_RTPStat(int channel, unsigned long *avg_jitter_MS, unsigned long *max_jitter_MS, unsigned long *discardedPackets)	GIPSVE_RTPStatistics(int channel, unsigned long& averageJitterMs, unsigned long& maxJitterMs, unsigned long& discardedPackets)	Modified API name and replaced pointer parameters with reference parameters.
GIPSVE_RTCPStat(int channel, GIPSVE_CallStatistics * stats)	GIPSVE_RTCPStatistics(int channel, GIPS_CallStatistics& stats)	Modified API name and replaced pointer parameter with reference parameter.
	GIPSVEVolumeControl	Declared in GIPSVEVolumeControl.h
GIPSVE_GetSpeakerVolume()	GIPSVE_GetSpeakerVolume(unsigned int& volume)	Added unsigned integer reference parameter for volume output.
GIPSVE_GetMicVolume()	GIPSVE_GetMicVolume(unsigned int& volume)	Added unsigned integer reference parameter for microphone output.
GIPSVE_GetInputLevel()	GIPSVE_GetSpeechInputLevel(unsigned int& level)	Added integer reference parameter for input speech-level.
GIPSVE_GetOutputLevel()	GIPSVE_GetSpeechOutputLevel(int channel, unsigned int& level)	Added integer reference parameter for output speech-level.
GIPSVE_GetChannelOutputVolumeScale(int channel)	GIPSVE_GetChannelOutputVolumeScaling(int channel, float& scaling)	Modified API name. Added float reference parameter for scaling output.
GIPSVE_GetWaveOutVolume()	GIPSVE_GetWaveOutVolume(unsigned int& volume)	Added unsigned integer reference parameter for volume output.
NA	GIPSVE_GetOutputVolumePan(float& left, float& right)	New API in 3.x.
GIPSVE_GetChannelOutputVolumePan(int channel, float *left, float *right)	GIPSVE_GetChannelOutputVolumePan(int channel, float& left, float& right)	Replaced pointer parameters with reference parameters.
GIPSVE_MuteMic(int channel,int Mute)	GIPSVE_SetInputMute(int channel, bool enable)	Modified API name.
NA	GIPSVE_GetInputMute(int channel, bool& enabled)	New API in 3.x.
	GIPSVEVQE	Declared in GIPSVEVQE.h

GIPSVE_SetNRStatus(int mode)
GIPSVE_SetNRpolicy(int mode)
	GIPSVE_SetNSStatus(bool enable, GIPS_NSmodes mode = NS_LOW_SUPPRESSION)	Two old APIs merged into one combined. GIPS_NSmodes enumerator returns output mode.

GIPSVE_GetNRStatus()
GIPSVE_GetNRpolicy()
	GIPSVE_GetNSStatus(bool& enabled, GIPS_NSmodes& mode)	Two old APIs merged into one combined. Reference parameters added for the combined output state.

GIPSVE_SetAGCStatus(int mode)
GIPSVE_SetAGCType(int mode)
	GIPSVE_SetAGCStatus(bool enable, GIPS_AGCmodes mode = AGC_ANALOG_DIGITAL_COMBINED)	Two old APIs merged into one combined. GIPS_AGCmodes enumerator returns output mode.

GIPSVE_GetAGCStatus()
GIPSVE_GetAGCType()
	GIPSVE_GetAGCStatus(bool& enabled, GIPS_AGCmodes& mode)	Two old APIs merged into one combined. Reference parameters added for the combined output state.

GIPSVE_SetECStatus(int mode) GIPSVE_SetECType(int type, int mode = 0,	GIPSVE_SetECStatus(bool enable, GIPS_ECmodes mode = EC_AEC, GIPS_AESmodes AESmode =	Two old APIs merged into one combined. GIPS_ECmodes and

int attn = 0)	AES_NORMAL, int AESattn = 28)	GIPE_AESmodes enumerators return output mode.

GIPSVE_GetECStatus()
GIPSVE_GetECType(int *mode = NULL, int *attn = NULL)
	GIPSVE_GetECStatus(bool& enabled, GIPS_ECmodes& mode, GIPS_AESmodes& AESmode, int& AESattn)	Two old APIs merged into one combined. Reference parameters added for the combined output state.

GIPSVE_SetConfMode(int enable, int mode = 0)	GIPSVE_SetConfStatus(bool enable, GIPS_ConfModes mode = TWO_PARTICIPANTS)	Modified API name. GIPS_ConfModes enumerator is used to set the mode.
GIPSVE_GetConfMode(int *enable, int *mode)	GIPSVE_GetConfStatus(bool& enabled, GIPS_ConfModes& mode)	Modified API name. Reference output parameters instead of pointer parameters
GIPSVE_GetECActivity()	NA	Removed API.
GIPSVE_GetVoiceActivityIndicator(int channel)	GIPSVE_VoiceActivityIndicator(int channel)	Get-prefix excluded.
	GIPSVEVQMon	Declared in GIPSVEVQMon.h
GIPSVE_EnableVQMon(int channel, bool enable)	GIPSVE_SetVQMonStatus(int channel, bool enable)	@TBW
GIPSVE_EnableRTCP_XR(int channel, bool enable)	GIPSVE_SetRTCPXRStatus(int channel, bool enable)	@TBW
GIPSVE_InstallAlertHandler(vqmon_alert alert_callback)	GIPSVE_SetVQMonAlertCallback(GIPSVE_VqmonAlert callback)	@TBW
GIPSVE_SetAlert(int channel, int type, int param1[4], int param2[4], int param3[4])	GIPSVE_EnableVQMonAlert(int channel, int type, int param1[4], int param2[4], int param3[4])	@TBW
GIPSVE_RemoveAlert(int channel, int type)	GIPSVE_DisableVQMonAlert(int channel, int type)	@TBW
GIPSVE_VQMonIPInfo(int channel, unsigned char *local_IP, int local_port,unsigned char *remote_IP, int remote_port)	GIPSVE_SetVQMonIPInfo(int channel, const unsigned char* localIP, int localPort, const unsigned char* remoteIP, int remotePort)	@TBW
GIPSVE_GetVoIPMetrics(int channel, unsigned char *dst, unsigned int bufSize)	GIPSVE_GetVoipMetrics(int channel, unsigned char* data, unsigned int& length)	@TBW