Audio Transfer Requirements
Cerence Drive Framework / CSDK

Contents

1 Purpose of this document

This document describes which requirements shall be fulfilled for a successful integration of Cerence Drive Framework (CDFW) or Cerence SDK (CSDK) on a target system regarding the audio data transfer and the behavior of the involved audio devices.

Unless stated otherwise, all requirements listed in this document are also valid for the VoCon Framework (NATP) which is the predecessor product of DDFW.

The scope of this document is audio requirements for speech dialog use cases, i.e. audio input for speech recognition and audio output for speech synthesis and playback.

The audio requirements which are described in this document assume familiarity with audio device functionality and terminology as described in the general audio transfer requirements document [1]. Please read document [1] carefully before continuing to ensure a full understanding of the following CDFW and CSDK audio requirements.

2 Basic audio integration requirements

For full service CDFW projects where Cerence has offered deep audio integration the following requirements must be fulfilled by customers. For Do-It-Yourself projects with CSDK it is the responsibility of customers to implement and test working audio device interfaces.

For an efficient audio integration of the Cerence Drive software into a customer target hardware it is mandatory that the Cerence porting team receives at least one target system for integration testing.

The target shall be shipped with a microphone which can be connected to the target and the possibility to attach speakers, headphones or other required equipment so that all relevant audio use cases can be tested after the PAL audio layer of CDFW has been ported to the target audio system.

It is crucial for an efficient and cost effective audio porting/integration by Cerence that the audio I/O is fully functional on the target system which is handed over for porting to Cerence. The target supplier shall verify properly working audio I/O before sending any target system to Cerence. Verification shall be done by running the basic audio tests described in document [2].

Besides basic input and output testing, these audio I/O tests must also prove that parallel audio capturing and playback is supported on the target, i.e. it must be possible to open all involved audio input devices and all involved audio output devices in parallel using the required sample rates and buffer sizes accordingly. In other words, while devices are in use it must be possible to open and close additional devices in parallel without disturbing the data transfer. A typical use case for this requirement is: A navigation announcement has to be played during an ongoing speech dialog.

Test results (according to [2]) shall be documented in an audio test report that shall be shipped together with the target to Cerence.

In addition, Cerence needs documentation and sample programs how to use the audio APIs on the target system.

For standard audio APIs (see Table 1 in Appendix A.2) the sample programs coming with the operating system can be used to demonstrate working audio I/O if the audio APIs to be used by CDFW are built on the same audio API and are using the required API modes. For instance:

- On Linux, using *arecord* and *aplay* to demonstrate a working ALSA interface is sufficient.
- On QNX, using *waverec* and *wave* to demonstrate a working io_audio interface is sufficient.

- On Android, the CDFW PAL audio implementation uses the OpenSLES API, which runs at C level. Therefore, it is not sufficient testing audio I/O using the java based recorders/players, since the OpenSLES implementation might have issues that cannot be seen on the JAVA layer. The customer shall provide an audio test application based on the C-API which will be used in the project. The whole implementation should be based on the simple buffer queues (as recommended by the official OpenSLES Android documentation [3]).

For “nonstandard” audio APIs (not in Table 1), Cerence needs a programmer’s reference or equivalent documentation, including examples demonstrating the non-blocking usage of such an audio API. Adapting the PAL layer to any “nonstandard” audio API will usually cause significant higher porting efforts and costs that need to be considered by Cerence Sales Engineering when creating an offer.

In general, the Cerence porting team needs well documented and tested target systems and expert support from audio engineers at customer´s side for a successful audio integration of the CDFW software through the PAL audio layer.

3 CDFW and CSDK audio transfer requirements

The following sections list the Cerence Drive requirements regarding the audio data transfer and the behavior of the involved audio devices for several use case groups.

Requirements for a specific project are determined by the required feature set (i.e. voice barge-in requested or not, single microphone or microphone array with beamforming, etc.). Generally, access to audio devices used by CDFW must not be interrupted by the target system while CDFW is using these devices.

Multi-channel audio data are processed as non-interleaved data within the Cerence Drive audio framework. Thus, for multi-channel use cases the PAL audio layer requests by default non-interleaved data from the audio device drivers to avoid unnecessary data format conversion.

Multi-channel audio data are processed within the Cerence Drive audio modules as non-interleaved data by default. Thus, it is recommended to use the non-interleaved format in case the external audio adapter supports it. If only interleaved data are supported, then the AudioInput module needs to be configured correspondingly (by setting the configuration key "interleaved_format":"true"). The AudioInput audio module will then convert the interleaved audio data received from the audio input adapter to the non-interleaved format for subsequent processing in the Cerence Drive audio framework.

3.1 Audio Input Requirements for Speech Recognition Systems

This includes 3 main variants:

- No SSE included in audio path (single channel input)
- SSE is included in the audio input path, e.g. for beam forming (multi-channel input)
- SSE is used for echo cancellation in a barge-in or wake-up-word use case (additional requirements for reference channel)

3.1.1 Requirements for Single Channel Input

1. Non-blocking efficient access to audio input with linear 16 bit samples, little endian at a sampling rate as agreed in the contract (e.g. 16000Hz or 22050Hz)
2. The recommended sample rate of the capture device is 16000Hz. To get a good tradeoff between I/O load and delay the recommended recording time per buffer is between 30ms and 130ms. It is preferred that the used frame count is a power of two. This corresponds to frame counts of 512, 1024 or 2048 frames (@16kHz) per capture buffer. If a different capture sample rate than 16000Hz is agreed on in the contract then the frame counts have to be adapted to the recommended recording times (e.g. a frame count of 1024 @22050 Hz equals 46ms).

Note that using sample rates greater than the recommended 16000 Hz will have impacts on CPU load, because the Cerence Drive audio framework needs to perform a sample rate conversion. Sample rates below 16000 Hz are not recommended, because maximum recognition accuracy is reached with 16000 Hz versions of the Vocon ASR engines.

3. All audio data shall be provided in real-time: The mean arrival time between two succeeding buffers of audio data shall be constant and identically to the physical recording time defined by the frame count and sample rate the audio device is operating on.
4. After an audio input device was started and is generating audio data only audio data should be provided which has been captured after that starting point in time.
5. It shall be ensured that Cerence Drive audio gets a sufficient high scheduler priority to be able to process the audio in a time frame that is smaller than 80% of the physical time period of an audio input buffer.
6. All captured signals must be free of sample jitter.
7. For details about audio input signal quality requirements for single channel input see also the section “Audio Quality Requirements” in the VoCon documentation [4].

3.1.2 Additional Requirements for Multi-Channel Input

All requirements for single channel input (section 3.1.1) must also be fulfilled with multi-channel input. In addition, the following requirements must be fulfilled:

1. All microphone signals must be available within the same audio device (one multi-channel input device).
2. No additional delay offset or audio sample jitter between each of the different microphone channels shall occur.
3. The relative delay between the microphone channels shall be constant without any variation.

3.1.3 Additional Audio Input Requirements for Barge-In and Wake-up-Word

All requirements for multi-channel input (section 3.1.2) must also be fulfilled if a barge-in or wake-up-word use case shall be implemented with Cerence Drive. In addition, the following requirements must be fulfilled:

1. Reference Signal
    - A suitable reference signal for the speech prompt has to be provided.

- An external reference signal of the audible speech output from the amplifier of the car audio system is highly recommended.
- The reference signal and the microphone signal(s) shall be available within the same audio device.

2. The relative delay between the microphone signal(s) and the reference signal must be constant without any variation over one complete audio session. This means that the derived relative delay for any arbitrarily time instance within this audio session is always the same. For several audio sessions the maximum relative delay between microphone and reference signal must not vary by more than +/- 5 ms.
3. Non-linear or time-variant audio processing introduced by other audio modules is not allowed in the echo path, because it results in poor acoustic echo cancelation performance. The echo path consists of the following sub audio paths:
   - Cerence Drive audio framework → loudspeaker (including the loudspeaker amplifier and the loudspeaker itself)
   - Microphone(s) → Cerence Drive audio framework
   - Reference signal → Cerence Drive audio framework
4. All audio devices shall be synchronized to the same audio base clock and as a result be totally synchronized to each other. In other words, all audio devices which Cerence Drive audio is using shall be incorporated within the same clock domain of the customer audio system. No audio signal drift between any audio signals regarding barge-in shall occur.

For details about audio input signal quality requirements for multi channel input and barge-in see the SSE audio requirements documentation [5].

3.2 Audio Output Requirements for Text-To-Speech

1. Non-blocking efficient access to audio output with linear 16 bit little endian at a sampling rate as agreed in the contract (e.g. 22050Hz).
2. The recommended sample rate of the playback device is 22050 Hz. To get a good tradeoff between I/O load and delay the recommended playback time per buffer is between 40ms and 200ms. It is preferred that the used frame count is a power of two. This corresponds to frame counts of 1024, 2048 or 4096 frames (@22050Hz) per playback buffer.
   - If a different playback sample rate than 22050Hz is agreed on in the contract then the frame counts have to be adapted to the recommended playback times (e.g. a frame count of 2048 @48000 Hz equals 42ms).
     Note that a different sample rate than the recommended 22050 Hz will have impacts on CPU load, because the Cerence Drive audio framework then needs to

perform a sample rate conversion. Sample rates below 22050 Hz are not recommended, because speech audio output quality will decrease considerably.

3. All audio data shall be consumed in real-time: The mean consumption time between two succeeding buffers of audio data shall be constant and identically to the physical playback time defined by the frame count and sample rate the audio device is operating on.
4. It shall be ensured that Cerence Drive audio gets a sufficient high scheduler priority to be able to generate the audio in a time frame that is smaller than 80% of the physical time period of an audio output buffer.

A Appendix

A.1 References

Label	Document
[1]	General_Audio_Transfer_Requirements.pdf
[2]	Framework_platform_customer_tests.pdf
[3]	http://mobilepearls.com/labs/native-android-api/ndk/docs/opensles/
[4]	Audio Hardware Recommendations For Nuance´s Vocon 3200 ASR Engine (application note included in VoconHybrid SDK user documentation)
[5]	CerenceSSEAudioRequirements_HandsFree_BargeIn_WuW.pdf

A.2 Supported standard audio driver

OS	Description of audio driver
Windows/Windows CE	WinMM, alias Waveform Audio
Linux	ALSA using save ALSA subset
QNX	IO-Audio, alias lib-asound
Android	OpenSLES

Table 1: Supported standard audio drivers

A.3 History of Change

Version	Date	Editor	Status	Description
0.1	20.09.2013	Niels Does	Draft	First Version
0.2	29.10.2013	Niels Does	Draft	Review with SSE team
0.3	19.11.2013	Niels Does	Draft	Review third iteration
04.	10.12.2013	Niels Does	Draft	Review fifth iteration
0.5	21.01.2015	G. Hanrieder	Draft	Reworked chapters 1,2
0.6	27.01.2015	G. Hanrieder A. Kirbach M. Schick	Draft	Prepare final review part I
0.7	29.01.2015	G. Hanrieder A. Kirbach M. Schick	Draft	Prepare final review part II
0.8	30.01.2015	G. Hanrieder	Draft	Candidate for final review
0.9	03.02.2015	G. Hanrieder	Draft	Corrections from reviewers
1.0	04.02.2015	G. Hanrieder	Final	Reviewed by DragonDrive and SSE teams
1.1	16.02.2015	G. Hanrieder	Final	Adding reference to SSE requirements document [5]

1.2	25.08.2015	G. Hanrieder	Final	Minor changes: Clarified scope (no handsfree requirements) Replaced the term“SOW” with “contract” Add hint to higher porting costs for “nonstandard” audio APIs in chapter2 Removing ASRC requirement from 3.1.2
1.3	14.09.2015	G. Hanrieder B. Vater	Final	Chapter 3: Adding requirement that multi-channel use cases expect non-interleaved data
1.4	07.09.2016	G. Hanrieder	Final	Chapter 3: Adding that interleaved audio data are also supported since 2016_cw38 on the DDFW main development branch
1.5	11.07.2019	G. Hanrieder	Final	Increase scope to Companion SDK Minor corrections and updating references
2.0	16.12.2019	G. Hanrieder	Final	Migrate document from Nuance to Cerence