System Requirements for Porting Cerence Drive Framework / CSDK

Contents

1 Scope of Document

This document describes the requirements that any customer target system must fulfill for successful porting and integration of the Cerence Drive Framework (CDFW) software. Unless stated otherwise, all requirements listed in this document are also valid for porting the Cerence SDK.

This document is relevant for the following parties:

- Cerence Sales Engineering (SE): this document shall be referenced in any SOW offering a custom port of CDFW or CSDK. By signing the SOW, the customer needs to confirm that his target system meets all relevant requirements listed hereunder.
- Cerence Professional Services (PS): this document may be used by PS to verify with the customer on his target system that requirements are fulfilled before porting can start.
- Customer Engineers: this document shall allow engineers on customer side to analyze whether Cerence Drive software can be ported and deployed to their target system environment.

The main purpose of this document is to communicate the Cerence Drive system requirements to customers in pre-sales phases to determine whether the Cerence Drive SW can be ported to the customer system. In case of an offer, this document must become part of the SOW to get a confirmation that the requirements are understood and will be fulfilled.

2 Porting Requirements

The Cerence Drive software is implemented in ANSI C and in C++.

High portability of the software is ensured by encapsulating any platform-specific calls into a platform abstraction layer (PAL).

To port the Cerence Drive software to a new platform it must support the following requirements:

2.1 Compiler

2.1.1 C compiler

- ANSI C, at least ISO C89
- Source-level Debugger and Performance Profiler supporting conditional breakpoints, single-stepping, display of memory contents, etc.
- All C data types need to have at least 32 bits, except char and short.
- Heap, including *realloc()* without knowledge of the previous size of the heap block. No specific memory allocation requirements beyond total size in the predefined space.
- Standard C runtime library support, including a complete math library and file I/O, and floating point support (for SSE and biometry).

2.1.2 C++ compiler

- At least conforming to (ISO/IEC 14882:2003)

The compiler and linker must support these main features that are used by Cerence Drive:

- template support
- partial template specialization
- creating shared libraries
- exceptions (only required if biometry is included)
- STL (only required if biometry is included)

2.2 CPU

- The target system must support atomic operations (to be called from the PAL layer):
- Ideally, this is provided as a single assembler instruction of the CPU. This instruction needs to be accessible from the C language (e.g. inline assembler function or intrinsic function).
- As an alternative, atomic operations could be provided as an OS call: for example, in ARMV6 on Linux it is usually a Kernel Call, because the CPU does not support the necessary operations natively, the scheduler must be involved to guarantee atomicity.

- On x86-based systems, support for the Streaming SIMD Extensions 2 (SSE2) instruction set is required.

2.3 Operating System

The target system must provide a pre-emptive multitasking OS. It must not impose any restrictions on maximum time for returning from functions.

To ensure portability of the Cerence Drive software to the target, the integration of a Platform Abstraction Layer (PAL) from Cerence is necessary. The PAL is architecturally close to POSIX and/or Win32 API based systems, so if the target´s OS API complies to POSIX, it is very likely that porting can be done without too much effort. To port the PAL, the following OS features are required:

- Threads with priorities, real-time scheduling is required for audio. Three priority levels must be supported: real-time threads (e.g. for audio), normal standard priority threads (e.g. for the execution of recognition, TTS and dialog), and long term background threads with a low priority (e.g.for G2P tasks).
- Mutexes (timeout required, not recursive)
- Semaphores (timeout required)
- TCP networking (sockets, server and client side), also necessary during tuning sessions (e.g. for barge-in) in late project phases
- File system / directory handling (C library file access (fopen, …), directory iteration, creation and removal)
- Raw IO (open/read/write/close via file descriptors, without going through the C-library). If it does not exist it may be possible to emulate this via *fopen* etc., but this will impose a large performance penalty in the context of Cerence Drive.
- Select/Poll call for file descriptors (optional)
- Shared library loading under program control (*dlopen, dlsym*, …), incl. C++ shared libraries
- Some means for printing to a console or *std output*
  (required only during development)
- Accurate timer facility (for retrieving time stamps as well as sleeping for a defined time (at least at millisecond resolution). Memory mapping of files (*mmap*)
- Shared memory (between processes, *shmem*)

2.4 Target Environment

- The target system must support diagnostic and informational output, similar to *printf()*.
- It is necessary to have access to the file-system from outside of the target. This is needed to change or add configuration and parameter files during development and testing. Typically this will also be used during tuning sessions in late project phases (e.g. barge-in tuning for specific car-lines which are not available earlier).

CDFW runs as an executable on the target. It needs the following privileges for operation and imposes some limitations in the configuration:

1. Read and write access to all file systems (flash, RAM, HDD, ..) and to the audio system.
2. The process must have permission to change thread priorities (incl. real-time priorities).
3. Access to TCP/IP for IPC communication (if customer application uses the default TCP/IP-based implementation to connect to the CDFW process).
4. Access to the systems real time clock, that provides current time and date. (Needed e.g. for the server certificate handling)

For the initial testing of CDFW 500 MB RAM and 3 GB in the file-system must be available.

Required footprint at runtime is highly dependent on the requested feature set of a specific project and must be determined on this basis.

2.5 Toolchain / Build environment

CDFW Build infrastructure is based on a VMWare vSphere infrastructure allowing a continuous integration/build for multiple projects and branches.

Project specific toolchains to build CDFW components are installed on that CI infrastructure. Therefore toolchains must be provided in a “pure” form.

Project specific toolchains encapsulated in VMs or pre-installed on PCs cannot be deployed on our infrastructure and are therefore not accepted.

2.5.1 Toolchain License

Should a license be needed for a project specific toolchain, we need static licenses that can be installed on our build nodes.

At least, one static license has to be provided by customers. By the nature of our continuous build setup and the various build variants we deliver, we recommend to provide 3 static licenses that can be used in parallel to speed up SW builds.

The use of dynamic licenses that are retrieved via VPNs or unsecure network connections to customer license servers cannot be accepted due to data security concerns and stability issues introduced to our CDFW build process.

Security and integrity of our build infrastructure and the sensitive data hosted there cannot be guaranteed, if this host establishes TCP/IP connections to servers outside the Cerence network. This will also introduce security risks on customers side so dynamic licenses will not be accepted.

The provided license must be valid during the complete project runtime. Restricted evaluation licenses cannot be accepted.

2.5.2 Documentation

Documentation that describes the installation and setup of customer specific toolchains in details must be provided along with the toolchain itself.

2.6 Update of Server Certification Bundle on Target

If CDFW is used for remote applications that connect to Cerence Cloud Services (NCS) then – starting with VoConHybrid 4.10 – server authentification is required without exception and the target system must support certificate updates.

For security reasons updates of the certificate bundle on the client platform may be needed. Such updates can be requested by Cerence at any time during service lifetime to comply with security standards. The client platform must be able to handle such updates of the certificate bundle in use at any time during service lifetime as directed by Cerence.

It is **not** Cerence's responsibility to update the certificate bundle in use on the client platform.

2.7 Audio

CDFW is usually offered in “full service” projects including deep audio integration on the target system. This audio integration is done through the “PAL audio” software layer which accesses the target audio. Successful audio integration requires that the target audio system meets the Cerence Drive audio requirements.

The Cerence Drive audio requirements are described in the separate document [1] which assumes familiarity with basic audio device functionality and terminology as described in [2].

3 Checklist

To analyze the feasibility and required efforts for porting the Cerence Drive software to a specific customer target platform, the following information shall be provided by the customer:

General hardware description incl. documentation how to setup the target	
Processor speed	
RAM	
Flash (reading speed)	
Processor speed available to ASR/TTS/SSE	
Audio Driver Architecture and Version (e.g Alsa, OpenSLES, WinMM, …) incl. documentation how to access which audio devices	
Target Operating System (Name + Version)	
Compiler Host OS (Windows or Linux, 32/64 bit?)	
Debugger	
C++ Compiler (Name + Version)	
C Compiler (Name + Version)	
Microphone	1 or 2 microphones, directional
Audio Input	
Audio Output	

A Appendix

A.1 References

Label	Document
[1]	Audio_Transfer_Requirements_CDFW_CSDK.pdf
[2]	Audio_Transfer_Requirements_General.pdf

A.2 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
0.4.	10.12.2013	Niels Does	Draft	Review fifth iteration
0.5	20.01.2015	Gerhard Hanrieder / Stefan Sablatnög	Draft	Review and update of DD porting requirements Candidate for final review
0.6	02.02.2015	G. Hanrieder	Draft	Changed title to clarify that requirements hold for DDFW and DDSDK
1.0	04.02.2015	G. Hanrieder	Final	Reviewed by DragonDrive PM and teams
1.1	21.08.2015	G. Hanrieder	Update	Increase scope to VoCon Framework (NATP)
1.2	20.05.2016	G. Hanrieder / J. Anastasiadis	Update	Added section 2.6 on server certificate update

1.3	06.09.2016	B. Vater	Update	Add chap. 2.5 Toolchain / Build Environment
1.4	09.11.2017	G. Hanrieder	Update	Adding requirement for floating point support in chap. 2.1.1
1.5	20.06.2018	B. Vater	Update	Add need of Real Time Clock in Chapter 2.4
1.6	14.05.2019	G. Hanrieder	Update	Extend chapter 2.2 with SSE2 requirement on x86
1.7	10.07.2019	G. Hanrieder	Update	Increase scope to Companion SDK
2.0	13.12.2019	G. Hanrieder	Update	Migrate document from Nuance to Cerence
2.0.1	24.01.2020	G. Hanrieder	Update	Correcting file name in reference [1]