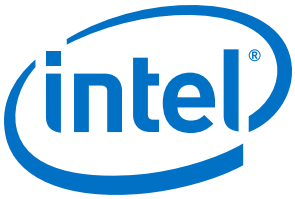


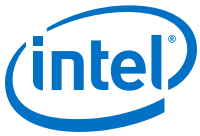
Visual Cloud Accelerator Card - Rendering Software Installation Guide

User Guide

Rev. 3.0

December 2019

Order No.: 611522-3.0



Contents

Revision History

Date	Revision	Description
December 2019	3.0	1.1 Introduction on page 7: Updated section for name change of Android solution. 1.2 Hardware Configuration on page 8: Added links to hardware specifications datasheet and Celestica support. 1.3 Build Machine Requirement on page 9: Updated link to docker installation. 1.6.2.2 Setup nginx/RTMP Server on page 12: Added more details about nginx and RTMP module. 2.1.2 VCAC-R Contents of the Release Package on page 22: Updated links to various components for the release. 2.1.3 Hardware Configuration on page 23: Added links to hardware specifications datasheet and Celestica support. 2.1.4 Software Configuration on page 24: Updated version information in table. 2.2.3 Build Machine Requirement on page 25: Added additional links. 2.2.5 Build Host Kernel and VCAC-R Driver on page 26: Updated links for the current release. 2.2.6.1 Windows Tools on page 27: Updated links to latest components. 2.2.7.2 VCAC-R Software Installation on Host on page 30: Updated commands for filenames. 2.2.7.3 Accessing VCAC-R Card via Host Using DHCP Setup on page 31: Updated Commands. 2.2.7.4 VCAC-R Card Boot up with vcad Image on page 32: Updated commands. 2.2.8 PXE Feature Description on page 33: Added a new chapter. 2.3 Additional Specific Software Requirements on page 34: Updated section. Appendix B Setting up Radeon™ GPU for Gaming on page 67: Added new chapter.
October 2019	2.1	1.1 Introduction on page 7: Added architecture diagram. 1.2 Hardware Configuration on page 8: Updated CPU information. 1.3 Build Machine Requirement on page 9: Added new chapter. 1.4 Setup Proxy for Docker (if needed) on page 9: Added new chapter. 1.5.1 README.txt on page 9: Added new section. 1.5.9 build.sh on page 10: Updated kernel version information. 1.6.1 Preparation on page 10: Updated directory structure. 1.6.2.1 Update Host Kernel on page 12: Updated version of linux kernel. 1.6.2.4 Install VCA Software on page 14: Added instructions for updating BIOS And EEPROM. 1.6.2.5 Start ICR Service in VCA Node on page 16: Updated commands and text for clarity. 1.7.1.1 Initiate Streaming Service in VCA Node on page 17: Updated commands. 2.1.1 Architecture on page 21: Updated architecture diagram to reflect support for DX9. 2.1.3 Hardware Configuration on page 23: Updated CPU information. 2.2.6.1 Windows Tools on page 27: Added more details about installation components. 2.2.7.3 Accessing VCAC-R Card via Host Using DHCP Setup on page 31: Added new topic. 2.3 Additional Specific Software Requirements on page 34: Added information about Vulkan SDK and chrome browser. 2.5 Automatic Environment Configuration on page 36: Reorganized the chapter for clarity. 2.10 E2E Latency Measurement on page 49: Added a new section on end to end latency measurement.

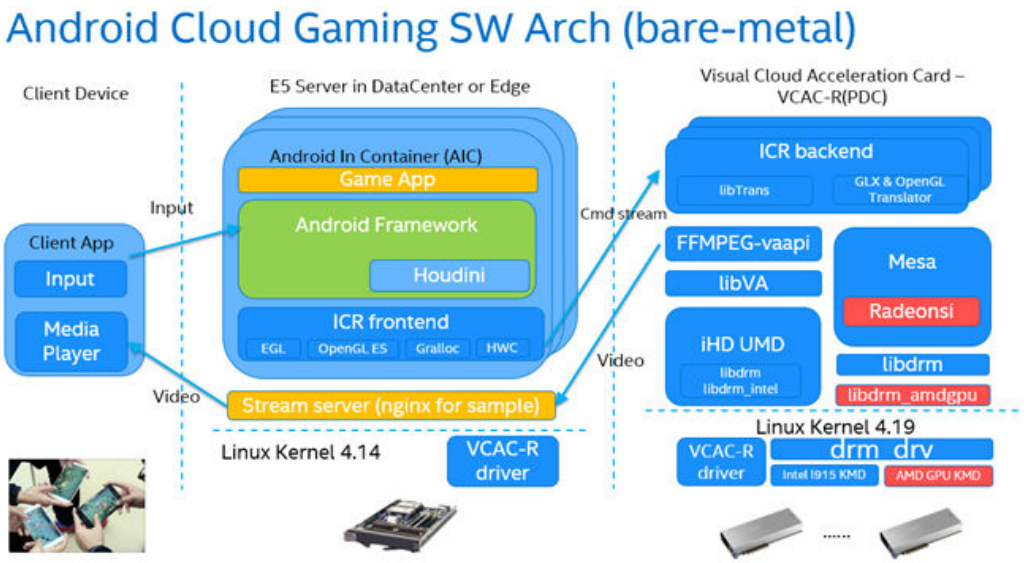
continued...

Date	Revision	Description
		Appendix A Intel® VCA Utility (`vcactl`) on page 50: Added a chapter on Intel VCA Utility (vcactl).
August 2019	1.0	Release 1.0

1.0 Android Solution

1.1 Introduction

The purpose of this section of the document is to provide guidance on how to use the release package for *Cloud Gaming Reference Solution for Android* on platforms with Intel CPU and integrated GPU. Here after, Visual Compute Accelerator card will be referred to as VCA. The software guide can be used for VCAC-R(Visual Cloud Accelerator Card - Rendering), so the terms will be used interchangeably.



Solution contains the following main parts:

1. Android in Container(AiC): Customized Android running in docker container, to ease deployment in data center or edge server. Android game app running in Android container.
2. Intel Cloud Rendering (ICR): The ICR front-end receives rendering requests from the Android game app and Android framework, converts those requests to command streams and sends them to the ICR back-end. The ICR back-end performs real rendering and encoding on the accelerator card GPU, and then sends the video stream.

3. Visual Cloud Accelerator Card - Rendering: Provides high-density GPU rendering capability and high-performance hardware video codec, with whole software stack including user mode middleware, user mode drivers, and kernel mode drivers.
4. Scripts to build and setup the entire reference solution.

1.2 Hardware Configuration

The following table describes recommended hardware configuration:

A Server with 2nd Generation Intel® Xeon® processor is recommended as the host to mount the VCAC-R card through PCIe.

Table 1. Recommended System Hardware Configuration

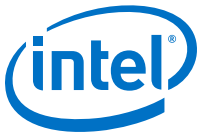
Component Type	Part Name
KNOCK DOWN KIT	Intel® Server System R2312WFTZS 12x3.5in 2x10GbE
CPU	Intel® Xeon™ Platinum 8280M Processor 38.5M Cache, 2.70 GHz ***2**
MEMORY	16GB 2666 Reg ECC 1.2V DDR4 Crucial CT16G4RFD8266 or Micron® MTA18ASF2G72PDZ-2G6D1 ***12**
ATA HARD DRIVE	960GB SSD 2.5in SATA 3.0 6Gb/s Intel® Youngsville SE SSDSC2KB960G701 DC S4500 Series
Add-in Card	Intel® Visual Cloud Accelerator card - Rendering (VCAC-R)*2
NETWORK ADAPTER	On Board
CHASSIS COMPONENT	PCIe 2U Riser Spare Intel® A2UL16RISER2 (2 Slot)
CHASSIS COMPONENT	Passive Airduct Bracket Kit Intel® AWFCOPRODUCTBKT
CHASSIS COMPONENT	Passive Airduct Kit Intel® AWFCOPRODUCTAD
HEATSINK	Included

Table 2. VCAC-R Hardware Configuration

Board Feature Set	Description
CPU	**2***Intel® Core(TM) i7-8709G CPU @ 3.10GHz
Intel GPU	Mesa DRI Intel® HD Graphics 630
AMD GPU	Radeon™ RX Vega M GH Graphics
Memory	8G*2 per CPU; DDR4 2400 ECC SODIMMs, 2 channels per CPU
PCIe* configuration	Gen3, x16, 8 lanes per CPU
BIOS	1* 16 MB SPI flash per CPU
HEATSINK	Included

Hardware specifications datasheet published by Intel is available here: https://www.intel.com/content/dam/www/public/us/en/documents/datasheets/cloud-graphics-vcac-r-accelerator-card-by-celestica-datasheet.pdf

Hardware Customer Support is available from Celestica™ https://hardwaresupport.celestica.com.



NOTE

Customers need to work with their Celestica sales representative to get credentials for the website.

1.3 Build Machine Requirement

The build machine for acgss should meet the following requirements:

- Linux based OS: the scripts are tested on Ubuntu 16.04
- Docker installed: Docker is used in the build process. Refer to https://docs.docker.com/install/linux/docker-ce/ubuntu/ for how to install Docker (Community Edition).
- User privilege: root account (or sudo) is required to give set up permissions for user to use docker.

1.4 Setup Proxy for Docker (if needed)

Internet connection is needed to build Docker container, download source code and dependencies. If you are behind proxy, it needs to be setup for Docker.

1.5 Build acgss Package from Source Code

End users can employ several scripts to download open source code for project components from the Internet. More details are given in the sections below.

1.5.1 README.txt

There is a README.txt which guides user on how to build acgss package from the source code and lists all the commands that user needs to run.

1.5.2 source/rr_code

There is icr (Intel Cloud Rendering) front-end source code provided in source/rr_code, which will be built into AIC

1.5.3 docker/Dockerfile

There is a Dockerfile to provide build environment which contains all environment dependencies. As a first step, a user needs to build and run docker image built by this Dockerfile.

1.5.4 patches/dependencies

There are necessary dependencies/code patches that exist in patches/dependencies.

1.5.5 patches/aosp

There are necessary bsp(board support package) projects which include aosp(Android Open Source Project) patches that exist in patches/aosp directory.

1.5.6 patches/package

There are deployment scripts and documents which get packaged into acgss package.

1.5.7 recipe.json

There is a `recipe.json` file which describes all required open source project components. For each project component, it describes its url, branch, revision. For example:

```
"libva": {
            "url":"https://github.com/intel/libva",
            "branch":"master",
            "revision":"250b3dc8f370bc6d85be767c9722fd98e8b02ebb"
        },
```

1.5.8 download.py

Using the information from recipe.json on page 10, a download script named `download.py`, can download all required source code from Internet. In addition, there are some local patches, which will be applied to the open source project components by the build script as in build.sh on page 10.

1.5.9 build.sh

A build script named `build.sh` builds system in the following sequence:

1. 4.14 kernel
2. 4.19.0 kernel
3. 4.14 modules and 4.19.0 modules
4. Host daemon
5. Handle vcad image build dependencies and create vcad image
6. Build out ICR(Intel Cloud Rendering) back-end
7. Build out AIC image
8. Package

Final compressed file `acgss-$version.tar.gz` will be generated once all steps are done.

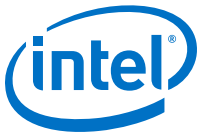
1.6 Install Software with Binary

1.6.1 Preparation

Prepare host system (2nd Generation Intel® Xeon® Scalable Processor) with VCAC-R card installed ready, with Ubuntu16.04 LTS OS installed on host.

The host system BIOS must be configured to enable large Memory-Mapped Input/ Output (MMIO), and allow for large per-device BAR (Base Address Register) allocations. BAR must have 64-bit address enabled.

The minimum requirements for BAR and MMIO are:



- MMIO mapping above 4 GB is enabled
- Minimum MMIO size is 4 GB/CPU (node)

For example, on Intel® Server Board S2600WT based systems, this can be enabled in BIOS setup by configuring the following two options on the PCI Configuration screen:

- Set Memory Mapped Above 4 GB to Enabled
- Set Memory Mapped IO size to 256 GB or higher

Get Android cloud gaming software stack package (acgss) from steps in Build acgss Package from Source Code on page 9. It is a compressed file named as `acgss-$version.tar.gz`. The software stack has support for both baremetal and VM pass through solutions.

Download the file and extract to current path. Then you can find a directory in `./acgss`. Expanded directory structure is shown below:

acgss

```
├── host
│ ├── 1-deploy_vca.sh
│ ├── aic-rr
│ │ ├── aic
│ │ ├── aic-manager.tar.gz
│ │ └── android.tar.gz
│ ├── kernel
│ │ ├── linux-headers-4.14.xxxxxxx-xxx.vca+2019xxxx_amd64.deb
│ │ └── linux-images-4.14.xxxxxxx-xxx.vca+2019xxxx_amd64.deb
│ └── vca
│ ├── daemon-vca-xx-amd64.deb
│ ├── vca_disk24_reference_k4.19.0-1_ubuntu16.04-xx.xx.vcad.gz
│ ├── vcass-modules-xxxxxx_amd64.deb
│ └── 00-net.rules
├── icr
│ └── release_irr_build
│ │ ├── intel-cloud-rendering_ubuntu-16.04_xxxxxxxxxx .tgz
│ │ ├── irr_test_tool
└── script
│ ├── rr_server_file
```

```
│ ├── check-android.sh
└─ └── install_apk.sh
```

1.6.2 Setup SW in Host

Steps for setting up software in the host system are listed in the sections below.

1.6.2.1 Update Host Kernel

1. Install updated kernel package

```
$ cd acgss/host/kernel
$ sudo dpkg -i ./linux-headers-<kernel_ver>.<version>.amd64.deb
$ sudo dpkg -i ./linux-images-<kernel_ver>.<version>.amd64.deb
```

2. Configure the new kernel as default boot option

   Edit `/etc/default/grub`,

   Change the setting

   `GRUB_DEFAULT=0`

   to

   `GRUB_DEFAULT="Advanced options for Ubuntu>Ubuntu, with Linux4.14-xxx-xx.vca+"`

   NOTE

   Kernel name of "Ubuntu, with Linux 4.14-xxx-xx.vca+" is obtained from the `menuentry` output of `$ cat /boot/grub/grub.cfg | grep menuentry`

3. Reload grub

   `$ sudo update-grub`

4. Reboot system

   `$ sudo reboot`

1.6.2.2 Setup nginx/RTMP Server

Download the `nginx` source code, assume it will be downloaded to `acgss/host/nginx`:

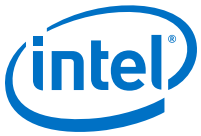
```
$ wget http://nginx.org/download/nginx-1.13.1.tar.gz
```

NOTE

You can find the latest version on the nginx download page.

Get the RTMP module source code from git:

```
$ wget https://github.com/arut/nginx-rtmp-module/archive/master.zip
```



Navigate to `nginx` directory

```
sudo apt-get install openssl libssl-dev libpcre3 libpcre3-dev
$ cd acgss/host/nginx
```

1. Extract both zip files present in the folder and go to `nginx-1.13.1` directory

   ```
   $ tar -zxvf nginx-1.13.1.tar.gz
   $ unzip master.zip
   $ cd nginx-1.13.1
   ```

2. Build nginx

   ```
   $ ./configure --with-http_ssl_module --add-module=../nginx-rtmp-module-master
   $ make
   $ sudo make install
   ```

   nginx is installed, default installation path is `/usr/local/nginx`.

3. Configure nginx to use RTMP

   Open config file which is located by default at `/usr/local/nginx/conf/nginx.conf` and add the following lines at the very end of the file:

   ```
   rtmp
   {
     server{
             listen 1935;
             chunk_size 4096;
             application demo{
              live on;
              record off;
             }
           }
    }
   ```

4. Start nginx server

   `$ sudo /usr/local/nginx/sbin/nginx`

   To verify nginx is running, navigate to the browser and enter http://localhost:8080, you should get the "`Welcome to nginx!`" page.

5. Configure nginx to auto-start at system reboot

   Add `/usr/local/nginx/sbin/nginx` to `/etc/rc.local`, see file example:

   ```
   #!/bin/sh -e
   #
   # rc.local
   #
   # This script is executed at the end of each multiuser runlevel.
   # Make sure that the script will "exit 0" on success or any other
   # value on error.
   #
   # In order to enable or disable this script, just change the execution
   # bits.
   #
   # By default this script does nothing.
   ```

```
/usr/local/nginx/sbin/nginx
dhclient
exit 0
```

1.6.2.3 Setup Docker

Refer to https://docs.docker.com/install/linux/docker-ce/ubuntu/ for detailed steps to install docker.

Optional reference to setup docker: https://docs.docker.com/config/daemon/systemd/

1.6.2.4 Install VCA Software

There is one key deployment script developed to combine the VCA software installation and deployment process. It includes VCA daemon tool installation, loading vcad and configuring auto boot of vca nodes every time the host system restarts.

There are 2 `systemd` services (`loadvcad.service` and `bootvca.service`) provided for managing VCA image load and vca boot status.

```
$ cd acgss/host
$ ./1-deploy_vca.sh
```

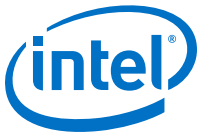
NOTE

It takes about 10-15 minutes for VCA image setup at first time. User may need to reboot if there is a message saying "VCA software install done, please reboot and wait about 10-15 minutes to load vcad."

Login to VCA node with the username and password set during build (take node00 as example):

```
$ ssh root@172.33.1.1

password: vista1
```



NOTE

The EEPROM and BIOS of VCAC-R are also available in the public website. User can find them from the following links:

EEPROM: https://01.org/openvisualcloud/download

BIOS: https://hardwaresupport.celestica.com

Following steps can be referred to flash the EEPROM and BIOS in need, but user is not recommended to upgrade EEPROM and BIOS by themselves unless there are critical bug fixes.

Update BIOS:

1. Power on the host.
2. Unzip the downloaded VCAC-R BIOS update package.
3. Reset the node using the following command:

   ```
   $ sudo vcactl reset
   ```

4. Wait for the reset to complete; the command `sudo vcactl wait-BIOS` or `sudo vcactl status` returns `bios_up` when ready.
5. Run the following command:

   ```
   $ sudo vcactl update-BIOS <BIOS file name>
   ```

6. Wait for the update to complete; the command `sudo vcactl wait-BIOS` or `sudo vcactl status` returns `bios_up` when ready.
7. After BIOS update is complete, the node(s) reboot to come online with the new BIOS version.
8. When the last command finishes successfully, G3 (mechanical off) is necessary. To perform G3, execute the power off command on the host and then disconnect from the power supply for 15 seconds.

Update EEPROM:

1. Unzip the downloaded VCAC-R EEPROM Update package
2. Reset the node using the following command:

   ```
   $ sudo vcactl reset
   ```

3. Wait for the reset to complete; the command `sudo vcactl wait-BIOS` or `sudo vcactl status` returns `bios_up` when ready.
4. Run the following command:

   ```
   $ sudo vcactl update-EEPROM <EEPROM file name>
   ```

5. When the last command finishes successfully, reboot the host. After rebooting, the software installation is complete.

1.6.2.5 Start ICR Service in VCA Node

1. **Copy icr software to vca node**

```
root@vca_node_00:~# scp -r <username>@172.33.1.254:~/acgss/icr/ ~
```

2. **Stop and disable firewall**

```
root@vca_node_00:~# systemctl stop firewalld.service
root@vca_node_00:~# systemctl disable firewalld.service
root@vca_node_00:~# systemctl status firewalld.service
```

3. **Install icr**

   Extract acgss/icr/release_irr_build/intel-cloud-rendering_<irr-version>.tgz and install icr

```
root@vca_node_00:~# tar zxvf ~/icr/release_irr_build/intel-cloud-
rendering_<irr-version>.tgz -C ~
root@vca_node_00:~# cd ~/intel-cloud-rendering_<irr-version>/
root@vca_node_00:~# ./install.sh
```

4. **Install AMD driver in vca node**

   a. Configure proxy and dns to use apt install software.

   b. Install AMD GPU driver package: Download AMD 18.40 GPU driver for ubuntu 16.04(amdgpu-pro-18.40-697810-ubuntu-18.04.tar.xz) from https://www.amd.com/en/support/previous-drivers/graphics/radeon-rx-vega-series/radeon-rx-vega-series/radeon-rx-vega-56

   Click `Ubuntu x86 64-Bit` and then scroll to `Radeon Software for Linux Driver for Ubuntu 18.04.1` and click `Download` button.

```
root@vca_node_00:~# tar xvf amdgpu-pro-18.40-697810-ubuntu-18.04.tar
root@vca_node_00:~#  cd amdgpu-pro-18.40-697810-ubuntu-18.04
root@vca_node_00:~#  sed -i '/LIB32_META_PACKAGE=amdgpu-lib32/d' amdgpu-
install
root@vca_node_00:~#  sed -i '/LIB32_OPENGL_META_PACKAGE=amdgpu-pro-
lib32/d' amdgpu-install
root@vca_node_00:~#  ./amdgpu-install

if user meets amdgpu-lib32 dependencies error, please run
following command:

root@vca_node_00:~# dpkg --add-architecture i386
```

   c. Setup and start xorg service

```
root@vca_node_00:~#  cp ~/irr_test_tool/xorg.conf /etc/X11
root@vca_node_00:~#  cp ~/irr_test_tool/xorg.service /lib/systemd/system/
root@vca_node_00:~#  systemctl start xorg.service
root@vca_node_00:~#  systemctl enable xorg.service
```

   d. Check the OpenGL renderer string is `AMD VEGAM`

```
root@vca_node_00:~#  glxinfo |grep render
OpenGL vendor string: X.Org
OpenGL renderer string: AMD VEGAM (DRM 3.27.0 AMD 18.30 ,
```

```
4.19.0-1.35e5bcd-110.vca+, LLVM 6.0.1)
OpenGL core profile version string: 4.5 (Core Profile) Mesa 18.1.0-
rc4
```

5. **Check streaming connection with following command**

```
root@vca_node_00:~# cd ~/irr_test_tool/
root@vca_node_00:~# ./irr start -n 1 -url rtmp -brc cqp
```

This command will help user to setup `1x ICR` instance with `rtmp` path, and `brc` mode is `CQP`. See `./irr start -h` for detailed usage. See `./irr show -help` to see `help` to check ICR instance status.

There should be no error message reported in `journalctl log`, connection will be established once AiC is started. Make sure nginx server has started on host before initiating ICR instance with rtmp path.

1.7 Usage Scenarios

1.7.1 Multi-thread Remote Render

1.7.1.1 Initiate Streaming Service in VCA Node

1. Check firewall status

```
root@vca_node_00:~# systemctl status firewalld.service
```

Refer to Start ICR Service in VCA Node on page 16 to disable firewall if the status is active.

2. Initiate **20** threads of remote rendering service from `172.33.1.1`. The `rtmp` streaming server is located on host. The default host ip is 172.33.1.254 for VCA node00.

```
root@vca_node_00:~# cd ~./irr_test_tool
root@vca_node_00:~# ./irr start -n 20 -url rtmp -brc cqp
```

There should be no error message reported with "`systemctl status irr.$[i]`" after streaming setup. `$[i] should be 1,2,3…20`.

1.7.1.2 Set Up AIC and Install Game

1. Open another terminal and login to host.
2. Prepare remote render service config file for AIC to install multi-threads of android instance.

   Refer to `acgss/script/rr_server_file` for remote render service configuration file example. Note that the IP and the port must match with the setting to initiate remote render service in Initiate Streaming Service in VCA Node on page 17.

3. Install **20** android instances, specify adb port, RR server. For example:

```
$ cd acgss/host/aic-rr
$ ./aic install -n 20 -e -b 5590 -l server -f acgss/script/rr_server_file
```

4. Start AIC

```
$ ./aic start
```

It will report "`found a guest connection`" and FPS will display on VCA side once android connects to remote render service.

5. Install game

Refer to `acgss/script/install_apk.sh` to install target game `apk`. For example:

```
$ ./install_apk.sh  20 ~/dtpkbd_1499043809924.apk
```

6. Display the streaming

Streaming could be received on any devices within the LAN. Find a laptop or PC and connect to the LAN, install video player (take `vlc` as example), run the cmd "`vlc.exe rtmp://<The host IP>/demo/i.flv`" (`i` is a number from 1 to 20) to display the streaming.

1.7.2 Null-Render Usage

Use either adb or docker Command to Enable Null Rendering Function.

```
$ sudo docker exec -tid android0 sh -c "setprop debug.egl.skipDraw 1"
$ sudo docker exec -tid android0 sh -c "setprop debug.egl.skipSwap 1"
```

OR

```
$ adb shell
> setprop debug.egl.skipDraw 1
> setprop debug.egl.skipSwap 1
```

The display output will stop responding to mouse movement indicating null rendering works. Set both properties to `0` to disable null rendering. Display refresh will be restored.

1.8 Reference Steps to Setup VCA Manually

1.8.1 Install VCA Software

```
$ cd acgss/host/vca
$ sudo apt install ./daemon-vca_<vca_version>-amd64.deb
```

1.8.2 Load VCA Image and Configure VCA

Navigate to the vca directory:

```
$ cd acgss/host/vca/
```

1. Load VCA image

   Extract the persistent image copy for each node. Make sure that each node has its own copy.

   NOTE

   The extracted .vcad file needs 10GB of free disk space.

```
$ mkdir -p acgss/host/vca/node00
$ tar -xvf vca_reference_<vca-image-version>.tar.bz2 -C acgss/host/vca/node00
```

   Add the extracted vca_disk image as the primary BlockIO device.

```
$ sudo vcactl blockio open 0 0 vcablk0 RW acgss/host/vca/node00/
vca_reference_<vca-image-version>.vcad
```

   NOTE

   if you want to boot more than 1 vca node (eg: to boot 2 nodes), following steps are needed:

```
$ mkdir acgss/host/vca/node01

$ cp acgss/host/vca/node00/vca_reference_<vca-image-version>.vcad acgss/
host/vca/node01

$ sudo vcactl blockio open 0 0 vcablk0 RW acgss/host/vca/node01/
vca_reference_<vca-image-version>.vcad
```

2. Configure IP of VCA node

Refer to the script `acgss/script/enable_nat.sh` to enable NAT config.

1.8.3 Boot VCA Nodes

1.8.3.1 Check VCA Status

Make sure all pre-boot nodes are in the correct status.

```
$ sudo vcactl status
```

It should return the following status:

```
Card: 0 Cpu: 0 STATE: bios_up
Card: 0 Cpu: 1 STATE: bios_up
```

If not, run the following command and wait for all nodes to be ready

```
$ sudo vcactl wait-BIOS
```

1.8.3.2 Boot VCA Node and Login

Add `udev` rules for the VCA virtual interfaces (eth0/eth1/eth2) to avoid to be managed by NetworkManager. This is critical since the service may block communication between the host and Intel® VCA card.

Add the following lines to `/etc/udev/rules.d/00-net.rules`

```
ACTION=="add", SUBSYSTEM=="net",KERNEL=="eth0", ENV{NM_UNMANAGED}="1"
ACTION=="add", SUBSYSTEM=="net",KERNEL=="eth1", ENV{NM_UNMANAGED}="1"
```

Use `/etc/init.d/udev restart` command to let it take effect if you don't want to reboot the server.

Boot VCA node:

```
 $ sudo vcactl boot 0 0 vcablk0
```

Check the status for return value `net_device_ready`:

```
$ sudo vcactl status
Card: 0 Cpu: 0 STATE: net_device_ready
Card: 0 Cpu: 1 STATE: bios_up
```

2.0 Windows Based Solution

2.1 Introduction

The purpose of this chapter is to provide guidance on how to use *Cloud Gaming for Windows* reference solution on Windows 10 OS. It describes how to configure environment for building this solution and how to run it.

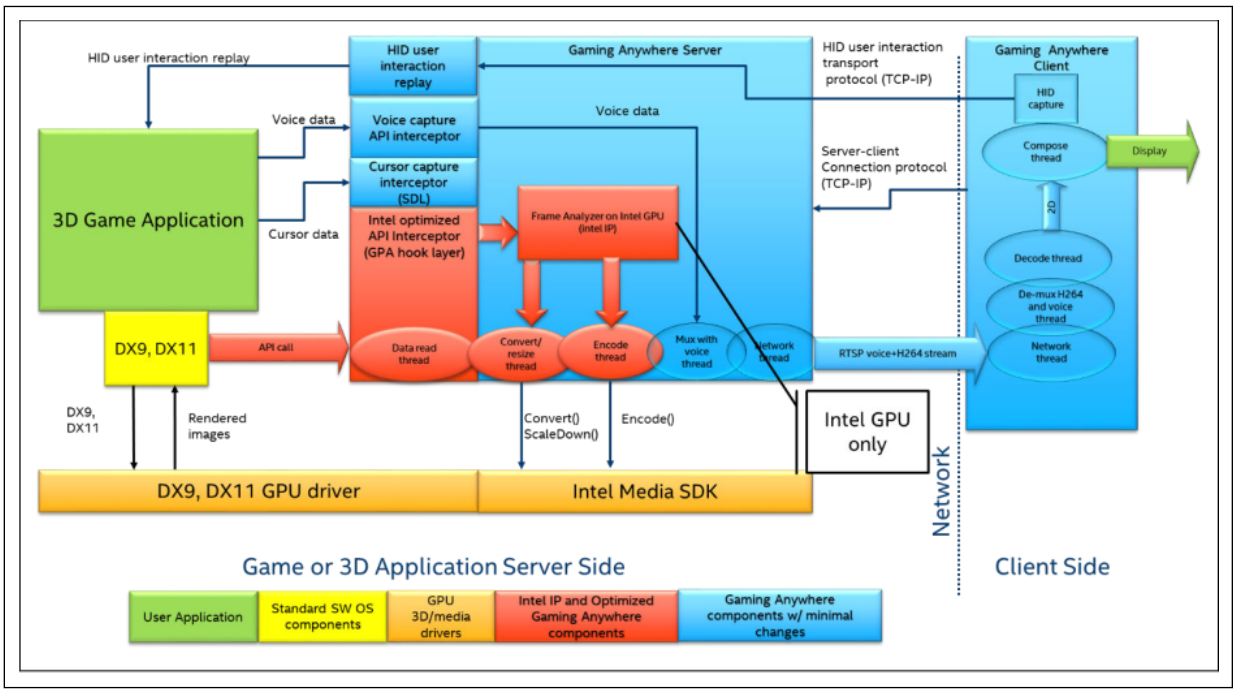
This product is based on open source GamingAnywhere (GA) solution, which is available on public github (http://gaminganywhere.org/). It provides end to end solution for cloud gaming, based on client–server architecture. Base solution has been modified and improved to benefit from the advantages provided by Intel hardware architecture. Modified code is provided as a reference solution for future implementations.

2.1.1 Architecture

Architecture of the *Cloud Gaming For Windows* is based on client-server model of communication. Server is responsible for receiving request from client, launching and getting content from game, encoding the content and sending it to the client. Client is responsible for connecting to the server, receiving the content from server, decoding content and displaying it on the client screen.

The following figure describes the architecture:

Figure 1. Cloud Gaming For Windows Reference Solution Architecture.



2.1.2 VCAC-R Contents of the Release Package

The contents of release package for VCAC-R based host SW package(kernel modules and user space app) are listed in the table below. Intel® drives these ingredients through continuous development and validation to ensure that software updates will perform correctly when integrated into a deployed system.

NOTE

The version of the release software might change. Please refer to the Release Notes Doc. No. 611946 accompanied with the latest Software Releases.

The latest release for VCAC-R Host SW can be found at https://github.com/OpenVisualCloud/VCAC-SW/tree/VCAC-R_R3. User can download it or clone the package from the github link.

Table 3. VCAC-R Contents of Release Package

Ingredient	Folders and Contents
VCAC-R host SW package	apps
	buildscripts
	eeprom
	modules
	patches
	windows

continued...

Ingredient	Folders and Contents
	README.md
	buildscripts/download_dependencies.sh
	buildscripts/master_build.sh
	buildscripts/windows/windows_part (*essential for windows image generation*)
	eeprom/VCAC-R/eeprom/VCAC-R (*contains DMA-disabled eeprom and normal eeprom*)
	windows/vcagent (*VCA Agent*)

NOTE

The EEPROM and BIOS for VCAC-R are also available in the public website. User can find them from the following links:

EEPROM:https://01.org/openvisualcloud/download

BIOS: https://hardwaresupport.celestica.com

The VCAC-R host SW package contains the following:

- script `./buildscripts/master_build.sh` to build the VCAC-R host support package including the kernel patch, VCAC-R PCIe driver and VCAC-R utility to be installed on Host server with VCAC-R card.
- script `./buildscripts/download_dependencies.sh` to download all source dependencies required for build process.

The VCAC-R Card SW package comprises of *Cloud Gaming For Windows* Application (*CloudGamingForWindows.exe*). User can follow the links given below to download the application:

- Link to public repository for Cloud Gaming for Windows: https://github.com/OpenVisualCloud/Cloud-Gaming-Windows-Sample
- The latest release is in "releases" tab: https://github.com/OpenVisualCloud/Cloud-Gaming-Windows-Sample/releases
- Link to zip containing "CloudGamingForWindows.exe": https://github.com/OpenVisualCloud/Cloud-Gaming-Windows-Sample/releases/download/v2.0/Cloud-Gaming-Windows-Sample-v2.0.zip

2.1.3 Hardware Configuration

The following table describes recommended hardware configuration:

A Server with 2nd Generation Intel® Xeon® processor is recommended as the host to mount the VCAC-R card through PCIe.

Table 4. Recommended System Hardware Configuration

Component Type	Part Name
KNOCK DOWN KIT	Intel® Server System R2312WFTZS 12x3.5in 2x10GbE
CPU	Intel® Xeon™ Platinum 8280M Processor 38.5M Cache, 2.70 GHz ***2**

continued...

Component Type	Part Name
MEMORY	16GB 2666 Reg ECC 1.2V DDR4 Crucial CT16G4RFD8266 or Micron® MTA18ASF2G72PDZ-2G6D1 ***12**
ATA HARD DRIVE	960GB SSD 2.5in SATA 3.0 6Gb/s Intel® Youngsville SE SSDSC2KB960G701 DC S4500 Series
Add-in Card	Intel® Visual Cloud Accelerator card - Rendering (VCAC-R)*2
NETWORK ADAPTER	On Board
CHASSIS COMPONENT	PCIe 2U Riser Spare Intel® A2UL16RISER2 (2 Slot)
CHASSIS COMPONENT	Passive Airduct Bracket Kit Intel® AWFCOPRODUCTBKT
CHASSIS COMPONENT	Passive Airduct Kit Intel® AWFCOPRODUCTAD
HEATSINK	Included

Table 5. VCAC-R Hardware Configuration

Board Feature Set	Description
CPU	**2***Intel® Core(TM) i7-8709G CPU @ 3.10GHz
Intel GPU	Mesa DRI Intel® HD Graphics 630
AMD GPU	Radeon™ RX Vega M GH Graphics
Memory	8G*2 per CPU; DDR4 2400 ECC SODIMMs, 2 channels per CPU
PCIe* configuration	Gen3, x16, 8 lanes per CPU
BIOS	1* 16 MB SPI flash per CPU
HEATSINK	Included

Hardware specifications datasheet published by Intel is available here: https://www.intel.com/content/dam/www/public/us/en/documents/datasheets/cloud-graphics-vcac-r-accelerator-card-by-celestica-datasheet.pdf

Hardware Customer Support is available from Celestica™ https://hardwaresupport.celestica.com.

NOTE

Customers need to work with their Celestica sales representative to get credentials for the website.

2.1.4 Software Configuration

Software configuration requirements for the hardware listed above are in the table below. Software specific for *Cloud Gaming For Windows* is described in the next section as Additional Specific Software Requirements on page 34

Table 6. Recommended Software Configuration

Component Type	Part Name
Host OS	CentOS 7.6 3.10.0-957
Card software on host	VCA SW 2.7
Card node OS	Windows 10 1803
Card software	VCA SW 2.7

2.2 Installation

2.2.1 Preparation

The host system BIOS must be configured to enable large Memory-Mapped Input/Output (MMIO), and allow for large per-device BAR (Base Address Register) allocations. BAR must have 64-bit address enabled.

The minimum requirements for BAR and MMIO are:

- MMIO mapping above 4 GB is enabled
- Minimum MMIO size is 4 GB/CPU (node)

For example, on Intel® Server Board S2600WT based systems, this can be enabled in BIOS setup by configuring the following two options on the PCI Configuration screen:

- Set Memory Mapped Above 4 GB to Enabled
- Set Memory Mapped IO size to 256 GB or higher

2.2.2 Build Host Kernel, VCAC-R Driver on Host and VCAC-R System Image

Build host kernel, VCAC-R driver on host and VCAC-R node OS system image using scripts included in the release package. The steps are described in subsequent sections.

2.2.3 Build Machine Requirement

The build machine should meet the following requirements:

- Linux based OS: the scripts are tested on Ubuntu 18.04
- Docker installed: Docker is used in the build process. Refer to https://docs.docker.com/install/ for how to install Docker. Minimal required version is Docker 17.05. Further information can be found at https://docs.docker.com/config/daemon/systemd/ and https://docs.docker.com/install/linux/linux-postinstall/.
- User privilege: root account (or sudo) is required to give set up permissions for user to use docker.

2.2.4 Setup Proxy for Docker (if needed)

Internet connection is needed to build Docker container, download source code and dependencies. If you are behind proxy, it needs to be setup for Docker.

`./buildscripts/master_build.sh` would take into account `http_proxy` environment variable and will pass it to docker. Additional information for setting http proxy can be found at https://docs.docker.com/config/daemon/systemd/.

2.2.5 Build Host Kernel and VCAC-R Driver

VCAC-R latest release can be cloned/downloaded from the link https://github.com/OpenVisualCloud/VCAC-SW/tree/VCAC-R_R3. The contents are mentioned in VCAC-R Contents of the Release Package on page 22 as VCAC-R host SW package. The scripts therein will be used to generate binaries.

`README.md` will be handy for regular Unix users. `master_build.sh` and `download_dependencies.sh` are in "`buildscripts`" directory.

NOTE

All scripts mentioned in this section print usage information when invoked without arguments.

NOTE

Single dot (.) in beginning of paths (i.e.: ./buildscripts/master_build.sh) refers to top-level directory of VCAC-R host SW package.

Download all required source dependencies:

```
$ ./buildscripts/download_dependencies.sh --downloads-dir
DIRECTORY_TO_STORE_DOWNLOADS
```

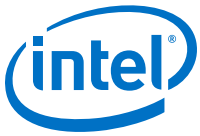
Usual `http_proxy` environment variables will be used by `wget` utility.

Build host kernel and VCAC-R driver modules and user space utility with the scripts included in the release package. (Run as a user with privileges to use docker). (Switches begin with two minus signs, command should be with three mandatory switches):

```
$./buildscripts/master_build.sh --vca-src-dir
PATH_TO_EXTRACTED_VCA_SW_SOURCE_RELEASE_PACKAGE \
        --downloads-dir DIRECTORY_WITH_DOWNLOADS_FROM_PREV_STEP \
        --out-dir PLACE_TO_PUT_RESULTING_VCA_SW_BINARIES
```

After compilation is completed, the output will be available in requested location above (`PLACE_TO_PUT_RESULTING_VCA_SW_BINARIES`). Most important parts of its content are following (they will be placed inside separate directories for CentOS and Ubuntu):

- Host kernel packages:
  - kernel-3.10.0-957.el7.centos.2.7.363.VCA.x86_64.rpm
  - linux-image-4.19.0-1.2.7.363.vca_1.0_amd64.deb
- VCAC-R driver module:
  - vcass-modules-3.10.0-957.el7.centos.2.7.363.VCA.x86_64-2.7.363-0.x86_64.rpm



  - vcass-modules-4.19.0-1.2.7.363.vca_2.7.363-1_amd64.deb
- Utility Application:
  - daemon-vca-2.7.363-x86_64.rpm
  - daemon-vca-2.7.363-x86_64.deb
- These files are also generated:
  - kernel-3.10.0-957.el7.centos.2.7.363.VCA.src.rpm
  - kernel-devel-3.10.0-957.el7.centos.2.7.363.VCA.x86_64.rpm
  - linux-headers-4.19.0-1.2.7.363.vca_1.0_amd64.deb

2.2.6 Build Windows System Image for VCAC-R node

2.2.6.1 Windows Tools

The following hardware and software resources are required to create Windows images:

- Windows drivers and VCAgent service can be downloaded from this link: https://downloadcenter.intel.com/download/28451/Intel-Visual-Compute-Accelerator-VCA1283LVV-VCA1585LMV-Windows-Image-Creation-Package and https://openvisualcloud.github.io/VCAC-SW/windows/vcagent/(latest version of VCAgent service is available here).
- Windows installation ISO (for example, `en_windows_10_business_edition_version_1803_updated_jul_2018_x64_dvd_12612769.iso`).
- Windows Server 2012/2016 system to perform build operation. System should be at least as up-to-date as the installation ISOs used to create the images and all Microsoft* updates that are available should be installed.
- Oracle VM VirtualBox software (available at https://www.virtualbox.org/wiki/Downloads). These need to be installed prior to the build process.
- Hyper V Server Role installed. This could be selected during Windows Server setup or later using standard techniques via Server Manager -> Manage -> Add Roles and Features Wizard.
- Windows Assessment and Deployment Kit (Windows ADK) version 10 (available at https://developer.microsoft.com/en-us/windows/hardware/windows-assessment-deployment-kit).
- 45 GB of free space. (This is the default amount; the actual amount depends on container size.)
- OpenSSH(if required for intended use). It can be downloaded from https://www.openssh.com/.
- Powershell script for image creation (`vca_create_windows_image.ps1` from `buildscripts/windows/windows_part` in VCAC-R host SW package(VCAC-R Contents of the Release Package on page 22) (which also contains all other source files for Windows image generation, particularly the folder `answer_files`).

2.2.6.2 Creating Windows* Baremetal Images

Windows baremetal images are dedicated for BlockIO boot mode (default, persistent approach) or via KVM hypervisor. `boot_part_type` parameter set should be set to `GUID Partition Table (GPT)` and can be used for BlockIO or KVM, Master Boot Record (MBR) for KVM or Xen.

2.2.6.3 Image Creation Script Parameters

- -iso – Path to Windows installation iso file.
- -output_dir – Directory which will contain compressed resulting img for VCA (if not specified, default directory name is `out`).
- -mount_dir – Directory to which `wim` file from windows installation iso will be mounted during image creation (if not specified, default directory name is `mount_dir`).
- -tmp_dir – Directory used for temporary files (if not specified, default directory name is `tmp`).
- -driver_dir – Directory containing VCA drivers (available with build release). This directory needs to contain the `Win8.1Release-x64` folder, as the folder name is hardcoded.
- -vcagent_dir – Directory containing VCAgent (available with build release). This directory should contain `VCAgent.*` files.
- -answer_file – Path to Windows `answerfile` (same as `unattend` file).
- -dism_path – Path to `dism` file (from Windows ADK).
- -vca_image_version – Added to resulting file name (default is `0.0.0`).
- -gfx_drv_dir – Optional. Path to directory containing Intel graphics driver.
- -win_edition – Windows edition to be incorporated into img file (for example ServerStandard, Enterprise or Windows 10 Pro).
- -netkvm_drv_dir – Optional. If image is going to be used on KVM, it specifies path to NetKVM driver directory (optional).
- -xen_driver_dir – Optional. If image is going to be used on Xen, it specifies path to Xen driver directory (containing XenBus, XenNet, and Xenvif) (optional).
- -openssh_dir – Optional. Path to OpenSSH installation files directory, if it is to be installed (optional). Hardcoded in `AutoUnattend.xml` file for `setupssh-7.4p1-1.exe`.
- -virtualbox – Path to `VBoxManage.exe` file.
- -vhd_size – Resulting image size in GB. Recommended minimum size is 15 GB.
- -boot_part_type – Type of created image. Only GPT (for BlockIO) is supported.
- -zip_img – Name of zip file containing resulting Windows image file (optional). Default is `vca_windows_baremetal.zip`.

2.2.6.4 Creation of BlockIO/KVM Windows Images

Once all required software and files are present, image creation is quite straightforward. The script has to be executed with parameters specified above in Image Creation Script Parameters on page 28. It results in a zipped .img file created in the output directory. This file can be booted (depending on parameters provided) on BlockIO/KVM/Xen.

Directories tmp_dir, out_dir, and mount_dir can be specified on different drives (SSD or M.2) to speed up image building process. If not defined, they are created in the same directory in which the PowerShell script is executed.

An example use of the image building script is below:

```
.\vca_create_windows_image.ps1 -iso .\en_windows_server_2016_x64_dvd_9718492.iso -
driver_dir .\VcaKmdWin\Win8.1Release-x64 -vcagent_dir .\VCAgent -answer_file .
\answer_files\WS2016\AutoUnattend.xml -gfx_drv_dir .
\GFX_driver_20.19.15.4549_BDW_SKL -dism_path 'C:\Program Files (x86)\Windows Kits
\10\Assessment and Deployment Kit\Deployment Tools\x86\DISM\dism.exe' -
win_edition ServerStandard -virtualbox 'C:\Program Files\Oracle\VirtualBox
\VBoxManage.exe' -vhd_size 11GB -boot_part_type GPT  -xen_driver_dir .\Xen  -
openssh_dir .\OpenSSH -netkvm_drv_dir .\NetKVM\2k16 -tmp_dir d:\tmp -mount_dir d:
\mount_dir  -zip_img vca_windows_10_1803_baremetal_GPT.zip
```

2.2.6.5 Windows Image Resizing

If required, images or containers can be resized simply by running following command:

```
dd if=/dev/zero bs=1M count=4000 >> ./binary.img
```

The count parameter depends on the amount of space to be added. After resizing the file, the image should be booted and the disk resized normally in Windows disk manager using unallocated space.

2.2.7 Using VCAC-R System Image and Installation Package

This method requires that VCAC-R card is plugged in to the PCIe slot of the Intel® Xeon® Scalable Processor.

2.2.7.1 Setting up Intel® Xeon® Scalable Processor Server Host

- Install CentOS7.6 in host Intel® Xeon® Scalable Processor server

  (Optional with some types of Scalable Processor servers: If you see an error like "CPU #* stuck for ** seconds", try the following:

  Boot from USB drive, when you boot into install grub page, press Tab button, add "*nomodeset*" in grub command line)

- Disable SSC in BIOS (*the menu path might be different in different BIOS*):

  ***Advanced -> integrated IO Configuration** I **-> Pcie P11 SSC** (set to Disable)*

- Set proxy in host (Optional, depends on local network environment):

```
# vim /etc/yum.conf
export https_proxy="local network https proxy"
export http_proxy="local network http proxy"
```

```
export ftp_proxy="local network ftp proxy"

# vim /etc/apt.conf

Acquire::http::proxy "local network http proxy";
Acquire::https::proxy "local network https proxy";
# yum update
```

- Before installing the VCAC-R software package on host, check PCIe device availability through the following:

```
# lspci | grep PLX
af:00.0 PCI bridge: PLX Technology, Inc. Device 8733 (rev ca)
b0:08.0 PCI bridge: PLX Technology, Inc. Device 8733 (rev ca)
b0:09.0 PCI bridge: PLX Technology, Inc. Device 8733 (rev ca)

# lspci | grep 295
af:00.2 System peripheral: Intel Corporation Device 2952 (rev ca)
af:00.4 System peripheral: Intel Corporation Device 2952 (rev ca)
b1:00.0 Bridge: Intel Corporation Device 2954 (rev ca)
b2:00.0 Bridge: Intel Corporation Device 2955 (rev ca)
```

2.2.7.2 VCAC-R Software Installation on Host

Follow the steps below to install the VCAC-R software on host:

1. Copy the kernel and VCAC-R card driver package compiled in Build Host Kernel and VCAC-R Driver on page 26 to a temporary folder on the host server.
2. Install the kernel packages:

```
#sudo yum -y localinstall kernel-3.10.0-957.el7.centos.2.7.363.VCA.x86_64.rpm
#sudo yum -y localinstall kernel-devel-3.10.0-957.el7.centos.
2.7.363.VCA.x86_64.rpm
```

3. Configure the new kernel to be the default at boot.

NOTE

This step is very important to boot the operating system with proper kernel.

```
#sudo grub2-set-default 0
```

4. (Optional) If updating from a previous VCAC-R version, remove the older RPMs with the following command:

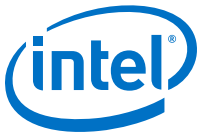
```
#rpm -qa | grep -e daemon-vca -e vcass-modules | xargs yum -y erase
```

5. Install the VCAC-R driver and utility.

```
#sudo yum -y localinstall vcass-modules-3.10.0-957.el7.centos.
2.7.363.VCA.x86_64-2.7.363-0.x86_64.rpm
#sudo yum -y localinstall daemon-vca-2.7.363-x86_64.rpm
```

6. Reboot the system to enable the new kernel.

```
#sudo reboot
```



7. After reboot, confirm that the expected kernel on the host is being used.

```
#uname -r 3.10.0-957.el7.centos.2.7.363.VCA.x86_64
```

2.2.7.3 Accessing VCAC-R Card via Host Using DHCP Setup

Following steps can be used to access the VCAC-R card via host using DHCP setup:

1. Stop Network Manager

```
systemctl stop NetworkManager
service NetworkManager stop
```

2. Create a bridge configuration file (e.g. `/etc/sysconfig/network-scripts/ifcfg-virbr0-vca`) containing the following lines:

```
DEVICE=virbr0-vca
TYPE=Bridge
BOOTPROTO=dhcp
ONBOOT=yes
NM_CONTROLLED=no
```

3. Modify the main interface file (e.g. `/etc/sysconfig/network-scripts/ifcfg-enp3s0f0`) to contain the following lines. The main interface is the interface physically connected to the network.

NOTE

`enp3s0f0` is the main Network Interface,, this name could change in every system.

```
DEVICE=enp3s0f0
TYPE=Ethernet
BOOTPROTO=dhcp
ONBOOT=yes
NM_CONTROLLED=no
BRIDGE=virbr0-vca
UUID=UUID_OF_DEVICE
```

4. Restart the network.

```
systemctl restart network
```

5. Configure Intel® VCAC-R nodes to use the new network bridge.

```
vcactl config bridge-interface virbr0-vca
```

6. Increase the MTU size of the main network interface used .

```
ifconfig enp3s0f0 mtu 9000 (Note :replace enp3s0f0 with main network
interface name)
```

7. For DHCP configuration, use the following command:

```
vcactl config ip dhcp
```

2.2.7.4 VCAC-R Card Boot up with vcad Image

Boot up the VCAC-R card with the Windows Image created before as in chapter Build Windows System Image for VCAC-R node on page 27 and in section Creation of BlockIO/KVM Windows Images on page 29:

- Boot VCAC-R card

```
#vcactl blockio open 0 0 vcablk0 RW $PATH/
vca_windows_10_1803_baremetal_GPT_2.6.265.img
#vcactl boot 0 0 vcablk0
```

- Check VCAC-R status:

```
#vcactl status
```

If status is :

```
Card: 0 Cpu: 0 STATE:link_down,
```

Then try the steps below:

```
#vcactl pwrbtn-long
#vcactl pwrbtn-short
```

Check the status again via:

```
#vcactl status
```

It should be in "bios_up".

```
Card: 0 Cpu: 0 STATE:bios_up
```

```
#vcactl reset 0 0 --force
#vcactl boot 0 0 vcablk0
#vcactl status
Card: 0 Cpu: 0  STATE: net_device_ready
```

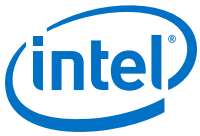
- If the device status is ready, you can login into the VCAC-R card using its IP address.

  Check IP address of the VCAC-R. Following is an example of one host connected with two VCAC-R cards:

```
#vcactl network ip  0  0:
172.33.1.1
Card 1 Cpu 0:
172.33.2.1
```

- Login to VCAC-R Card 0 through RDP protocol/viewer.

```
Username : Administrator
Password: vista12#
```



2.2.8 PXE Feature Description

PXE (Pre eXecution Environment) boot is a mechanism allowing to start an operating system from network, without any physical media attached to the computer. It is not unique to the VCA card - the protocols used by the feature are the same as the ones powering the process on a desktop or server PC.

2.2.8.1 Usage on the Card

The PXE boot feature allows booting the card via a virtual network card that appears on the host, separate from the one that is used by the running OS.

The workflow is as follows:

1. Prepare the PXE boot setup - network, images, servers, etc.
2. Enable the PXE virtual network card.
3. Configure the virtual network card using Linux tools (usually bridging it to another network).
4. Boot the system using PXE.

2.2.8.2 Network Preparation

The PXE boot feature uses DHCP extensively - the server hands out not only addresses, but also location of the bootloader. This makes use of static addressing impossible - if each node is to have a predetermined address, the DHCP server should be configured to hand out static IP addresses based on the client's MAC address.

Preparing the configuration depends on the server, as well as target node operating systems, and is not discussed in this manual - the setup does not differ from preparing for PXE booting an ordinary machine.

2.2.8.3 Limitations and Precautions

Since there is no way to interact with the software running on the card before operating system boots, the boot process has to be non-interactive.

The recommended way to boot Windows is to:

1. serve the image via iSCSI
2. use iPXE open source software to boot over iSCSI – the supported version is the **SNP only version**

The network configuration for the booted OS (usually) has to match the configuration used in PXE boot.

2.2.8.4 vcactl Commands Reference

2.2.8.4.1 Enable the PXE Network Card

```
# vcactl pxe enable
```

Executing this command creates PXE boot network cards, one for each node, named `vcapxe<card><node>`. Note that interfaces are not configured and down by default.

If activation for only one node is desired, it can be done by specifying it on the command line, for example, to enable PXE for card 0, node 2:

```
# vcactl pxe enable 0 2
```

2.2.8.4.2 Disable the PXE Network Card

```
# vcactl pxe disable
```

This command is used to remove the PXE boot network cards. Note that if PXE boot was initiated and it has not finished, either by a success or choosing another boot method, it won't be possible to withdraw the interface, as it will be in use.

2.2.8.4.3 List the PXE Network Card Status

```
# vcactl pxe status
vcapxe00        Active
vcapxe01        Inactive
vcapxe02        Active
```

Lists the status of each network card:

- `Inactive` means that the PXE boot is completely disabled. The dedicated interface is not visible, and the card can’t be booted using PXE.
- `Active` means that the interface is visible and can be configured, or disabled if not needed. Appropriate command will trigger the PXE boot process.
- `In use` means that the node is right now using the interface in its boot process.

2.2.8.4.4 Boot Using PXE

```
# vcactl boot vcapxe
```

Boots the card using previously enabled and configured interface (see Enable the PXE Network Card on page 33). If only one node is to be booted, it can be done, for example to boot card 0, node 2:

```
# vcactl boot 0 2 vcapxe
```

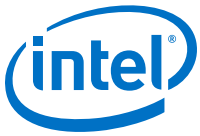
2.3 Additional Specific Software Requirements

Cloud Gaming For Windows solution needs proper graphics drivers to work properly. The two graphics adapters are from Intel and AMD. Proper version of these graphics drivers are as follows:

1. Intel® HD Graphics 630 driver (proper version 25.20.100.6373) - https://downloadmirror.intel.com/28289/a08/win64_25.20.100.6373.exe
2. Radeon™ RX Vega M GH Graphics driver - https://downloadcenter.intel.com/download/28432/Radeon-RX-Vega-M-Graphics?product=136865. Refer to the link Setting up Radeon™ GPU for Gaming on page 67 for setting appropriate GPU for gaming application.

This release of the *Cloud Gaming For Windows* is dedicated for Windows 10 OS. Resources mentioned below are required to build and run it.

Required to build and run:



- Microsoft Visual Studio 2017 (Windows SDK verison: 10.0.17763.0)
- Intel® Media SDK 2018 R2 - https://software.intel.com/en-us/media-sdk/choose-download/client
- Vulkan® SDK version 1.1.114.0 - https://vulkan.lunarg.com/sdk/home#sdk/downloadConfirm/1.1.114.0/windows/VulkanSDK-1.1.114.0-Installer.exe
- Chrome browser and additional proxy settings if needed.
- FFmpeg binaries(supported version 4.1.4):
  - Shared - https://ffmpeg.zeranoe.com/builds/win64/shared/ffmpeg-4.1.4-win64-shared.zip
  - Dev - https://ffmpeg.zeranoe.com/builds/win64/dev/ffmpeg-4.1.4-win64-dev.zip
- EasyHook binaries (supported version 2.7) - https://github.com/EasyHook/EasyHook/releases/tag/v2.7.6789.0

Required to run only:

- Microsoft Visual C++ Redistributable for Visual Studio 2010
  - x86 - https://www.microsoft.com/en-us/download/confirmation.aspx?id=5555
  - x64 - https://download.microsoft.com/download/3/2/2/3224B87F-CFA0-4E70-BDA3-3DE650EFEBA5/vcredist_x64.exe
- Microsoft Visual C++ Redistributable for Visual Studio 2017
  - x86 - https://aka.ms/vs/16/release/vc_redist.x86.exe
  - x64 - https://aka.ms/vs/16/release/vc_redist.x64.exe
- Intel® Media SDK 2018 R2 - https://software.intel.com/en-us/media-sdk/choose-download/client
- Vulkan® SDK version 1.1.114.0 - https://vulkan.lunarg.com/sdk/home#sdk/downloadConfirm/1.1.114.0/windows/VulkanSDK-1.1.114.0-Installer.exe
- Chrome Browser.
- FFmpeg binaries(supported version 4.1.3):
  - Shared - https://ffmpeg.zeranoe.com/builds/win64/shared/ffmpeg-4.1.3-win64-shared.zip
  - Dev - https://ffmpeg.zeranoe.com/builds/win64/dev/ffmpeg-4.1.3-win64-dev.zip
- EasyHook binaries (supported version 2.7) - https://github.com/EasyHook/EasyHook/releases/tag/v2.7.6789.0

2.4 List of Modules

Cloud Gaming For Windows solution contains following projects:

- **asource-system** – audio initialization and configuration
- **CloudGamingForWindows** – bundle of installers
- **CloudGamingForWindowsInstaller** – main installer of components
- **ctr-sdl** – sdl events handling

- **download-extra-elements** – download 3rd party component as FFmpeg and EasyHook
- **encoder-audio** – audio encoding handling
- **encoder-mxf** – video encoder handling
- **filter-rgb2yuv** – screen image processing
- **ga-client** – client related components
- **ga-hook** - additional GFX API calls hooking mechanism implementation
- **ga-server-event-driven** – run media server and start injecting into game process
- **ga-server-manager** – Main application, run server and wait for response from client
- **gpa-hook** – main GFX API calls hooking mechanism implementation
- **libga** – processing components, configurations, loading modules
- **live555-adapter** – make dll from static libraries of live555 streaming media
- **server-live555** – live555 media server related components

2.5 Automatic Environment Configuration

Environment configuration for *Cloud Gaming For Windows* can be done via two methods. First(preferred) method is to automatically configure the environment using an installer utility (`CloudGamingForWindows.exe` from VCAC-R Contents of the Release Package on page 22). Installer requires internet access to work properly. Second method is to manually configure the environment as described in Manual Environment Configuration on page 45.

Step by step instructions on how to set up environment with installer are described below:

NOTE

Chrome Browser should be installed on the machine as a pre-requisite. The executable file needs Internet Access via chrome.

1. Agree to License Terms and Conditions

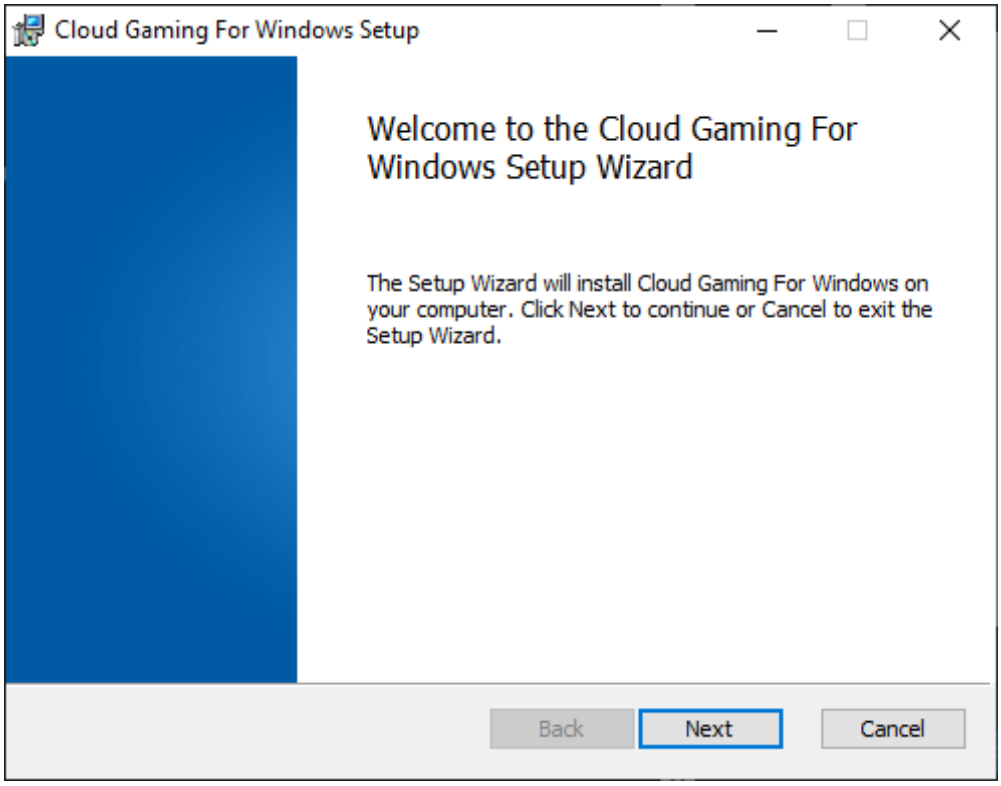
The first step is to agree to the license terms and conditions which are required to proceed with the installation. Agreeing to install external (or 3rd Party) components is optional but these files are necessary for proper working of the environment. After checking all agreements, click “Install”.

2. Processing: Installation of components from the Web

"Processing" is the stage where all necessary components are getting installed from the Web. It may take some time. Following screen shows this step:

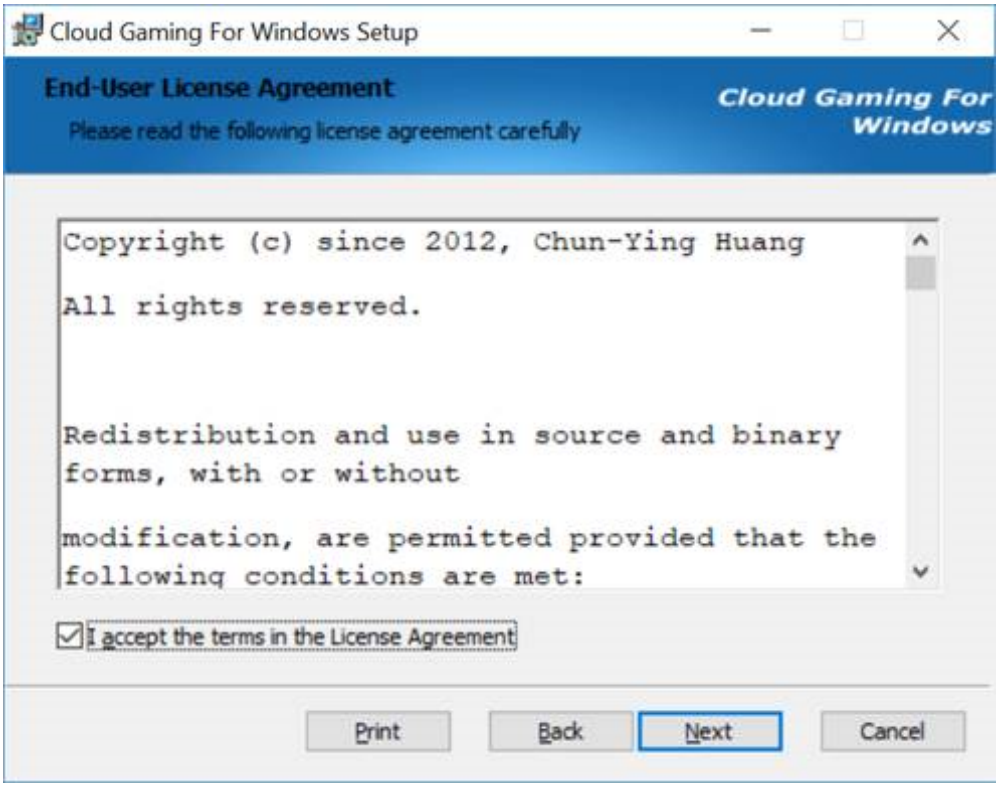
3. Main Component Installer

The initial window of the main component installer for *Cloud Gaming for Windows* is shown in the following image. Read the information and click “Next”.



4. License Agreement of Cloud Gaming for Windows Setup

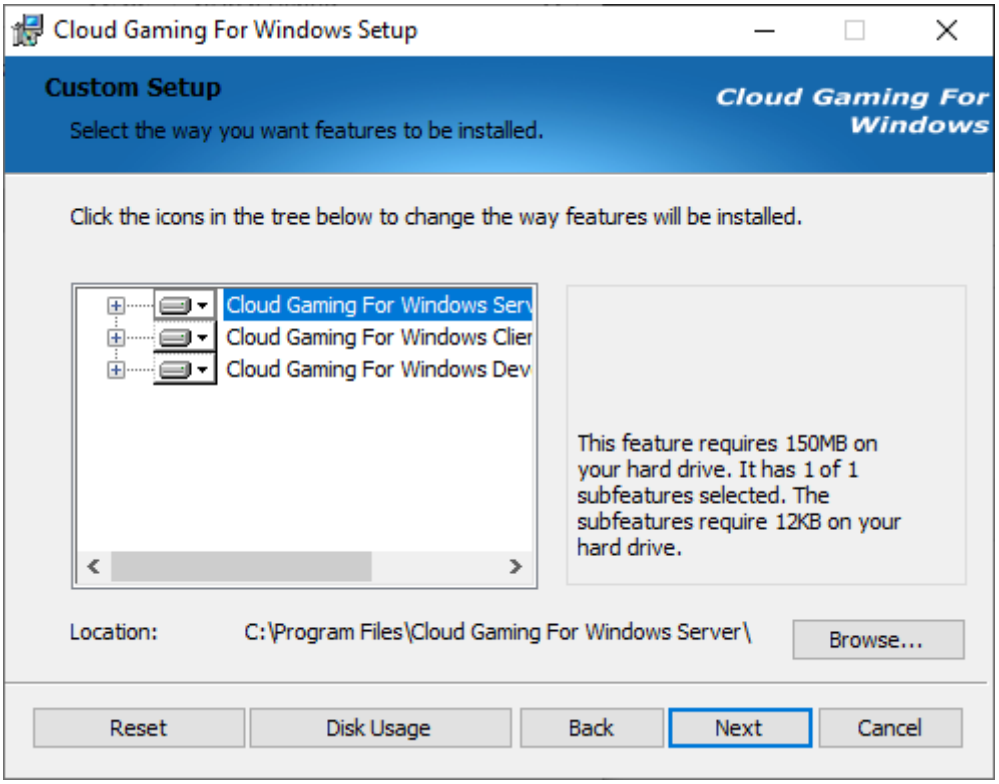
This section is dedicated to the License agreement of the main component. Select checkbox “I accept the terms in the License Agreement” and then click “Next”.



5. Custom Setup: Choose Components for Installation

In this section, user can choose which components to install on the machine. There are three options:

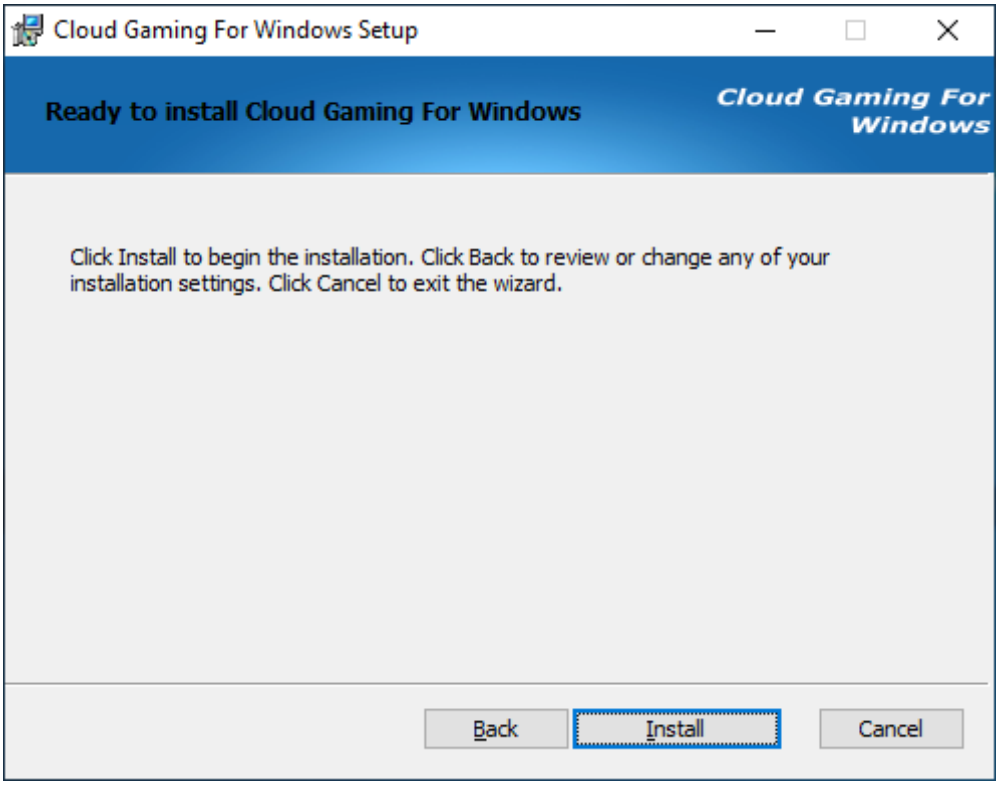
- Cloud Gaming For Windows Server – binaries required to run Server side
- Cloud Gaming For Windows Client – binaries required to run Client side
- Cloud Gaming For Windows Development Environment - preparing environment for developers



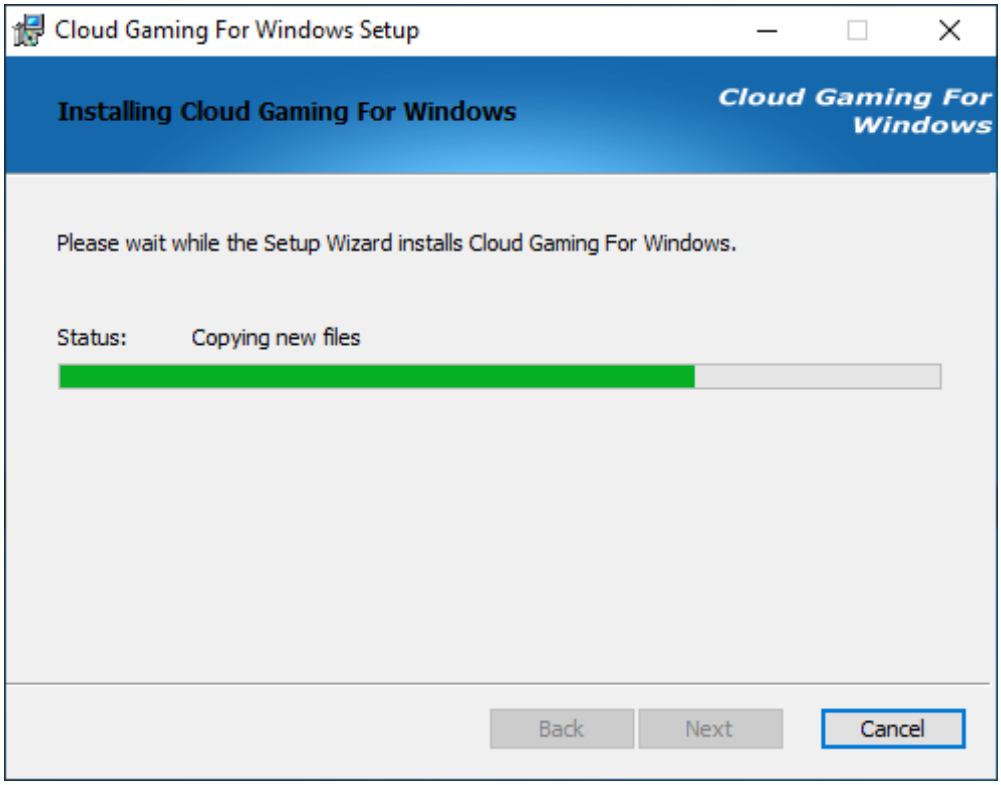
Once selections are made, click "Next".

6. Last screen for Processing Installation

The following screen appears and user can click on the “Install” button to start installation.

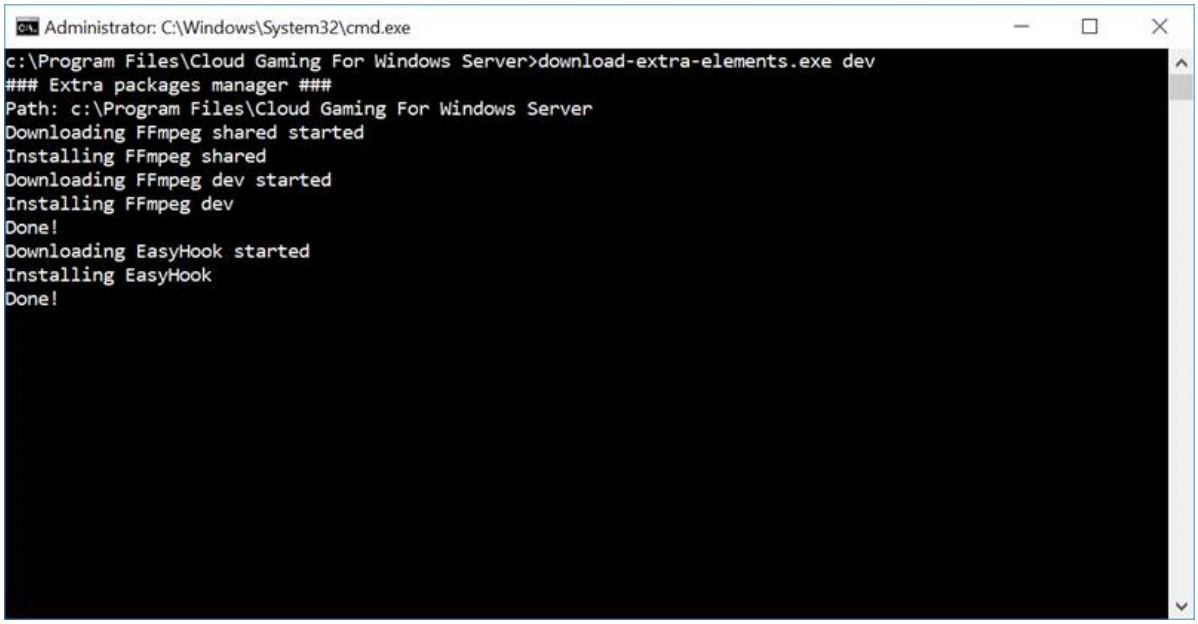


A new screen appears which updates the status of installation.



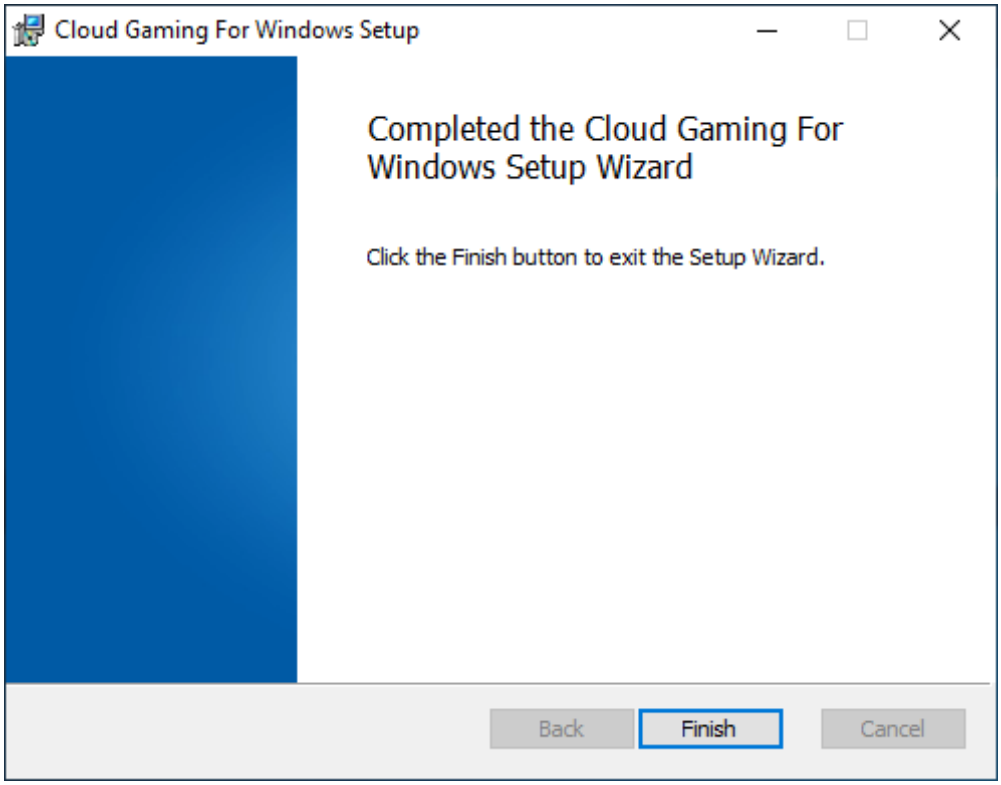
7. Downloading Additional Components

This part is responsible for downloading additional components such as FFmpeg and EasyHook from the Web (only if user agreed to download them). User does not have to do anything additional here. These components are required by the environment but not distributed.



8. Successful Installation of Main Component

The last screen shows that set up of Cloud Gaming for Windows is completed successfully. User can click on “Finish” button.



9. Installation of Intel® Media SDK component

Intel® Media SDK component will start getting installed automatically.

After installation with installer, user will have to install Microsoft Visual Studio 2017 or Microsoft Visual C++ Redistributable for Visual Studio 2010 and Microsoft Visual C++ Redistributable for Visual Studio 2017 (depends if user chooses runtime environment or development environment). The links are given in Additional Specific Software Requirements on page 34

2.6 Manual Environment Configuration

If a user chooses to configure the environment manually, steps are described below:

1. Install MS Visual Studio 2017
2. Install Intel Media SDK 2018 R2
3. Install Vulkan® SDK version 1.1.114.0 - https://vulkan.lunarg.com/sdk/home#sdk/downloadConfirm/1.1.114.0/windows/VulkanSDK-1.1.114.0-Installer.exe
4. Download *Cloud Gaming For Windows* source code from https://github.com/OpenVisualCloud/Cloud-Gaming-Windows-Sample
5. Download all required external resources and place them in proper directories:
   a. Static libraries (FFmpeg and EasyHook) files (.lib) should be placed in:
      - deps.win64->lib (for 64 bit version of files)
      - deps.win32->lib (for 32 bit version of files)
   b. Header files (.h) should be placed in:

- FFmpeg:
  - deps.win64->include->ffmpeg (for 64 bit version of files)
  - deps.win32->include->ffmpeg (for 32 bit version of files)
- EasyHook:
  - deps.win64->include->easyhook (for 64 bit version of files)
  - deps.win32->include->easyhook (for 32 bit version of files)

c. Executable files (.exe) and dynamic linked libraries files (.dll) should be placed in:

- FFmpeg:
  - deps.win64->bin->ffmpeg (for 64 bit version of files)
  - deps.win32->bin->ffmpeg (for 32 bit version of files)
- EasyHook:
  - deps.win64->bin->easyhook (for 64 bit version of files)
  - deps.win32->bin->easyhook (for 32 bit version of files)

2.7 Build

Build process is executed by MS Visual Studio 2017. To perform it, user should open *GamingAnywhere.sln* solution. IDE should allow to build x64 configuration only (in case of manual environment configuration, it should be possible to build Win32 configuration also). To build, click "select build configuration" e.g.:

Figure 2. Build Configuration

Then click "*Build->Build Solution.*"

Build results should be divided into client and server side. Installer is independent from client and server.

2.8 Run Game Server

If environment configuration was done automatically, there are two methods of launching *Cloud Gaming For Windows* server. First method is to launch using icon placed in the *Start* menu. There are separate icons for client and server.

Figure 3. Start Menu-icon

Second method is to run *Cloud Gaming For Windows* server via command line shell.

To run *Cloud Gaming For Windows* game server, open Windows command line shell and navigate to the folder with server binaries:

Example:

```
C:\ >cd <path_to_the_server_bin_folder>
```

Next, ensure *ga-server-manager.exe* file and other binaries are present in that folder.

Then execute *ga-server-manager.exe* file. This creates a process which waits for request from client to launch the selected game. When client sends a request, chosen game will be made to run.

To launch server with optional configuration file, following command line should be executed:

```
c:\><path_to_the_server_bin_filder>ga-server-manager.exe <optional_path_to_conf_file>
```

```
<optional_path_to_conf_file> - path to the server configuration file.
```

If *ga-server-manager.exe* is run without `<optional_path_to_conf_file>` parameter, application will get default configuration file which is placed in *config* subdirectory in the directory with binaries.

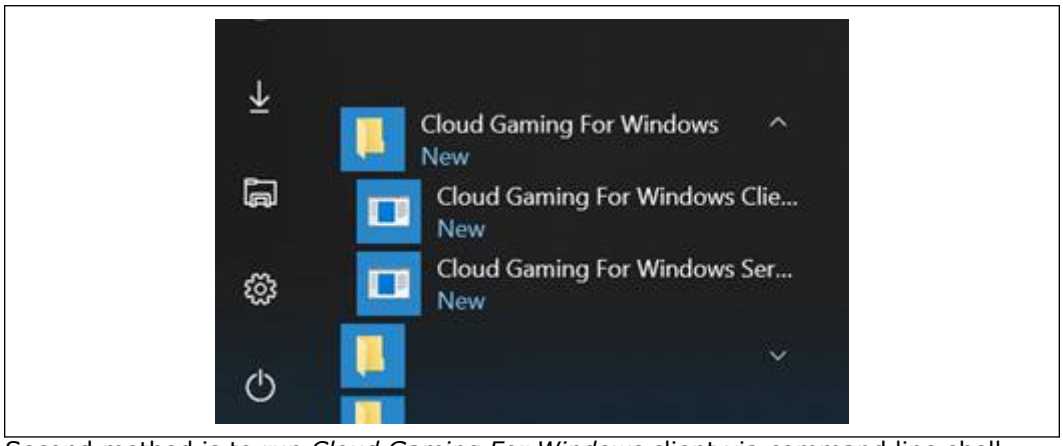
NOTES

- Before starting *Cloud Gaming For Windows* game server, make sure that process which will be run by *Cloud Gaming For Windows* is not running already.
- The server configuration file needs to be modified to point to the location of executable game file - `Game.exe` (it can be any game that user intends to run).

2.9 Run Client

Same as in case of server, if environment configuration was done automatically, there are two methods of launching *Cloud Gaming For Windows* client. First method is to launch the client using icon placed in the *Start* menu. There are separate icons for client and server.

Figure 4. Start Menu-icon



Second method is to run *Cloud Gaming For Windows* client via command line shell.

To run *Cloud Gaming For Windows* client, run Windows command line shell and navigate to the folder with client binaries:

```
C:\ >cd <path_to_the_client_bin_folder>
```

Next, ensure *ga-client.exe* file and other binaries are present in that folder.

Then launch *ga-client.exe* file, which sends request to the server. After sending request, client waits for some time and tries to connect to the running game.

NOTE

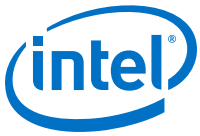
Before running the client, set proper IP address of the server in the configuration file.

To launch client, following command line should be executed:

```
C:\<path_to_the_client_bin_folder>ga-client.exe <optional_path_to_conf_file>
```

```
<optional_path_to_conf_file> - path to the client configuration file.
```

If *ga-client.exe* is run without `<optional_path_to_conf_file>` parameter, application will get default configuration file which is placed in `config` subdirectory in the directory with binaries.



NOTE

If client doesn't connect to the game, try to run `ga-client.exe` one more time. Problem could be caused by above average game launching time.

2.10 E2E Latency Measurement

The software has an option to enable end-to-end latency measurement, that is, time that elapses between hypothetical user action and the display time of first frame in which the effect of the action could be observed. To enable this, in the client configuration file, find a line looking like:

```
latency-measurement = false
```

and change it to:

```
latency-measurement = true
```

NOTE

No server configuration change is required.

The latency (in milliseconds) will be displayed in top-left corner of the screen when in-game. Measurement results can be saved to a text file. In the client configuration file, uncomment the latency-measurement-log line, and provide the name of the file to which results will be logged:

```
#latency-measurement-log = filename
```

and change it to:

```
latency-measurement-log = C:\Users\Gamer\Desktop\latency.log
```

NOTE

To use this feature, the latency measurement switch described above must be turned on.

Appendix A Intel® VCA Utility (vcactl)

A.1 Introduction to vcactl

The vcactl utility allows the user to interact with the Intel® VCA card and nodes. To execute the utility, the user must be either a member of the special group vcausers, which is created during utility installation; a member of the sudoers group (that is, they must have sudo privileges on the system); or a user with root privileges. In the first and last cases (a member of vcausers or a root user), no extra commands must be typed. In the second case (a member of the sudoers group), all commands must be prepended by sudo (sudo followed by a space character).

To add a user to the vcausers group, use the following command as either root or a root-privileged user (e.g., a sudoers user using sudo):

```
usermod -aG vcausers <username>
```

NOTE

To add a username to vcausers, a relogin or reboot is required.

A.2 vcactl Usage

Use the following syntax and structure to use the vcactl utility.

```
vcactl [execution_mode] command [subcommand] [card_id [cpu_id]] [command_params]
```

The available execution modes are:

```
-v : log verbose level set to DEBUG

-vv : log verbose level set to FULL-INFO

--skip-modprobe-check : turn off checking whether vca device is
ready

--force : force command execution (WARNING: you do it at your own
risk!)
```

A.3 Command Line Reference

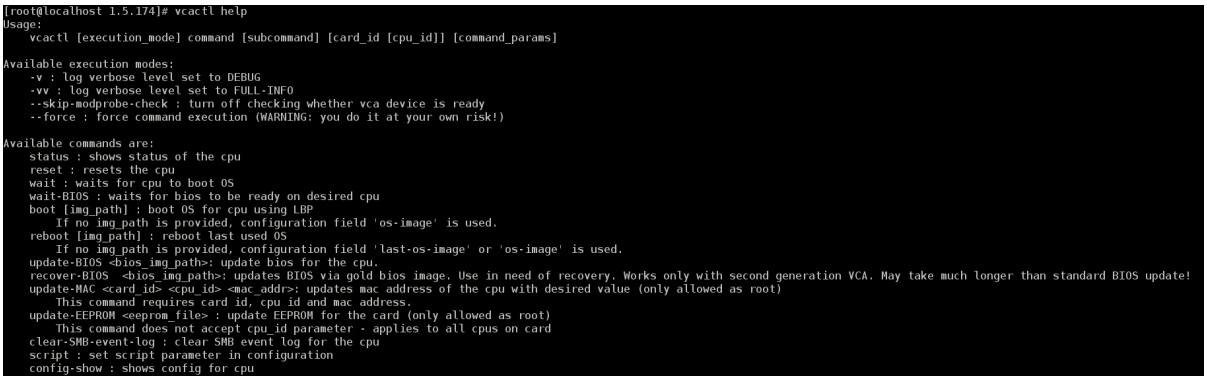
A.3.1 vcactl help

A.3.1.1 Usage

```
vcactl help
```

A.3.1.2 Description

Prints out help information about how to use `vcactl`.



A.3.2 vcactl wait-BIOS

A.3.2.1 Usage

```
vcactl wait-BIOS [<card_id> [<cpu_id>]]
```

A.3.2.2 Description

Waits for CPU `<card_id> [<cpu_id>]` to enter `BIOS is up and running` mode, typically used after host reboot or `vcactl reset` command. <cpu_id> is optional.

Reports an error if CPU cannot enter `BIOS is up and running` mode.

```
#vcactl wait-BIOS
Card: 1 Cpu: 0 - BIOS is up and running!
Card: 0 Cpu: 1 - BIOS is up and running!
Card: 0 Cpu: 0 - BIOS is up and running!
Card: 1 Cpu: 1 - BIOS is up and running!
```

A.3.3 vcactl status

A.3.3.1 Usage

```
vcactl status [<card_id> [<cpu_id>]]
```

A.3.3.2 Description

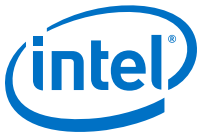
Reads the status of the CPUs in the system. Normally this command should return `bios_up`. Use `<card id>`to read all CPUs on the specified card. Use `<card id> [<cpu id>]` to read the status of the specified CPU. `<cpu id>` is optional.

```
#vcactl status
Card: 0 Cpu:0 STATE: bios_up
Card: 0 Cpu:1 STATE: bios_up
Card: 1 Cpu:0 STATE: bios_up
Card: 1 Cpu:1 STATE: bios_up
```

A.3.3.3 Result

Table 7. Possible Results for `vcactl status`

Result	Description
`link_down`	if card CPU is down due to any reason if CPU link to PLX is down (usually means that PLX eprom is invalid)
`power_off`	if card CPU is switched off or shutdown
`powering_down`	If card CPU is shutting down
`bios_down`	CPU BIOS failed upon initialization or still booting (usually when memory training fails such state is reported).
`bios_up`	Link is up and BIOS is ready for handshake.
`bios_ready`	Handshake has succeeded with BIOS.
`bios_done`	BIOS is in a temporary state after allocating resources for completing the operation.
`flashing`	BIOS is being updated.
`booting`	OS is being booted.
`drv_probe_done`	Main PCIe* driver has been loaded.
`drv_probe_error`	Main PCIe* driver couldn't be loaded.
`dhcp_in_progress`	Card is obtaining IP address from DHCP.
`dhcp_done`	Card obtained IP address from DHCP.
`dhcp_error`	Card could not obtain IP address from DHCP.
`nfs_mount_done`	Card mounted NFS resources.
`nfs_mount_error`	Card could not mount NFS resources.
`os_ready`	OS boot completed.
`net_device_up`	Intel® VCA network stack is up but OS is not yet loaded.
`net_device_down`	Intel® VCA network stack has been switched off.
`net_device_ready`	Intel® VCA network stack is up, OS is booted, and IP address is assigned.
`net_device_no_ip`	Intel® VCA network stack is up and OS is booted, but IP address is not assigned (for example, DHCP response has been not received).
`resetting`	Intel® VCA CPU is being reset.
`error`	Error occurred different than the specific errors listed above.



A.3.4 vcactl update-BIOS

A.3.4.1 Usage

```
vcactl update-BIOS [<card_id> [<cpu_id>]] <biosImageFile>
```

A.3.4.2 Description

Updates BIOS (flash SPI user BIOS partition) on specified CPU(s) in the system. Before running this command, CPUs must be in the `bios_up` or `bios_ready` state. (Use `vcactl reset` and `vcactl wait-BIOS` to force CPUs to enter the `bios_up` state.) Host reboot is not required after `vcactl update-BIOS`.

G3 (mechanical off) is required after `vcactl update-BIOS`. (To perform G3, execute the power off command on the host and then disconnect from the power supply for 15 seconds.)

```
# vcactl status
Card: 0 Cpu:0 STATE: bios_up
Card: 0 Cpu:1 STATE: bios_up
Card: 1 Cpu:0 STATE: bios_up
Card: 1 Cpu:1 STATE: bios_up
```

A.3.4.3 Result

The CPU status transitions from `bios_up` or `bios_ready` to `flashing` and completes in the `bios_ready` state. A temporary state of `bios_done` may be reported, as well.

If there is a failure, an error status is reported (via the `vcactl status` command).

A.3.5 vcactl recover-BIOS

A.3.5.1 Usage

```
vcactl recover-BIOS [<card_id> [<cpu_id>]] <biosImageFile>
```

A.3.5.2 Description

Updates BIOS via the Gold BIOS image. Use when in need of recovery. Before running this command, the Intel® VCAC-R nodes must be in the `bios_up` state. This command may take much longer than the standard BIOS update.

A.3.6 vcactl update-EEPROM

A.3.6.1 Usage

```
vcactl update-EEPROM [<card_id>] <eepromImageFile>
```

NOTE

This command should be executed as `root`.

A.3.6.2 Description

Updates PCIe* firmware on all Intel® VCA cards or on the specified card. After successful completion of the command, the host system is reboot to work with the new firmware on the card. This command requires root privileges. The Intel® VCA nodes must be in the `bios_up` state before running this command.

```
#vcactl status
Card: 0 Cpu:0 STATE: bios_up
Card: 0 Cpu:1 STATE: bios_up
Card: 1 Cpu:0 STATE: bios_up
Card: 1 Cpu:1 STATE: bios_up

#vcactl update-EEPROM eeprom_8183ffee.bin
EEPROM update started!
Update EEPROM process started (for card 0). Do not power down system!
Update EEPROM for first PCIe switch successful!
Update EEPROM for second PCIe switch successful!
Update EEPROM successful (for card 0). Reboot system is required to reload EEPROM.
EEPROM update started!
Update EEPROM process started (for card 1). Do not power down system!
Update EEPROM for first PCIe switch successful!
Update EEPROM for second PCIe switch successful!
Update EEPROM successful (for card 1). Reboot system is required to reload EEPROM.
```

A.3.6.3 Result

The EEPROMs on the card(s) are updated upon successful completion, at which point a reboot of the system is required.

If there is a failure, an error status is reported after completion of the command. In these circumstances, a retry of the update is strongly recommended prior to any reboot; otherwise, the card may be damaged.

A.3.7 `vcactl reset`

A.3.7.1 Usage

```
vcactl reset [<card_id> [<cpu_id>]]
```

A.3.7.2 Description

Resets the specified CPUs to `bios_up` state. Run `vcactl wait-BIOS` after `vcactl reset` to wait for `vcactl reset` to complete. Normally, the CPUs should enter the BIOS is up and running mode.

```
#vcactl reset
#vcactl wait-BIOS
Card: 0 Cpu:0 STATE: BIOS is up and running
Card: 0 Cpu:1 STATE: BIOS is up and running
Card: 1 Cpu:0 STATE: BIOS is up and running
Card: 1 Cpu:1 STATE: BIOS is up and running
```

A.3.8 vcactl boot

A.3.8.1 Usage

```
vcactl boot [<card_id> [<cpu_id>]] [os_image_file.img|vcablk0
(for vca-disk)]
```

A.3.8.2 Description

Boots the operating system via Leverage Boot Protocol. Image is from the default location (see the `vcactl config os-image` command in section vcactl config on page 60) or from the image file specified in the command. The command requires that the card CPU is in the `bios_up` or `bios_ready` state. Use `vcactl wait` to wait for the completion of the `vcactl boot` command.

NOTE

For booting via blockio (`vcablk0` parameter), there is a need to open it first, and add reference to `vcactl blockio open` subcommand.

It is possible to use the `last-os-image` field as a parameter to boot.

```
#vcactl boot vca_baremetal_4p4_centos7.2_1.5.174.img
#vcactl wait
Card: 0 Cpu:0 STATE: Net device ready!
Card: 0 Cpu:1 STATE: Net device ready!
Card: 1 Cpu:0 STATE: Net device ready!
Card: 1 Cpu:1 STATE: Net device ready!
```

A.3.8.3 Result

During the OS boot, the reported state is `booting` (the `bios_done` state may be visible, also). On successful completion, each CPU is in the `net_device_ready` state. If there is a failure, the specific error cause is reported (via the `vcactl status` command) as one of `drv_probe_error`, `dhcp_error`, `nfs_mount_error`, `net_device_no_ip`, or `generic error`.

A.3.9 vcactl reboot

A.3.9.1 Usage

```
vcactl reboot [<card_id> [<cpu_id>]] [os_image_file.img]
```

A.3.9.2 Description

Reboots the OS (resets the CPU and boots the selected OS) from the host via Leverage Boot Protocol. Image is from the default location (see the `vcactl config os-image` command in section vcactl config on page 60) or from the image file

specified in the command. The image file should have group access set to `vcausers`, otherwise wait will give TIMEOUT. Use `vcactl wait` to wait for the completion of the `vcactl boot` command.

```
#vcactl reboot
#vcactl wait
Card: 0 Cpu:0 STATE: Net device ready!
Card: 0 Cpu:1 STATE: Net device ready!
Card: 1 Cpu:0 STATE: Net device ready!
Card: 1 Cpu:1 STATE: Net device ready!
```

A.3.9.3 Result

During the OS boot, the reported state is `link_down`, `dio_down`, `bios_up`, `bios_ready`, `booting`, `net_device_up`, `dhcp_in_progress`, or `os_ready` (the `bios_done`, `mod_probe_done`, `dhcp_done`, and `nfs_mount_done` states may be visible for a short time, also). On successful completion, each CPU is in the `net_device_ready` state. If there is a failure, the specific error cause is reported (via the `vcactl status` command) as one of `drv_probe_error`, `dhcp_error`, `nfs_mount_error`, `net_device_no_ip`, or `generic error`.

A.3.10 vcactl wait

A.3.10.1 Usage

```
vcactl wait [<card_id> [<cpu_id>]]
```

A.3.10.2 Description

Waits until the OS boot is completed on the specified CPUs. This command is typically used after `vcactl boot` or `vcactl reboot` to report booting status.

A.3.10.3 Result

The command completes when the OS boot is done; otherwise, it is blocking. On successful boot, all CPUs finish in the `net_device_ready` state. If there is a failure, the specific error cause is reported as one of `drv_probe_error`, `dhcp_error`, `nfs_mount_error`, `net_device_no_ip`, or `generic error`.

A.3.11 vcactl temp

A.3.11.1 Usage

```
vcactl temp [<card_id> [<cpu_id>]]
```

A.3.11.2 Description

Retrieves the temperature of one or more CPUs in the system. If no card or CPU is specified, the command returns the temperature of all nodes on all cards.

A.3.11.3 Result

The current card output temperature and the node processor temperatures are reported for each card.

```
[root@vcademo VCA_1.2.81]# vcactl temp
Card 0 Cpu 0:
coretemp-isa-0000
Adapter: ISA adapter
Physical id 0:  +69.0°C  (high = +105.0°C, crit = +105.0°C)
Core 0:         +67.0°C  (high = +105.0°C, crit = +105.0°C)
Core 1:         +66.0°C  (high = +105.0°C, crit = +105.0°C)
Core 2:         +67.0°C  (high = +105.0°C, crit = +105.0°C)
Core 3:         +67.0°C  (high = +105.0°C, crit = +105.0°C)
Card 0 Cpu 1:
coretemp-isa-0000
Adapter: ISA adapter
Physical id 0:  +72.0°C  (high = +105.0°C, crit = +105.0°C)
Core 0:         +70.0°C  (high = +105.0°C, crit = +105.0°C)
Core 1:         +69.0°C  (high = +105.0°C, crit = +105.0°C)
Core 2:         +69.0°C  (high = +105.0°C, crit = +105.0°C)
Core 3:         +69.0°C  (high = +105.0°C, crit = +105.0°C)
Card 0 Cpu 2:
coretemp-isa-0000
Adapter: ISA adapter
Physical id 0:  +78.0°C  (high = +105.0°C, crit = +105.0°C)
Core 0:         +77.0°C  (high = +105.0°C, crit = +105.0°C)
Core 1:         +77.0°C  (high = +105.0°C, crit = +105.0°C)
Core 2:         +76.0°C  (high = +105.0°C, crit = +105.0°C)
Core 3:         +76.0°C  (high = +105.0°C, crit = +105.0°C)
Card 1 Cpu 0:
coretemp-isa-0000
Adapter: ISA adapter
Physical id 0:  +56.0°C  (high = +105.0°C, crit = +105.0°C)
Core 0:         +56.0°C  (high = +105.0°C, crit = +105.0°C)
Core 1:         +54.0°C  (high = +105.0°C, crit = +105.0°C)
Core 2:         +55.0°C  (high = +105.0°C, crit = +105.0°C)
Core 3:         +55.0°C  (high = +105.0°C, crit = +105.0°C)
Card 1 Cpu 1:
coretemp-isa-0000
Adapter: ISA adapter
Physical id 0:  +56.0°C  (high = +105.0°C, crit = +105.0°C)
Core 0:         +55.0°C  (high = +105.0°C, crit = +105.0°C)
Core 1:         +54.0°C  (high = +105.0°C, crit = +105.0°C)
Core 2:         +54.0°C  (high = +105.0°C, crit = +105.0°C)
Core 3:         +54.0°C  (high = +105.0°C, crit = +105.0°C)
Card 1 Cpu 2:
coretemp-isa-0000
Adapter: ISA adapter
Physical id 0:  +64.0°C  (high = +105.0°C, crit = +105.0°C)
Core 0:         +63.0°C  (high = +105.0°C, crit = +105.0°C)
Core 1:         +63.0°C  (high = +105.0°C, crit = +105.0°C)
Core 2:         +63.0°C  (high = +105.0°C, crit = +105.0°C)
Core 3:         +63.0°C  (high = +105.0°C, crit = +105.0°C)
```

A.3.12 vcactl blockio open

A.3.12.1 Usage

```
vcactl blockio open <card_id> <cpu_id> [vcablk<N>] [[RO|RW
<file_path>]|[ramdisk <size_mb>]]
```

where 0 ≤ N < 8.

A.3.12.2 Description

Opens block IO device.

A.3.12.3 Examples

```
vcactl blockio open 0 1 vcablk3 ramdisk 20
```

```
vcactl blockio open 0 1 vcablk3 RO ~/disk.img
```

A.3.13 vcactl blockio list

A.3.13.1 Usage

```
vcactl blockio list <card_id> <cpu_id>
```

```
vcactl blockio list <blockio_id>
```

A.3.13.2 Description

Lists block IO devices.

```
[root@localhost 2.0.201]# vcactl blockio open 1 0 vcablk0 RW ~/tmp/test.img
[root@localhost 2.0.201]# vcactl blockio open 1 1 vcablk3 RW ~/tmp/test.img
[root@localhost 2.0.201]#
[root@localhost 2.0.201]# vcactl blockio list
Card   Cpu      Name     State     Mode  Size(MB) FilePath
-------------------------------------------------------
   0     0   vcablk0  disabled        -         - -
   0     1   vcablk0  disabled        -         - -

   1     0   vcablk0   enabled       RW         0 /root/tmp/test.img
   1     1   vcablk0  disabled        -         - -
   1     1   vcablk3   enabled       RW         0 /root/tmp/test.img
```

A.3.14 vcactl blockio close

A.3.14.1 Usage

```
vcactl blockio close [<card_id>] [<cpu_id>] [vcablk<N>]
```

where 0 ≤ N < 8.

A.3.14.2 Description

Closes block IO device.

A.3.14.3 Examples

```
vcactl blockio close 0 1 vcablk3

vcactl blockio close vcablk0
```

A.3.15 `vcactl network`

A.3.15.1 Usage

```
vcactl network <subcommand> [<card_id> [<cpu_id>]]
```

A.3.15.2 Description

Reads current network configuration of the card CPU to learn the IP address assigned by DHCP and the MAC addresses of the network interfaces of the Dom0 OS and DomU OS. The subcommands are the following and work on all or on the specified node:

- `ip` – Displays IP address of the (specified) node's eth0 interface.
- `ip6` – Displays IP v6 address of the (specified) node's eth0 interface.
- `mac` – Displays MAC address of the (specified) node's eth0 interface so that user may configure DHCP server with that information.
- `vm-mac` – Displays MAC address of the DomU virtual network interface of the (specified) node's so that DHCP server can be set accordingly.
- `all` – Displays all information of the (specified) node's eth0 interface.
- `stats` – Displays statistics of the (specified) node's eth0 interface.
- `dhcp-renew` – Forces renewal of DHCP address for (specified) nodes.

A.3.16 `vcactl ICMP-watchdog`

A.3.16.1 Usage

```
vcactl ICMP-watchdog <1/0> [<card_id> [<cpu_id>]] [<ip_address>]
```

A.3.16.2 Description

Enables (1) or disables (0) ICMP watchdog service on the host that pings the specified node. The ICMP ping address is selected according to the current configuration. Optionally, the command may explicitly specify the IP address, so that the command can be used to monitor DomU activity.

A.3.17 vcactl update-MAC

A.3.17.1 Usage

```
vcactl update-MAC <card_id> <cpu_id> <macAddressCanonical>
```

A.3.17.2 Description

Updates MAC address (flash SPI GBE partition) on the specified CPU in the system. `card_id` and `cpu_id` are mandatory parameters for this command.

Command should be executed only in `bios_up` or `bios_ready`state.

A.3.17.3 Result

Status of the CPU transitions from `bios_up` or `bios_ready` to `flashing` and completes in the `bios_ready` state. If there is a failure, an error status is reported (via the `vcactl status` command).

A.3.18 vcactl script

A.3.18.1 Usage

```
vcactl script [<card_id> [<cpu_id>]] <scriptFile>
```

A.3.18.2 Description

Configures a bash script for the card CPU to perform Linux user-specific configuration that might be any configuration action such as:

- IP address assignment per virtIO interface or enabling the DHCP client
- Starting DomU guest OS

A.3.19 vcactl config

A.3.19.1 Usage

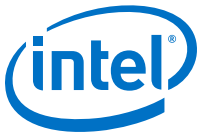
```
vcactl config [<card_id> <cpu_id>] <parameter-name> <parameter-value>
```

The available options and settings are listed in the table below:

Table 8. vcactl config parameters

Parameter Name	Available Values	Description
Global options		
`auto-boot`	1 – enable, 0 – disable	Enable/disable auto booting. Note that for auto-boot to work, the directory containing the boot image and all its parent directories must either be world-readable or be owned by the vcausers group.
`debug-enabled`	1 – enable, 0 – disable	Enable/disable debug commands.

continued...



Parameter Name	Available Values	Description
`link-up-timeout-ms`	# of milliseconds	Set timeout for link-up after reset in milliseconds. Default is 2000 milliseconds (2 seconds)
`handshake-irq-timeout-ms`	# of milliseconds	Set timeout for link-up after reset in milliseconds. Default is 30000 milliseconds (30 seconds).
`alloc-timeout-ms`	# of milliseconds	Set timeout for RAMDISK allocation in BIOS. Default 100 is milliseconds.
`cmd-timeout-ms`	# of milliseconds	Set timeout for LBP command reaction in BIOS. Default is 1000 milliseconds.
`mac-write-timeout-ms`	# of milliseconds	Set timeout for MAC update command processing time in BIOS. Default 1000 milliseconds.
`default-daemon-script`	Name and path to daemon script file	Name and path to daemon script file that is executed upon watchdog expiration.
`wait-cmd-timeout-s`	# of seconds	Set timeout for WAIT command in vcactl (time needed to boot OS up). Default is 160 seconds.
`wait-bios-cmd-timeout-s`	# of seconds	Set timeout for WAIT-BIOS command in vcactl for BIOS UP time needed. Default is 300 seconds.
`wait-bios-cmd-flashing-s`	# of seconds	Set timeout for WAIT-BIOS command in vcactl for completion of BIOS update. Default is 1200 seconds.
`ICMP-ping-interval-s`	# of seconds	Set ICMP watchdog period. Default is 1 second.
`ICMP-response-timeout-s`	# of seconds	Set ICMP response timeout. Default is 10 second.
`va-min-free-memory-enabled`	1 – enable, 0 – disable	Enable/disable minimum value (512 MB) of non-cached memory by Linux OS filesystem. This ensures that there will be enough free space to allocate buffers needed by virtual network over PCIe*. Option refers to host system.
Per card node options		
`os-image`	Path to image	Path to bootable OS image
`last-os-image`	Path to image (not user-configurable)	Shows last booted image on node
`script`	Path to script	Path and script file name that is executed on card after OS image has been booted successfully. /etc/vca_config.d/vca_<slot_id><cpu_id>_default.sh
`daemon-script`	Path to script	Name and path to daemon script file that is executed upon watchdog expiration.
`ip`	IP address or dhcp	Set the node IP address or enable DHCP operation. Default is 172.31.<slot_id * 3 + cpu_id + 1>.1. To enable DHCP (since Intel® VCA Software Release 1.2.) for automatic IP address assignment, set to dhcp. Enabling DHCP is allowed only if the node is already added to an existent bridge interface.
`mask`	# of bits	Set mask length in bits. Default is 24 bits.
`gateway`	IP address	Set gateway address. Default is 172.31.<slot_id * 3 + cpu_id + 1>.254.
`host-ip`	IP address	Set host IP address Default is 172.31.<slot_id * 3 + cpu_id + 1>.254.
`host-mask`	# of bits	Set host mask length in bits. Default is 24 bits.
`cpu-max-freq-non-turbo`	# of 100 MHz	Set CPU maximum frequency. Default is 0 (turbo enabled).

continued...

Parameter Name	Available Values	Description
bridge-interface	Name of the bridge network interface on the host or none to disable bridging	To enable bridging to the node, use the name of an existent bridge interface (for example, vcabr0). To disable bridging to the node, use none. Disabling is allowed if DHCP is disabled (meaning that the IP address already has a static value). The command is available since Intel® VCA Software Release 1.2.
node-name	Name of the node	Provide a unique name for each node in the system.
nfs-server	IP address or FQDN of NFS host for persistent image boot	Provide a path for the node to access persistent boot filesystem over the network. If using FQDN, name resolution must be operational
nfs-path	Path on NFS server where persistent filesystem is stored	Path to share on the NFS server. Use %s to indicate "node-name" value in the path.
block-devs	N/A	This is a section which describes some additional information about each defined VCA block device. By default there is only one entry: vcablk0. Another devices will appear if you add them using 'vcactl blockio open' command.
vcablk<N>	N/A	It is a root entry for each defined block device (where 0 ≤ N < 8).
mode	RW – read/write, RO – read-only, ramdisk – device will be created into RAM	This is a way (RW or RO) how VCA block device will be created. User can choose if block device may be read-only or with read/write permissions.
path	Path to block device.	Path to file used as VCA BlockIO device.
ramdisk-size-mb	Size of block device created inside node RAM.	VCA BlockIO device can be also created in node RAM memory (for example as SWAP partition as RAM is very fast memory type).
enabled	1 – enable, 0 – disable	Option used to enable/disable block device. All enabled devices will be created/opened during booting node phase.
va-min-free-memory-enabled-node	1 – enable, 0 – disable	Enable/disable minimum value (512 MB) of non-cached memory by Linux OS filesystem. This ensures that there will be enough free space to allocate buffers needed by virtual network over PCIe*. Option refers to node system (Linux only).

A.3.20 vcactl config-use

A.3.20.1 Usage

```
vcactl config-use
```

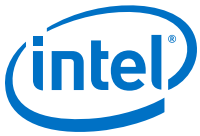
A.3.20.2 Description

Triggers the `vcactld` daemon to use the last configuration.

A.3.21 vcactl config-show

A.3.21.1 Usage

```
vcactl config-show [<card_id> [<cpu_id>]] <parameter-name>
<parameter-value>
```



A.3.21.2 Description

Shows the current configuration of the `vcactld` daemon.

```
[root@vcademo ~]# vcactl config-show 0 0
Global configuration:
  auto-boot: 1
  debug-enabled: 0
  link-up-timeout-ms: 2000
  handshake-irq-timeout-ms: 30000
  alloc-timeout-ms: 100
  cmd-timeout-ms: 1000
  mac-write-timeout-ms: 1000
  default-daemon-script:
  wait-cmd-timeout-s: 60
  wait-bios-cmd-timeout-s: 120
  wait-bios-cmd-flashing-s: 1200
  ICMP-ping-inverval-s: 1
  ICMP-response-timeout-s: 10
card 0 cpu 0:
  os-image:
  last-os-image: /home/vista/vvgold/VCA_1.2.81/vca_baremetal_centos7.1_1.2.81.img
  script:
  daemon-script:
  ip: 172.31.1.1
  mask: 24
  gateway: 172.31.1.254
  host-ip: 172.31.1.254
  host-mask: 24
  cpu-max-freq-non-turbo: 0
  bridge-interface:
  node-name: vca_node_00
  nfs-server: 172.31.1.254
  nfs-path: /mnt/%s
```

A.3.22 vcactl config-default

A.3.22.1 Usage

```
vcactl config-default
```

A.3.22.2 Description

Restores the `vcactld` daemon to default values.

A.3.23 vcactl info-hw

A.3.23.1 Usage

```
vcactl info-hw
```

A.3.23.2 Description

Displays hardware information for the Intel® VCAC-R card and its EEPROM version.

```
#vcactl info-hw
Card 0:VCAC-R, EEPROM version:VCAC-R 1.3(CRC:4bb11a52), Serial Number:not-yet-
readable
```

A.3.24 vcactl info-system

A.3.24.1 Usage

```
vcactl info-system
```

A.3.24.2 Description

Displays information of the software installed on the Intel® VCAC-R card including version and build date.

```
[root@localhost ]# vcactl info-system
Apps version: build: 2.6.265 build on: 2019-09-26 14:22:54 +0000
Modules build: Build number: 2.6.265 build on: 2019-09-26 14:21:49 +0000
```

A.3.25 vcactl pwrbtn-short

A.3.25.1 Usage

```
vcactl pwrbtn-short [<card_id> [<cpu_id>]]
```

A.3.25.2 Description

- Power button toggle of CPU `<cpu_id>`
- Shutdown active OS on the specified CPUs or turn CPU on when it's off. Command imitates short press the power button.

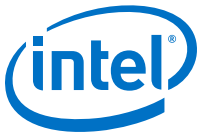
A.3.25.3 Result

- Command completes when CPU is off or on depending on previous state. All CPUs finish in `power_off` state in case of shutting down or in `bios_up` or `bios_ready` state when CPUs were off.

A.3.26 vcactl pwrbtn-long

A.3.26.1 Usage

```
vcactl pwrbtn-long [<card_id> [<cpu_id>]]
```



A.3.26.2 Description

- Power button override five seconds on CPU `<cpu_id>` to cut off Intel® VCA power supply immediately; the OS can be damaged. For comparison, `vcactl pwrbtn-short` first shuts down the OS gracefully, then cuts off Intel® VCA power supply.
- Normally, `vcactl pwrbtn-short` should be used to power off Intel® VCA. The `vcactl pwrbtn-long` command is only used when Intel® VCA is in strange mode and `vcactl pwrbtn-short` cannot power off the Intel® VCA card.
- CPU <cpu_id> is in `power_off` status until next `vcactl pwrbtn-short`.

A.3.26.3 Result

- Command completes when all CPUs finish in `power_off` state.

A.3.27 `vcactl os-shutdown`

A.3.27.1 Usage

```
vcactl os-shutdown [<card_id> [<cpu_id>]]
```

A.3.27.2 Description

Shuts down the running operating system on the specified CPU.

It is not recommended to use the `os-shutdown` command in combination with `pwrbtn-short`. (After using `os-shutdown`, it is necessary to wait some time before executing the `pwrbtn-short` command again to bring up CPU power.)

A.3.28 `vcactl get-BIOS-cfg`

A.3.28.1 Usage

```
vcactl get-BIOS-cfg [<card_id> [<cpu_id>]] [bios_cfg_name]
```

A.3.28.2 Description

Read BIOS configuration of the specified CPU. Available configurations to be read are:

- `sgx`
- `gpu-aperture`
- `tdp`.

A.3.28.3 Example

```
vcactl get-BIOS-cfg 0 1 sgx
```

A.3.29 vcactl set-BIOS-cfg

A.3.29.1 Usage

```
vcactl set-BIOS-cfg [<card_id> [<cpu_id>]] <bios_cfg_name>
<bios_cfg_value>
```

A.3.29.2 Description

Set/change BIOS configuration of specified CPU. Available configurations to be read are:

- `sgx`: Enable/disable Intel® Software Guard Extensions. Set to `enable` or `disable`.
- `gpu-aperture`: Megabytes of graphics memory. Set to 128, 256, 512, 1024, 2048, or 4096.
- `tdp`: Set to 0 to 11, where zero means base value, 1 means base value + 1, etc.

A.3.29.3 Examples

```
vcactl set-BIOS-cfg 1 0 sgx enable
```

```
vcactl set-BIOS-cfg gpu-aperture 1024
```

A.3.30 vcactl info

A.3.30.1 Usage

```
vcactl info <subcmd> [<card_id> [<cpu_id>]]
```

A.3.30.2 Description

Command allows to read specific information about either Intel® VCA SW/FW or Node OS. The subcommands are the following and work on all or on the specified node:

- `BIOS`: Displays BIOS version of the Intel® VCA2 nodes.
- `hw`: Displays hardware information for the Intel® VCA card including if the card is GEN 1 or GEN 2 and its EEPROM version.
- `system`: Displays information of the software installed on the Intel® VCA card including version and build date.
- `node-os`: Displays information of node OS on the Intel® VCA card.

A.3.30.3 Examples

```
vcactl info hw
```

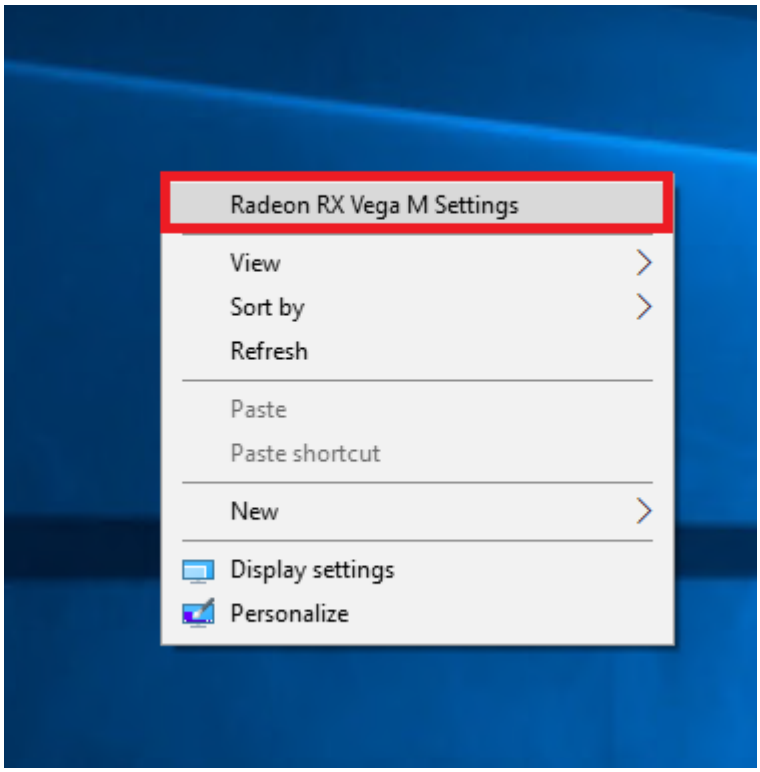
```
vcactl info BIOS
```

Appendix B Setting up Radeon™ GPU for Gaming

When Intel GPU is used as the default, the Radeon™ GPU has to be explicitly set for the game to achieve adequate performance.

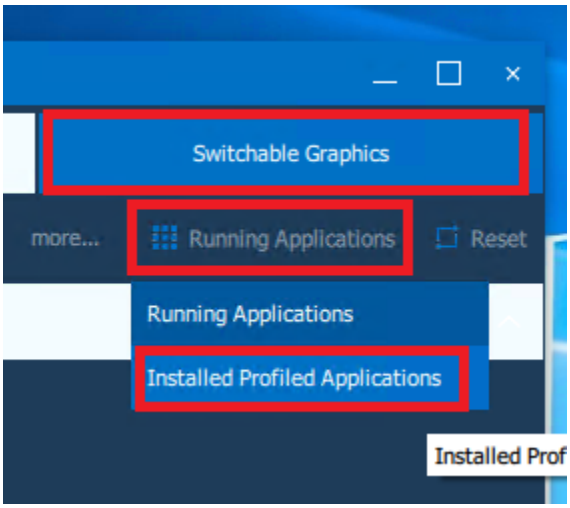
Adding the profile for the game

1. Install the game and see if it works. Use `Task Manager` to locate the executable of the game.
2. Right-click on the desktop, and choose "Radeon RX Vega M Settings" as shown below:



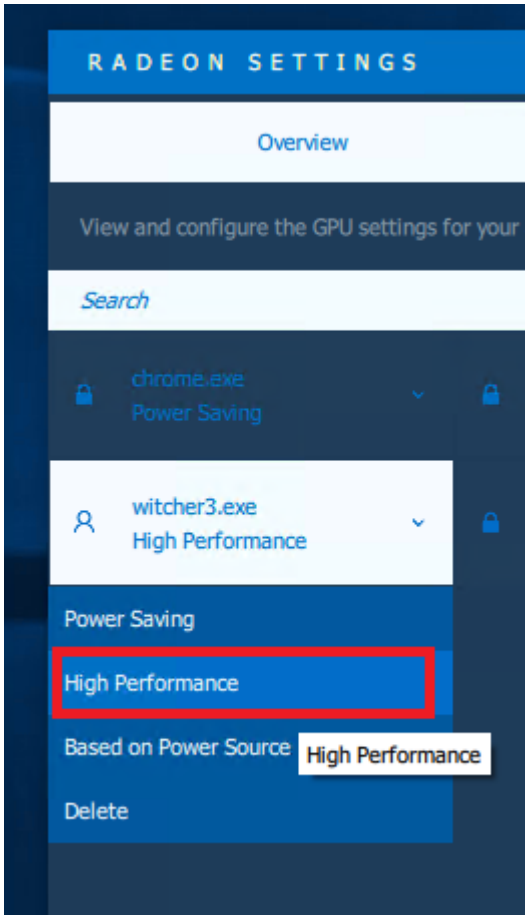
3. In the window that opens, choose `System tab`

4. Then, switch to `Switchable Graphics` view and click on `Running Applications` caption to reveal a dropdown list. Choose `Installed Profiled Applications` as shown in the figure below:



5. Click the `Browse` button and locate the game executable.

6. A box for the game should appear. Make sure that it has `High Performance` enabled. If not, click the arrow and choose appropriate setting for the game executable.



After these steps, the game should run on the fast Radeon graphics card. However, some extra steps may be required during each restart – they are described in next section.

Restoring the setting after restart

After resetting the node, the effect of setting the Radeon GPU to handle the game may be lost. To recover it, the following steps have to be done after each reset:

1. Repeat steps 2 to 4 of the list in above mentioned section `Adding the profile for the game`.
2. On any of the boxes corresponding to game executables (from the dropdown list), choose `High Performance` as in figure in step 6 above.

   NOTE

   This needs to be done even if the box already says `High Performance`.

Now all the games configured as in previous section should operate on Radeon GPU until next reset.

Glossary

acgss

Android Cloud Gaming Software Stack

AIC

Android in Container

NAT

Network Address Translation

PCIe

Peripheral Component Interconnect Express

VCA

Visual Compute Accelerator

VCAC-R

Visual Cloud Accelerator Card - Rendering