Radeon Developer Panel Documentation

Release 2.8.1

AMD Developer Tools

Feb 20, 2023

Contents

1 Graphics APIs, RDNA and GCN hardware, and operating systems **3**

2 Compute APIs, RDNA and GCN hardware, and operating systems **5**

3 Initial setup **7**

4 Remote connections **11**

5 System **13**
- 5.1 My applications . . . 14
- 5.2 My workflows . . . 17
- 5.3 Blocked applications . . . 21
- 5.4 System information . . . 22

6 How to profile your application **25**

7 Settings **29**

8 How to memory trace your application **31**

9 How to capture a raytracing scene from your application **35**

10 Using the Clock settings **39**

11 Connection Log **41**

12 The Radeon Developer Service **43**
- 12.1 Radeon Developer Service for desktop developer system . . . 43
- 12.2 Radeon Developer Service for headless GPU systems . . . 44

13 Bug Report **45**

14 Known Issues **47**
- 14.1 Cleanup After a RadeonDeveloperServiceCLI Crash . . . 47
- 14.2 Windows Firewall Blocking Incoming Connections . . . 47
- 14.3 Disabling Linux Firewall . . . 50
- 14.4 Setting GPU clock modes on Linux . . . 50
- 14.5 Enabling support for RMV tracing on Linux . . . 50

14.6 Radeon Developer Panel connection issues on Linux 50
14.7 Missing Timing Data for DirectX 12 Applications 50
14.8 Radeon Developer Service Port numbers 54
14.9 Problems caused by existing installation of RADV Linux Vulkan driver 54
14.10 Problems caused by the presence of non-AMD GPUs and non-AMD CPUs with integrated graphics . 54

The Radeon Developer Panel is part of a suite of tools that can be used by developers to optimize DirectX® 12, Vulkan® and OpenCL™ applications for AMD RDNA™ and GCN hardware. The suite is comprised of the following software:

- **Radeon Developer Mode Driver** – This is shipped as part of the AMD public driver and supports the developer mode features required for profiling and debugging.
- **Radeon Developer Service (RDS)** – A system tray application that unlocks the Developer Mode Driver features and supports communications with high level tools.
- **Radeon Developer Service - CLI (Headless RDS)** – A console (i.e. non-GUI) application that unlocks the Developer Mode Driver features and supports communication with high level tools.
- **Radeon Developer Panel (RDP)** – A GUI application that allows the developer to configure driver settings and generate profiles from DirectX12, Vulkan and OpenCL applications.
- **Radeon GPU Profiler (RGP)** – A GUI tool used to visualize and analyze the profile data.
- **Radeon Memory Visualizer (RMV)** - A GUI tool used to visualize and analyze the memory trace data.
- **Radeon Raytracing Analyzer (RRA)** - A GUI tool used to visualize and analyze the raytracing data.

  This document describes how the Radeon Developer Panel can be used to capture a profile, memory trace or a raytracing scene for an application on AMD RDNA and GCN graphics hardware. The Radeon Developer Panel connects to the Radeon Developer Service in order to collect a profile, trace or scene.

  RGP documentation: https://radeon-gpuprofiler.readthedocs.io/en/latest/

  RMV documentation: https://radeon-memory-visualizer.readthedocs.io/en/latest/

  RRA documentation: https://radeon-raytracing-analyzer.readthedocs.io/en/latest/

  Note: By default, the driver allocates a maximum of 75 MB video memory per Shader Engine to capture RGP profiles. The driver allocates 300 MB video memory for the single shader engine with instruction tracing enabled. As of v2.6 this can now be configured in the workflow settings.

CHAPTER 1

Graphics APIs, RDNA and GCN hardware, and operating systems

Supported APIs

- DirectX12
- Vulkan

Supported RDNA and GCN hardware

- AMD Radeon RX 7000 series
- AMD Radeon RX 6000 series
- AMD Radeon RX 5000 series
- AMD Radeon VII
- AMD RX Vega 64 and RX Vega 56
- AMD Ryzen™ Processors with Radeon Vega Graphics
- AMD Radeon R9 Fury and Nano series
- AMD Radeon RX 400 and RX 500 series
- AMD Tonga R9 285, R9 380

Supported Operating Systems

- Windows® 10
- Windows® 11
- Ubuntu 22.04 LTS (Vulkan only)

CHAPTER 2

Compute APIs, RDNA and GCN hardware, and operating systems

Supported APIs

- OpenCL
- HIP

Supported RDNA and GCN hardware

- AMD Radeon RX 7000 series
- AMD Radeon RX 6000 series
- AMD Radeon RX 5000 series
- AMD Radeon VII
- AMD RX Vega 64 and RX Vega 56
- AMD Ryzen Processors with Radeon Vega Graphics

Supported Operating Systems

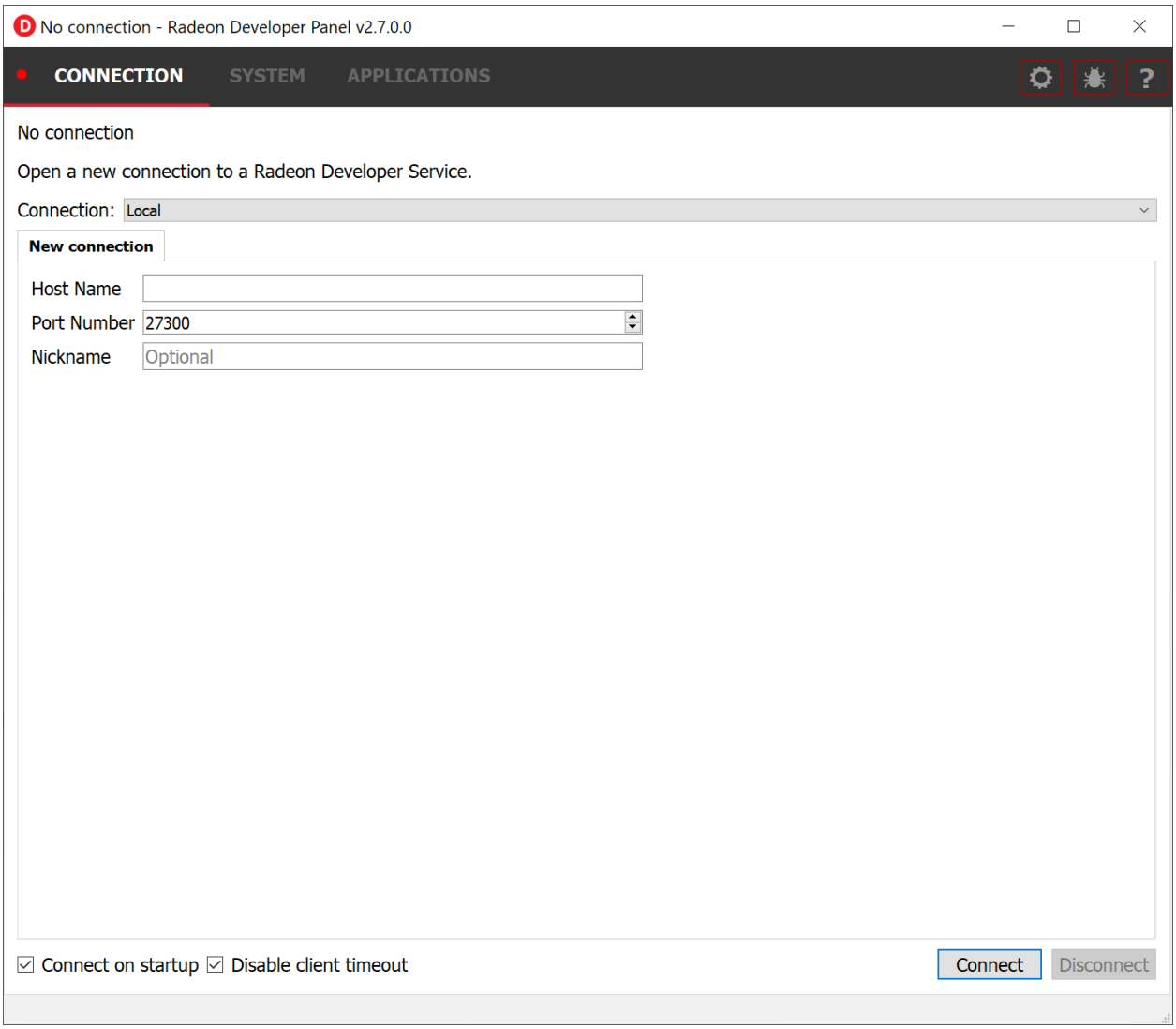
- Windows® 10
- Windows® 11

CHAPTER 3

Initial setup

IMPORTANT: The application you want to profile, trace or capture scenes from must **NOT** already be running. The panel needs to be configured in advance of starting your application.

1) Start the **RadeonDeveloperPanel(.exe)** on your local system. The panel will startup up with the Connection tab already highlighted (see below).



The connection panel has three main elements:

- **Connection status** – to the Radeon Developer Service (currently not connected)
- **Connection dropdown** - choose a previous connection to connect to. **Local** will always be available in this list
- **New connection** – section that allows you specify a new remote connection. New connections will be added to the connections list

2) Connect to a **Local** or **Remote** connection:

   Select an entry from the Connection dropdown, then click the "Connect" button. This will attempt to establish a connection to a **Radeon Developer Service**

Note that the red indicator to the left of the "CONNECTION" tab will change to green to indicate that the connection was successful.

Connections to applications will timeout after a brief period of no API calls being made. For example, a timeout will likely occur when a connected application is suspended by a debug breakpoint or if the application is only occasionally refreshing. Enabling the "Disable client timeout" toggle will stop Radeon Developer Panel disconnecting from inactive clients.

NOTE For Local connections, starting **Radeon Developer Service** is optional. For Remote Connections, a **Radeon**

Developer Service instance must be started on the remote machine (see below)

CHAPTER 4

Remote connections

1) Start the **RadeonDeveloperService(.exe)** on the **remote** system (the machine where the application is to be run). Make a note of the remote system's IP address (open a command prompt and type 'ipconfig').

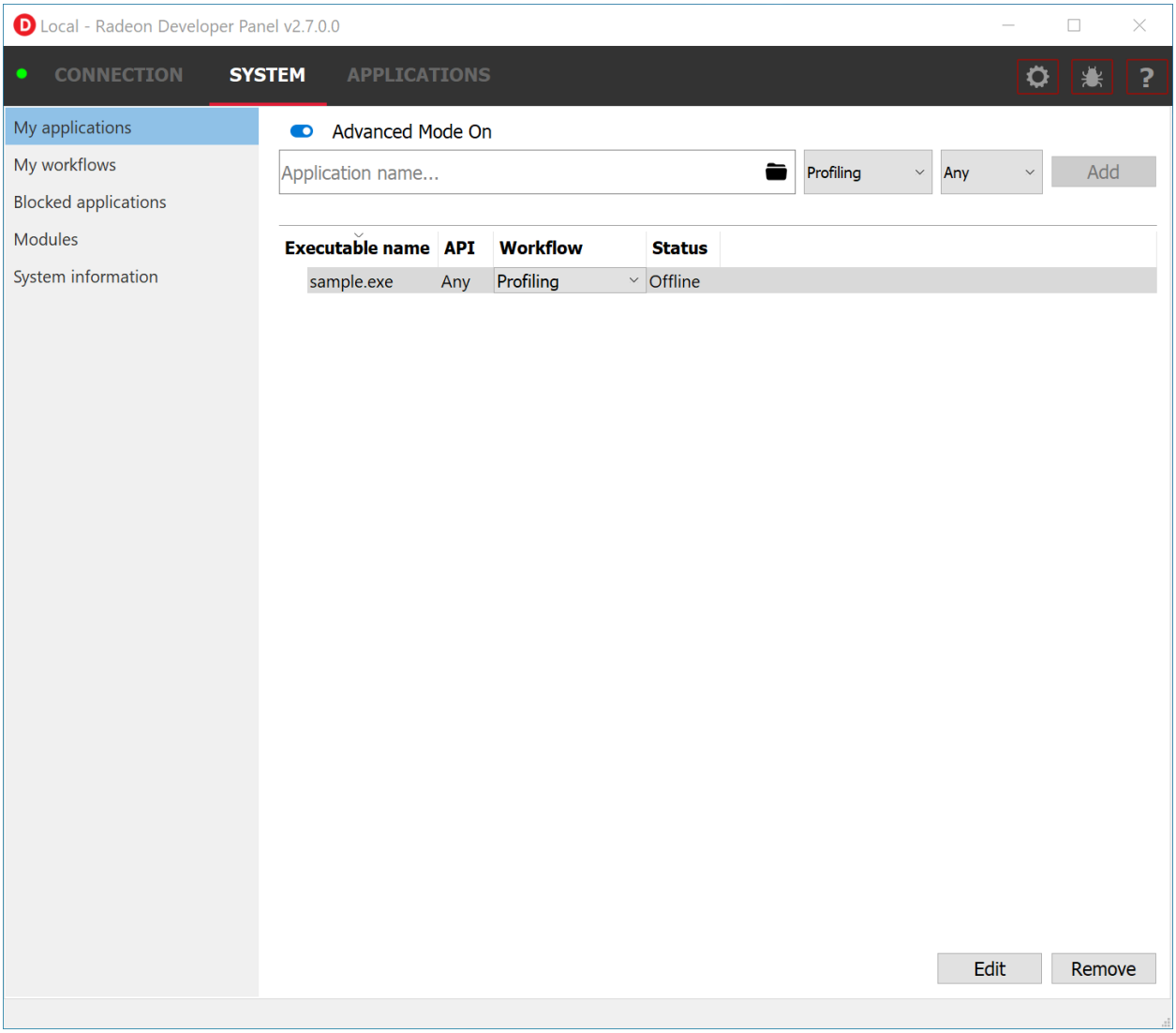
2) Start the **RadeonDeveloperPanel(.exe)** on the local system. On the **CONNECTION** tab, enter the IP address of the **remote** system in the **Host name** and then click the "Connect" button.

Optionally a nickname for the connection can be provided. This name will show in parentheses in the Connection dropdown.

CHAPTER 5

System

After a connection is made to the service, the panel will switch to the **System** tab.



The system tab contains various panels for configuration:

- *My applications* - List of applications enabled for driver connection
- *My workflows* - List of workflows defining pre-launch configuration settings
- *Blocked applications*- List of applications blocked from driver connection
- Modules - List of modules and their version numbers for the current connection
- *System information* - Lists detailed hardware and system information for the active Radeon Developer Panel connection

5.1 My applications

The **My applications** pane in Radeon Developer Panel contains the list of applications the user will want to connect with to capture a profile, trace or scene from.

There are two modes of connection available.

- **Basic Mode** - Any application run (not already in blocked applications list) will connect

- **Advanced Mode** - Only applications with entries specified in the application list will connect

These modes can be toggled using the **Advanced Mode** toggle at the top of the pane.

Advanced Mode toggled off is **Basic Mode**

Application entries can be added to the list using **Advanced Mode** as follows:

- Enter the executable name into the input field, or click the file icon at the end of the input field to select the executable using a file browser.
- Specify the workflow to be used for pre-launch configuration by this entry using the **Workflow** dropdown.
- Specify the **API** type to check against for this application from the dropdown.

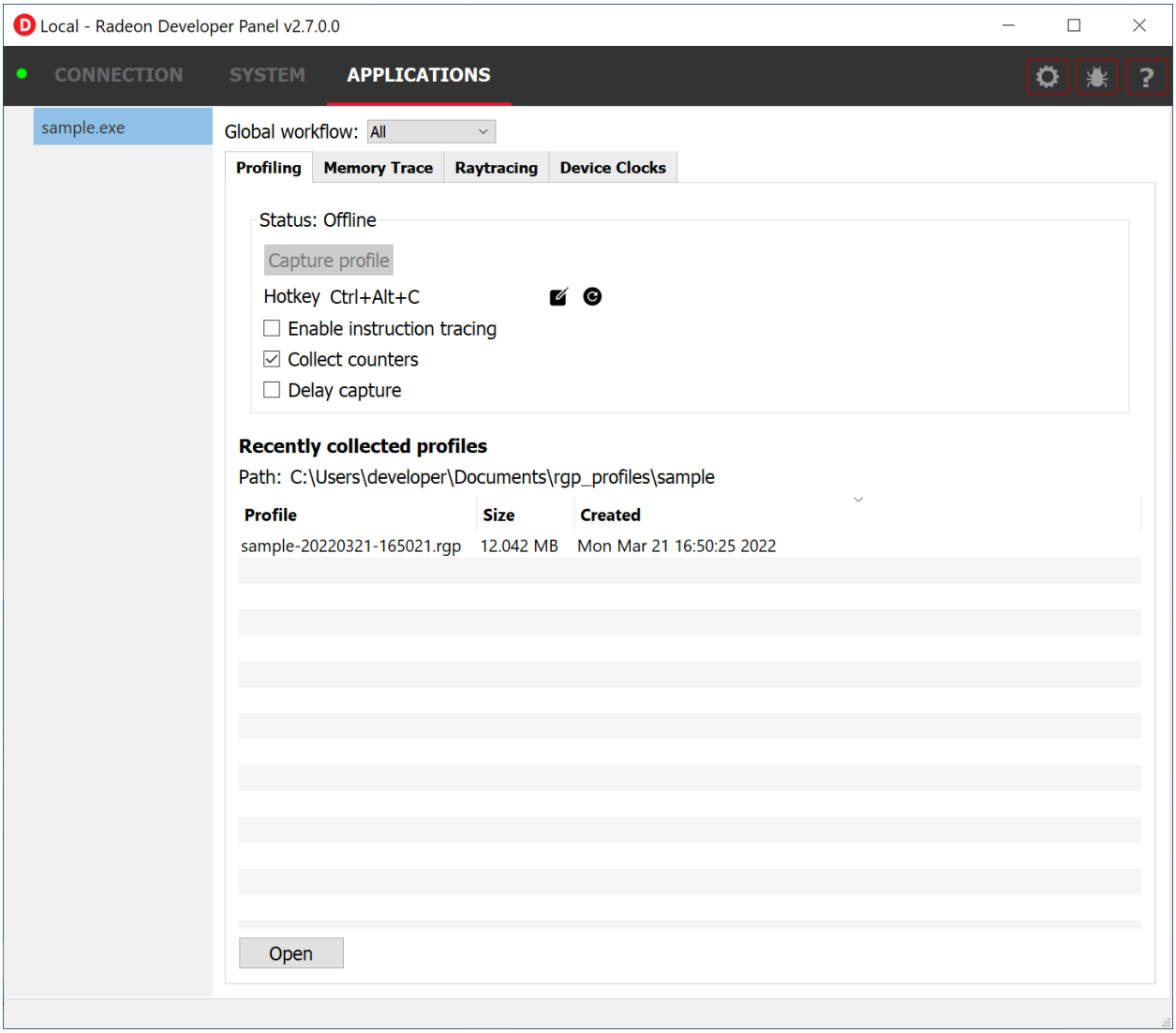
IMPORTANT Applications launched while using **Basic Mode** will automatically attempt a connection and (if an entry does not already exist in table) have an entry created in the table using the current workflow selected in the **Workflow** dropdown. If an entry existed for the application, then the global workflow chosen in the **Basic Mode** will override it. A proper warning message is shown in the status column in this view.

IMPORTANT The **API** specified works as a filter against the client application accepting the driver connection. If you are unsure of what **API** is being used or don't care use the default **Auto**

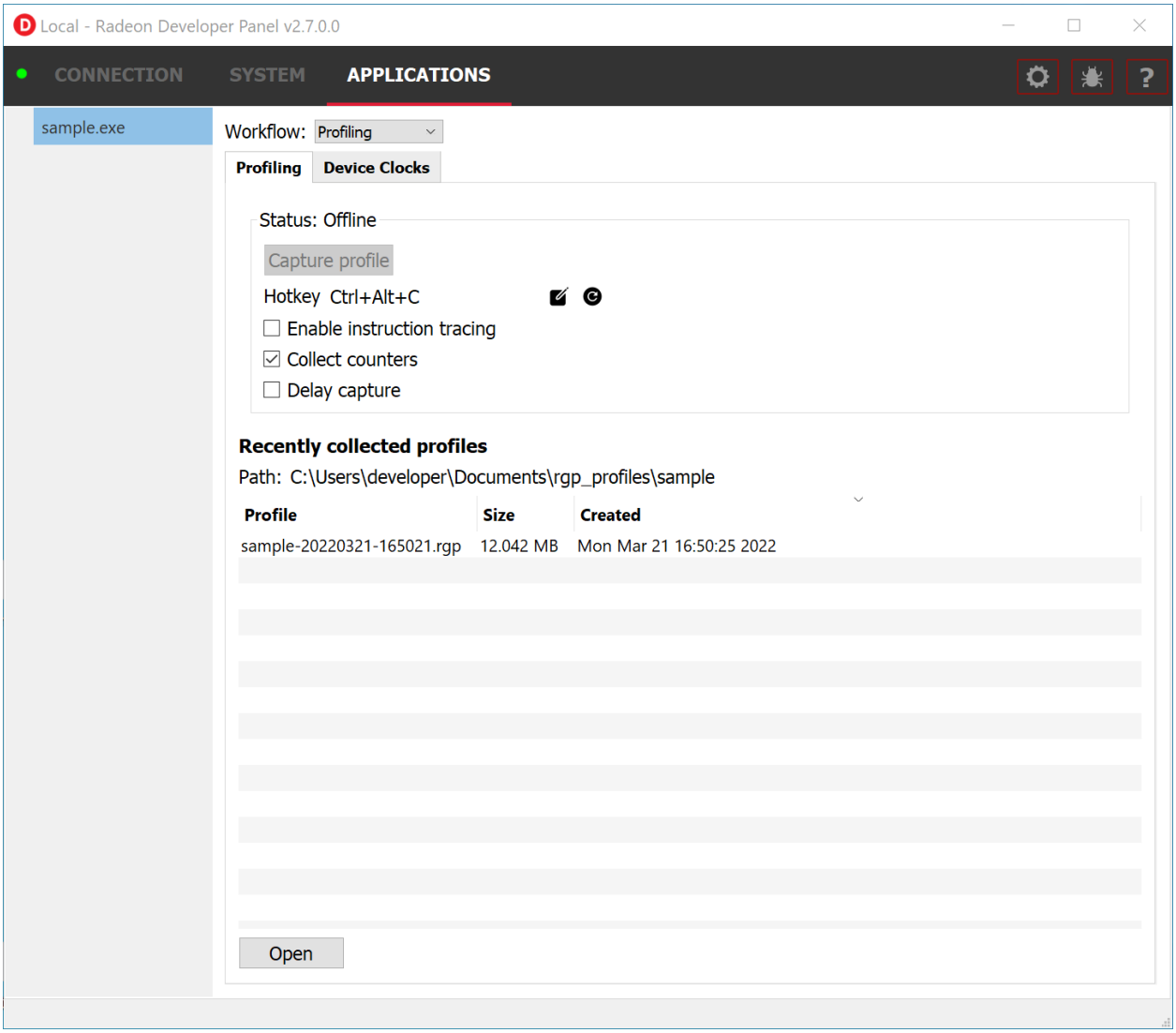
Once an application is added to the list, it can then be run on the system to start a driver connection.

When a connection to the client application has been established, the panel will then switch to the **Applications** tab.

When in **Basic Mode**, the global workflow can also be changed in the **Applications** tab. The dropdown on this tab is synced with the one in the **My applications** pane.



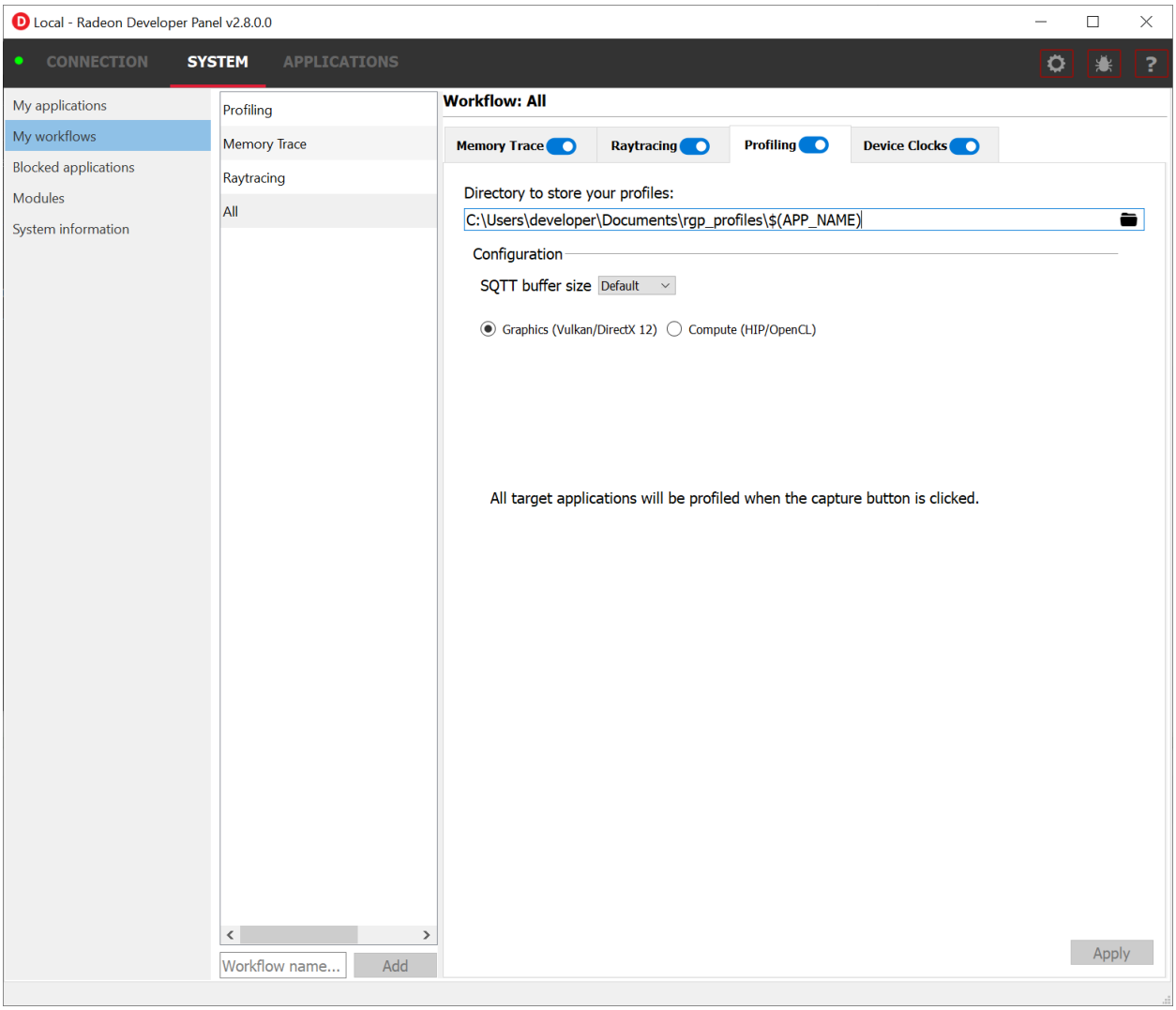
In **Advanced Mode**, the workflow for a specific application can also be changed by selecting it in the **Applications** tab and changing the Workflow dropdown. Any changes made to this dropdown will be reflected in the **My applications** pane.



5.2 My workflows

The **My workflows** pane in Radeon Developer Panel allows the user to specify a set of enabled features and pre-launch configuration options to be used when connecting an application.

Defining a workflow to contain these pre-launch settings such as the profile/trace/scene output path or capture mode allows for re-use of the settings across multiple applications.



Each workflow contains a list of features such as **Profiling**, **MemoryTrace**, **Raytracing**, or **DeviceClocks** which can be enabled or disabled

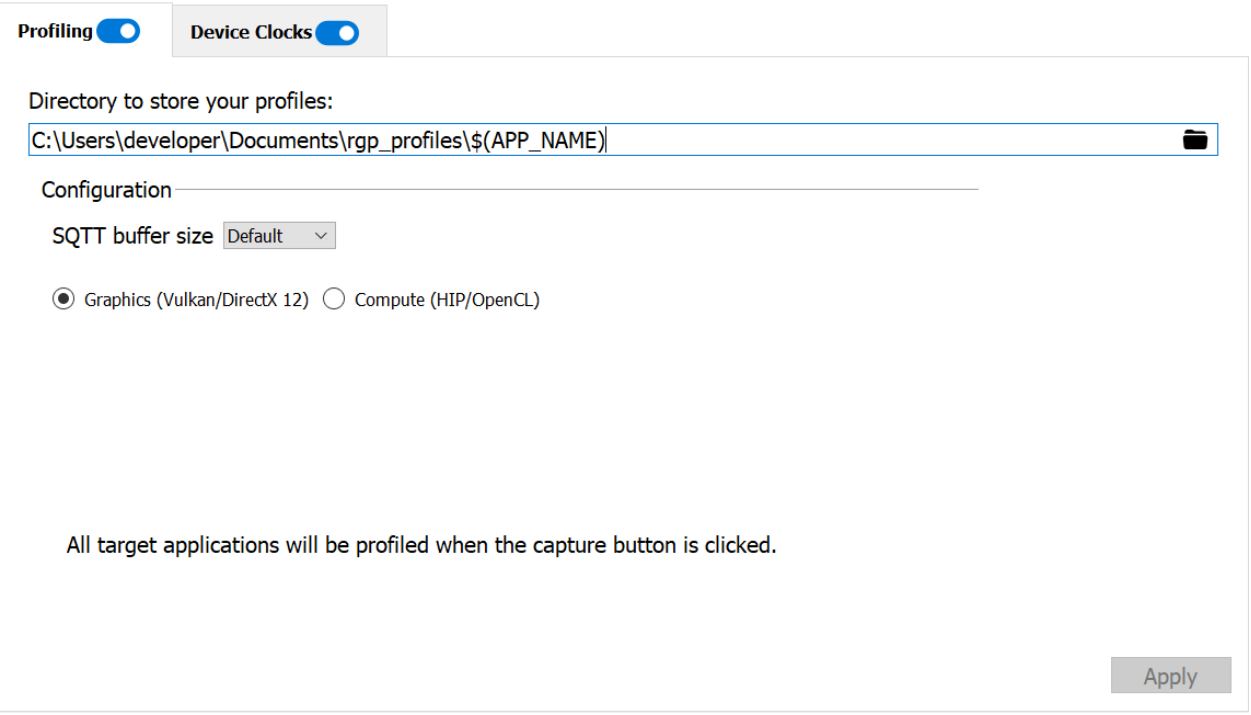
There are also configuration options available for these features:

Profiling Configuration

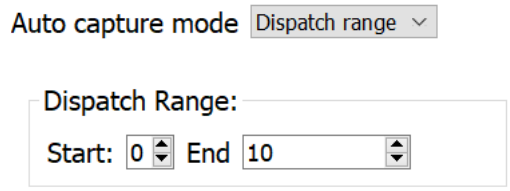
The following are the configurable options for profiling

- **Output Path:**
  - Defines the output path for saving captured profiles
  - Use the macro **$(APP_NAME)** to insert the connected application's name into path
- **SQTT Buffer Size:**
  - Defines the size of the buffer where SQTT data will be stored
  - If a profile has missing data, the SQTT buffer size can be increased to potentially remedy the issue
  - If an application experiences graphical corruption, decreasing the SQTT buffer size can potentially remedy the issue
- **Vulkan/DirectX12:**

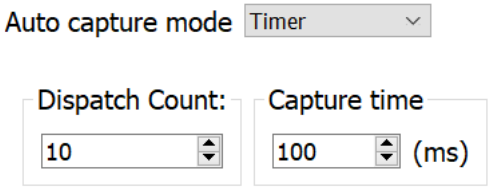
  - Displays information about the active trigger mode for profile capture



- **OpenCL:**
  - Displays configuration options for the trigger mode and dispatch range for profile capture
  - Enable auto capture checkbox can enable/disable automatic capture for OpenCL

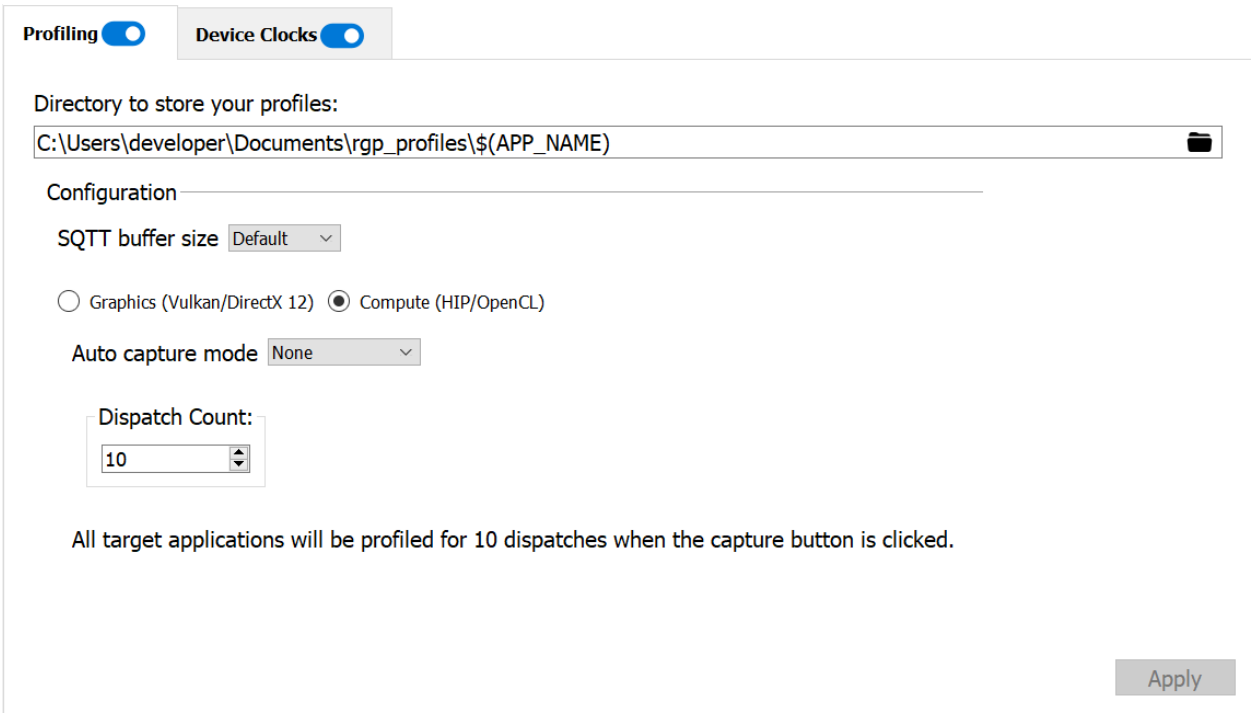


  - Dispatch Range allows for setting the start and stop dispatch indices to use during automatic profile capture



  - Dispatch count and capture time specifies the number of dispatches to capture after a specified elapsed time

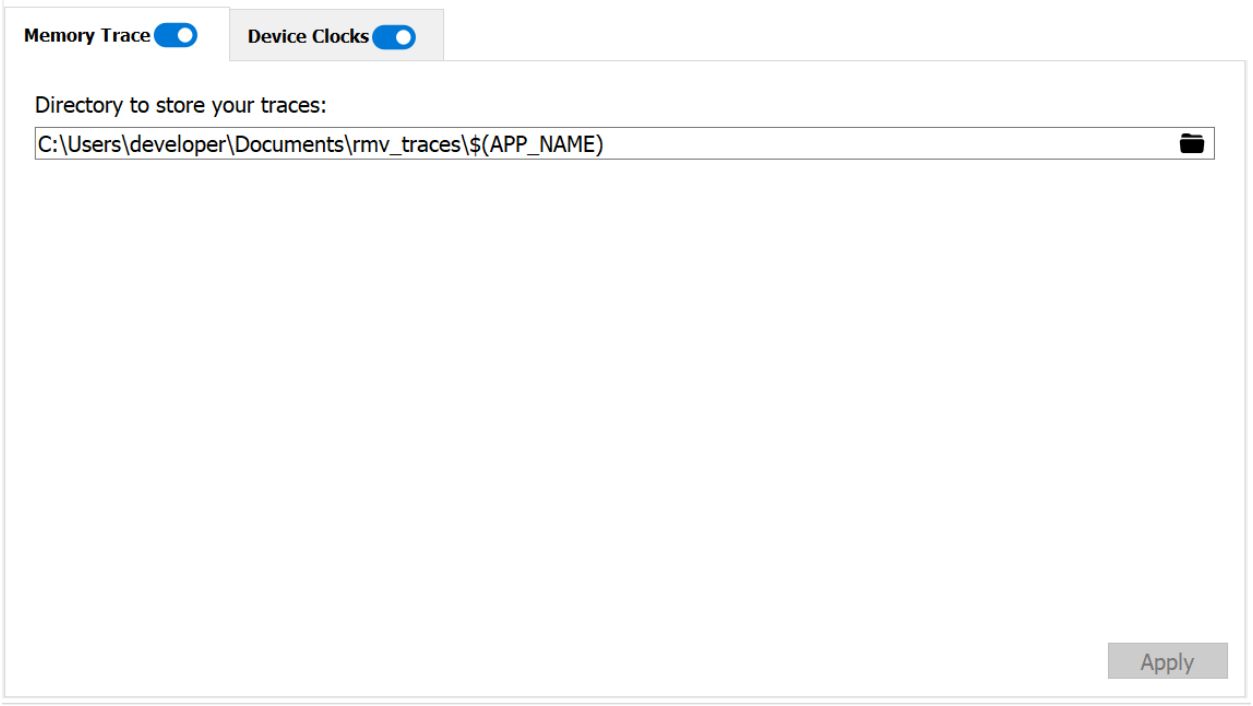
NOTE To reduce the chance of truncated profile data, OpenCL profiling is limited to 10000 dispatches



Memory Trace Configuration

The following are the configurable options for memory trace

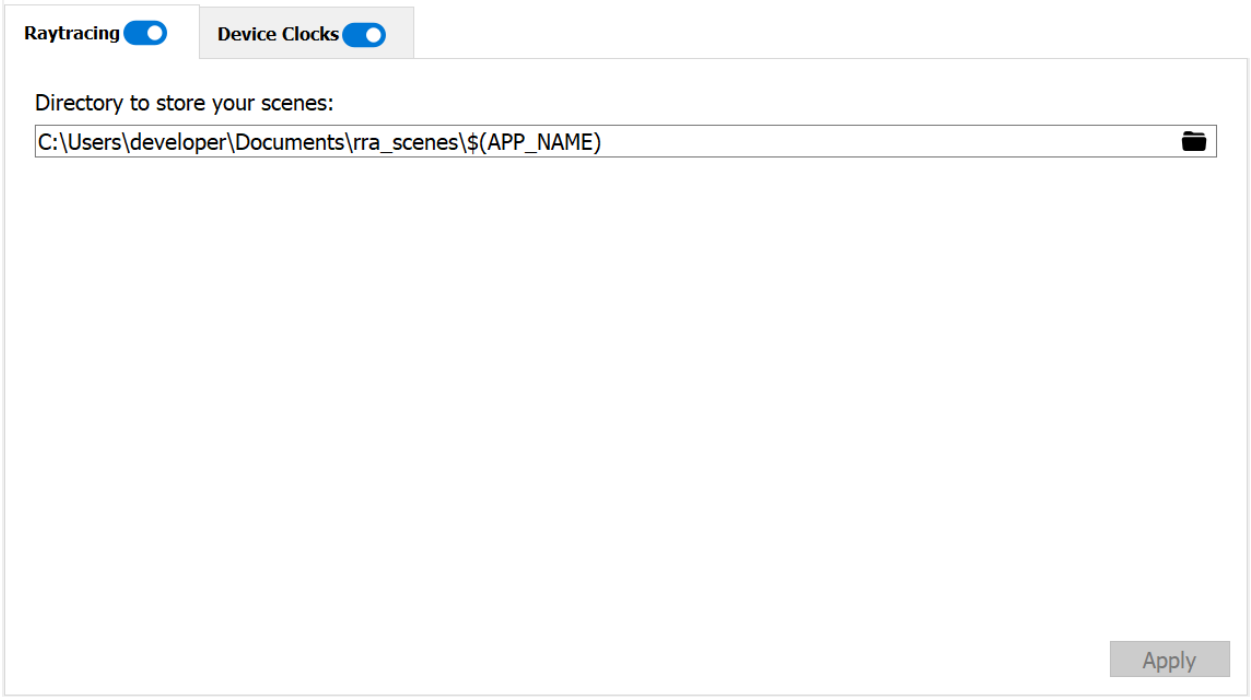
- **Output Path:**
  - Defines the output path for saving captured traces
  - Use the macro **$(APP_NAME)** to insert the connected application's name into path



Raytracing Trace Configuration

The following are the configurable options for raytracing

- **Output Path:**
  - Defines the output path for saving captured raytracing scenes
  - Use the macro **$(APP_NAME)** to insert the connected application's name into path



5.3 Blocked applications

Sometimes it is useful to completely exclude certain background applications from being recognized and displayed in the Radeon Developer Panel. For example, Windows 10 has applications that use DirectX 12 and when they are started can show up in the list of target applications. The **Profiling** feature also requires that only one application is started while using the feature so blocking applications, such as launchers that run before another application starts, can be useful.

The panel maintains a list of default applications that are blocked on either Windows or Linux. This list can be viewed from the **Blocked applications** subtab on the **System** tab which will appear once a local or remote connection has been established. Applications can be added or removed from the list by clicking one of the buttons below the list of process names. When editing or removing entries, first select the process name from the list then click the edit or remove button. The list can also be restored to the default set of blocked applications. Right clicking on a process name in the list will display context menu options to add, remove, or edit.

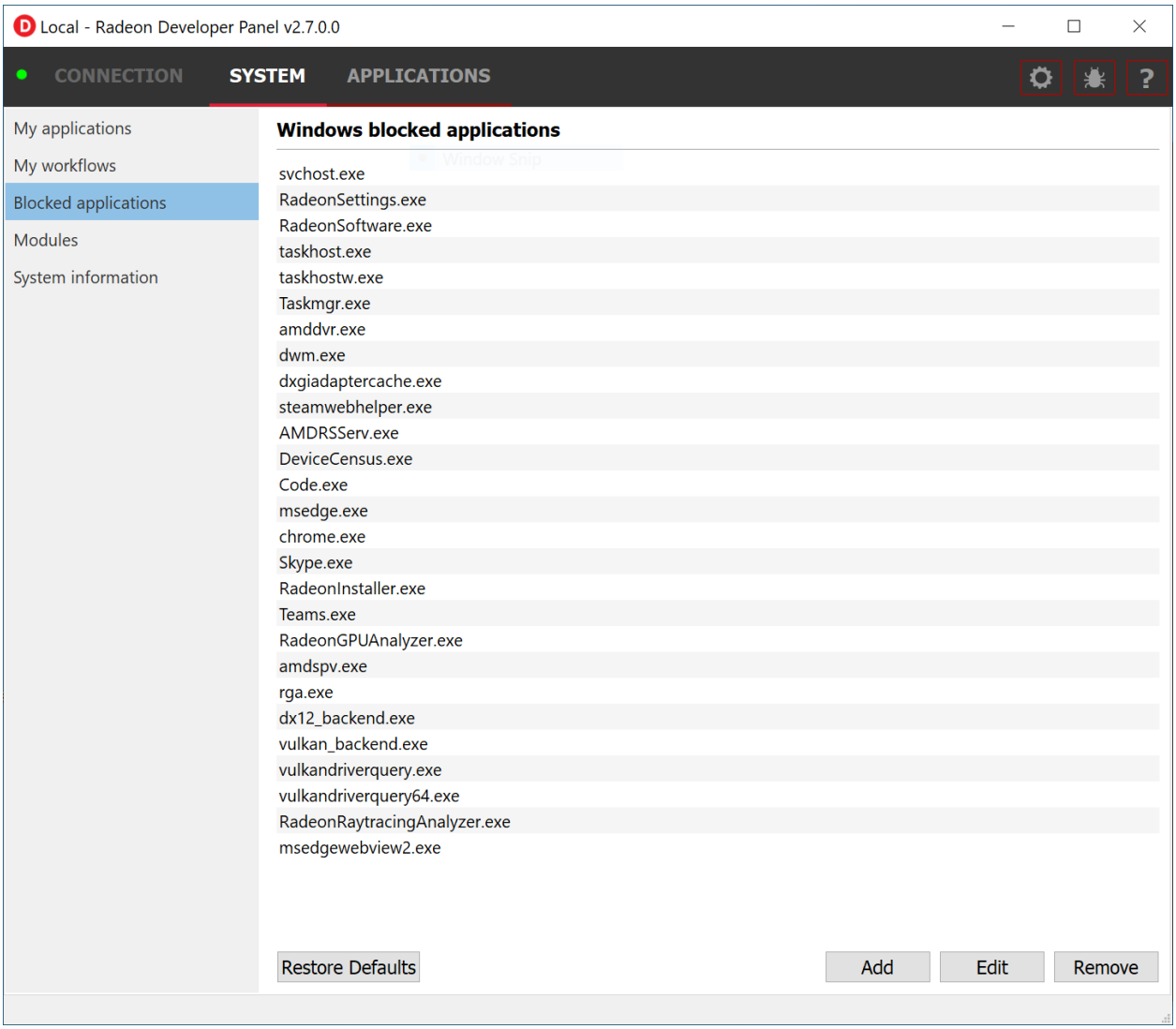
The blocked applications list supports wildcard matching. The following syntax is supported:

- ? : Matches exactly one of any character
- [. . .] : Matches one character in a set of characters
- * : Matches zero or more of any character

These can be escaped by using backslash.

Here are some examples of blocked application items that leverage the wildcard matching:

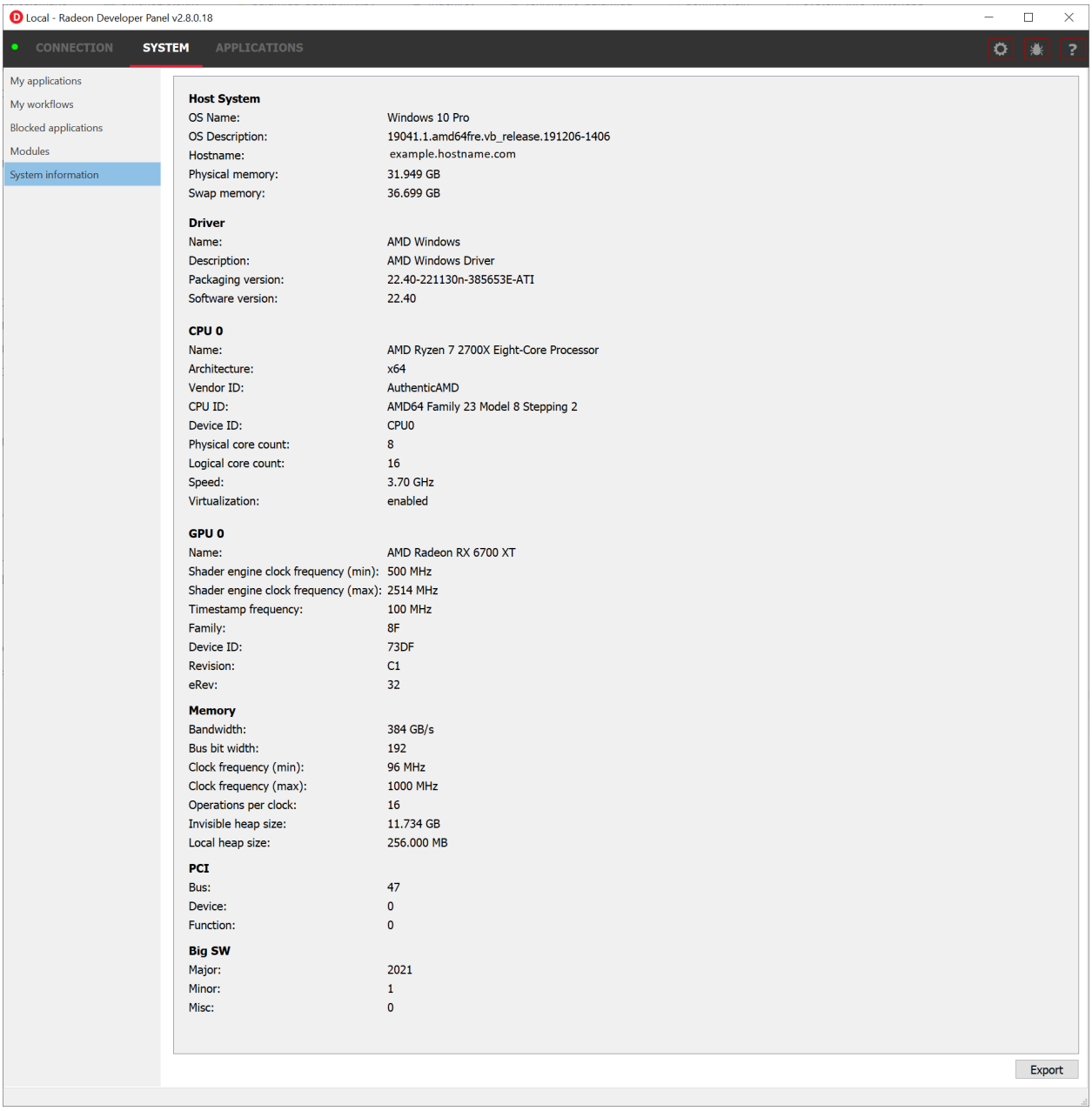
- [Gg]ears.exe : Blocks any application called gears.exe with either a lowercase or uppercase G
- gpu_info* : Blocks any applications who's name starts with gpu_info
- test?.exe : Blocks any application called test with a single character suffix – e.g. test1 or test6



5.4 System information

The system information pane lists detailed hardware and system information for the active Radeon Developer Panel connection.

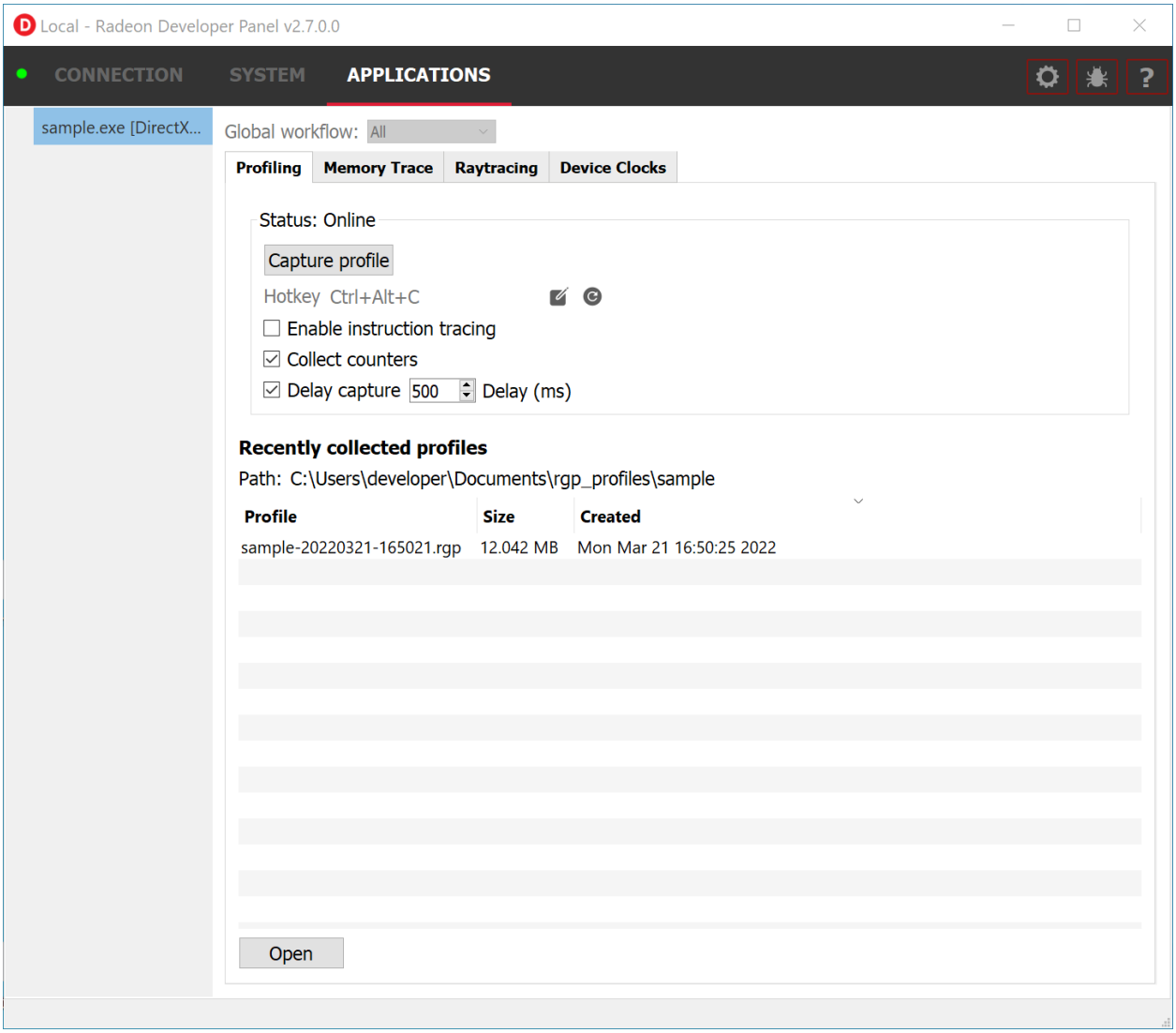
Pressing the Export button will open a dialog to choose a folder. Upon selecting a folder, the system information will be exported to that folder as a JSON file.


Local - Radeon Developer Panel v2.8.0.18
CONNECTION
SYSTEM
APPLICATIONS
My applications
My workflows
Blocked applications
Modules
System information
Host System
OS Name: Windows 10 Pro
OS Description: 19041.1.amd64fre.vb_release.191206-1406
Hostname: example.hostname.com
Physical memory: 31.949 GB
Swap memory: 36.699 GB
Driver
Name: AMD Windows
Description: AMD Windows Driver
Packaging version: 22.40-221130n-385653E-ATI
Software version: 22.40
CPU 0
Name: AMD Ryzen 7 2700X Eight-Core Processor
Architecture: x64
Vendor ID: AuthenticAMD
CPU ID: AMD64 Family 23 Model 8 Stepping 2
Device ID: CPU0
Physical core count: 8
Logical core count: 16
Speed: 3.70 GHz
Virtualization: enabled
GPU 0
Name: AMD Radeon RX 6700 XT
Shader engine clock frequency (min): 500 MHz
Shader engine clock frequency (max): 2514 MHz
Timestamp frequency: 100 MHz
Family: 8F
Device ID: 73DF
Revision: C1
eRev: 32
Memory
Bandwidth: 384 GB/s
Bus bit width: 192
Clock frequency (min): 96 MHz
Clock frequency (max): 1000 MHz
Operations per clock: 16
Invisible heap size: 11.734 GB
Local heap size: 256.000 MB
PCI
Bus: 47
Device: 0
Function: 0
Big SW
Major: 2021
Minor: 1
Misc: 0
Export

CHAPTER 6

How to profile your application

Upon running an application successfully the panel will have switched to the **Applications** tab shown here:



The profiling UI has the following elements:

- **Capture profile** - Captures a profile and writes to disk
- **Enable instruction tracing** - Enables capturing detailed instruction data
- **Collect counters** - Enables capturing GPU cache counter data. Systems with an AMD Radeon RX 6000 or AMD Radeon RX 7000 series GPU will also collect raytracing counter data.
- **Delay capture** - If this is enabled, pressing the capture profile button or triggering the hotkey will first wait the entered number of milliseconds before capturing a profile.
- **Recently collected profiles** - Displays any recently collected profiles found in the output directory

Capturing a profile can be achieved by the following:

- **Click the Capture profile button**

  Clicking the **Capture profile** button from the Profiling UI will capture a frame and write the results to disk.

- **Use the Ctrl-Alt-C hotkey**

Using Ctrl-Alt-C default hotkey on Windows or Linux® will capture a frame and write the results to disk.

This can be configured **before launching an application** by clicking the edit button to the right of the hotkey label and then entering a series of key presses.

Example output:

sample-20200908-092653.rgp

NOTE The profile output directory is specified as part of the associated **workflow** with this application entry in the **My applications** list

CHAPTER 7

Settings

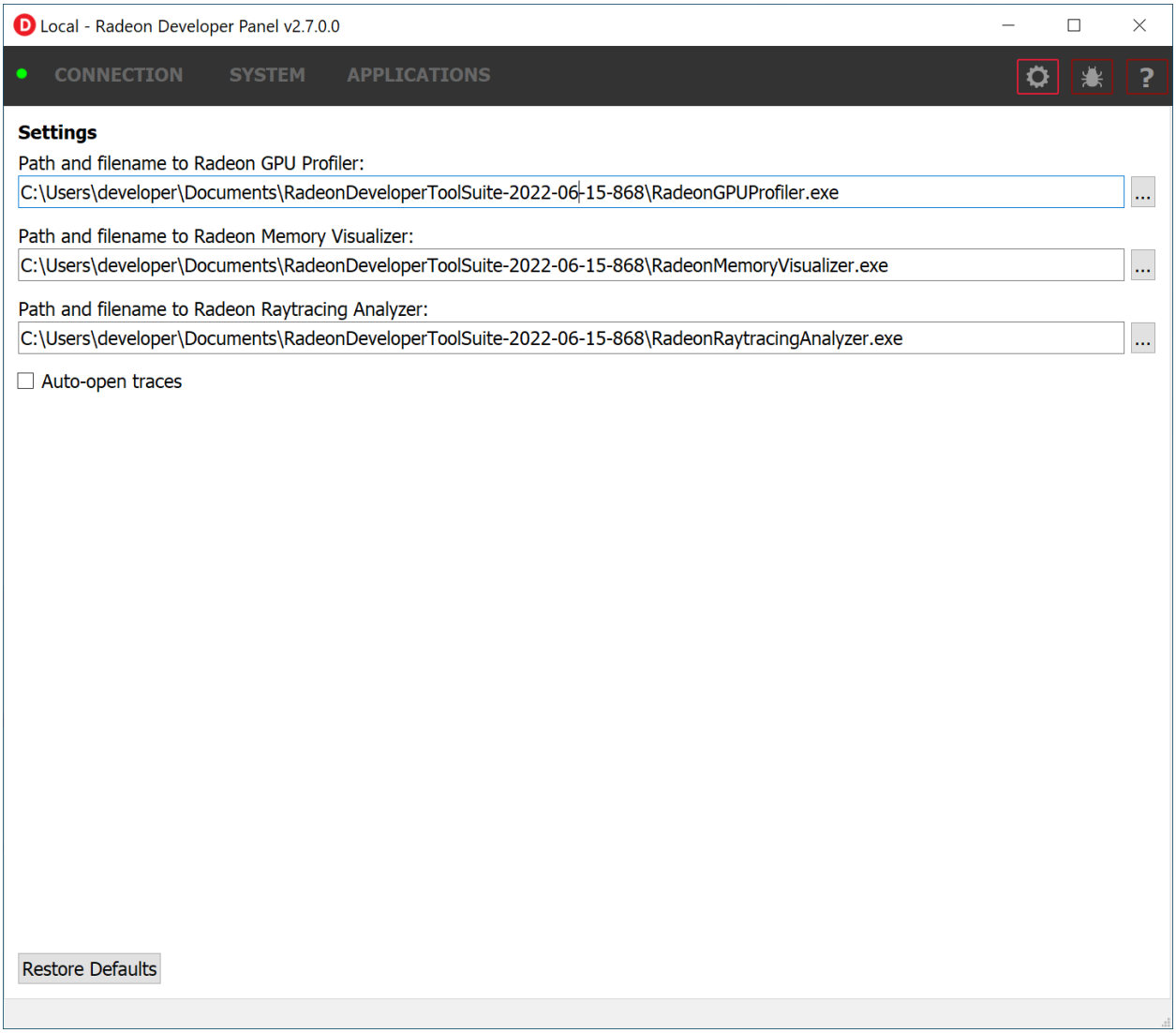
At any time the Radeon Developer Panel settings can be accessed by clicking the gear button in the upper right corner. This will open the settings pane.

After capturing a profile, trace or scene from an application, it is often desirable to open the output file using the associated tool such as **Radeon GPU Profiler**, **Radeon Memory Visualizer** or **Radeon Raytracing Analyzer**.

The settings pane allows for choosing the global path to the tool to be used by Radeon Developer Panel to open captured profiles, traces and scenes.

Additionally, the settings pane contains the Auto open traces toggle which will cause Radeon Developer Panel to open a captured profile, trace or scene with the correct tool as soon as it is captured.

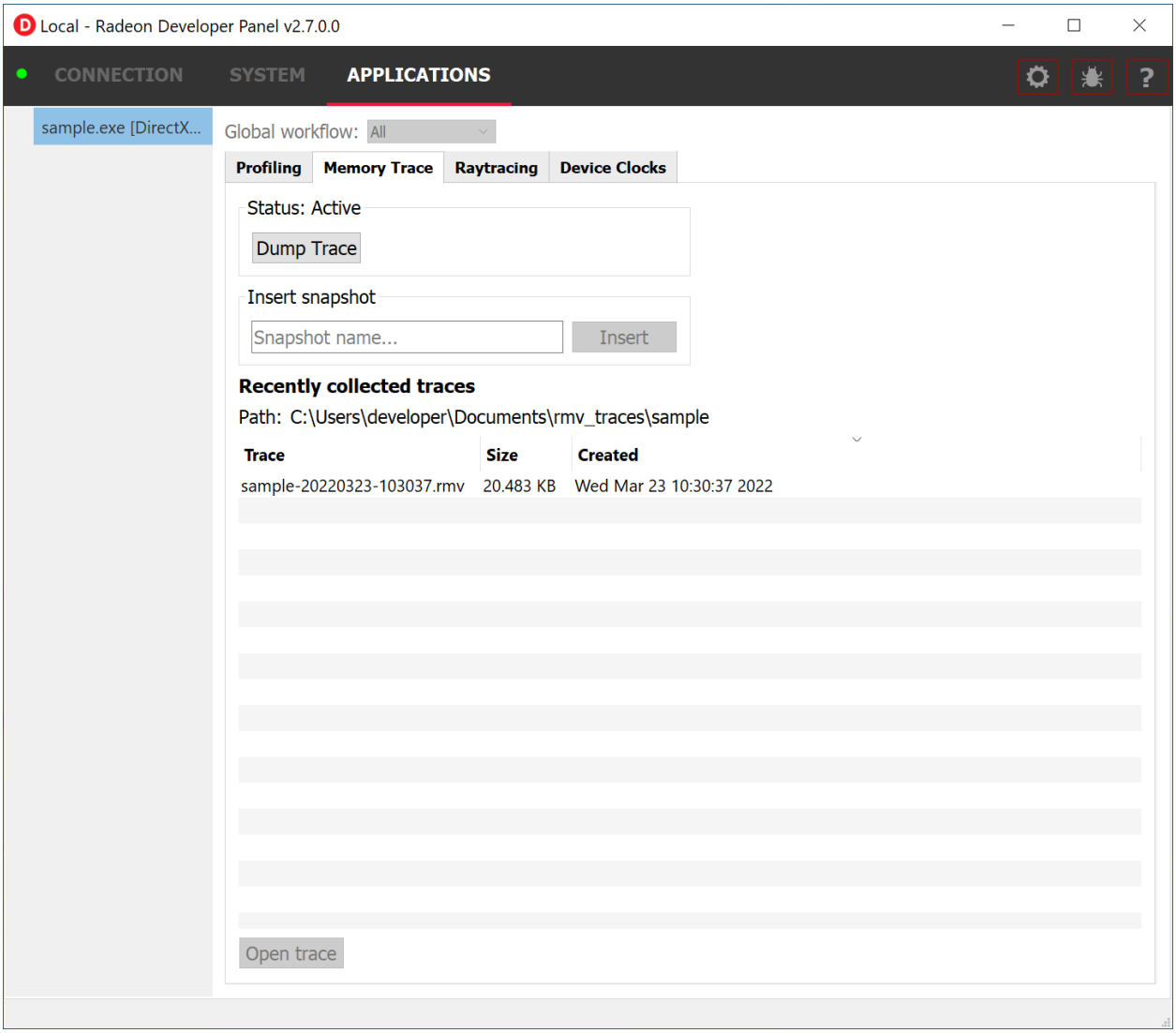
A **Restore Defaults** button allows for resetting the path and auto open settings to their default values. For the paths, this will reset them to the panel's executable path directory.


Local - Radeon Developer Panel v2.7.0.0
CONNECTION
SYSTEM
APPLICATIONS
Settings
Path and filename to Radeon GPU Profiler:
C:\Users\developer\Documents\RadeonDeveloperToolSuite-2022-06-15-868\RadeonGPUProfiler.exe
...
Path and filename to Radeon Memory Visualizer:
C:\Users\developer\Documents\RadeonDeveloperToolSuite-2022-06-15-868\RadeonMemoryVisualizer.exe
...
Path and filename to Radeon Raytracing Analyzer:
C:\Users\developer\Documents\RadeonDeveloperToolSuite-2022-06-15-868\RadeonRaytracingAnalyzer.exe
...
Auto-open traces
Restore Defaults

CHAPTER 8

How to memory trace your application

Upon running an application successfully the panel will have switched to the **Applications** tab shown here:



NOTE Memory tracing will have been implicitly started when the application was launched.

The memory trace UI has the following elements:

- **Dump trace** – stops memory tracing and writes results to disk
- **Insert snapshot** - insert user specified identifier to define snapshot in trace. A snapshot captures a moment in time in much the same way as a photograph. For example, to spot memory leaks, 2 snapshots can be added; one just before a game level is started after the menu screens and another snapshot when the game level finishes once the user is back in the game menus. Theoretically, the game should be in the same state in both cases (in the menus before and after a game level).
- **Recently collected traces** – displays any recently collected traces in output directory

Writing out the memory trace to file can be achieved by one of the following:

- **Close the running application**

  When the client application terminates, the memory tracing will stop and the results will be written to disk.

- **Click the Dump trace button**

Clicking the **Dump trace** button from the Memory Trace UI will stop memory tracing and write the results to disk.

Using either of the above methods to complete memory tracing will result in a **Radeon Memory Visualizer** trace file being written to disk.

Example output:

sample_20200316-143712.rmv

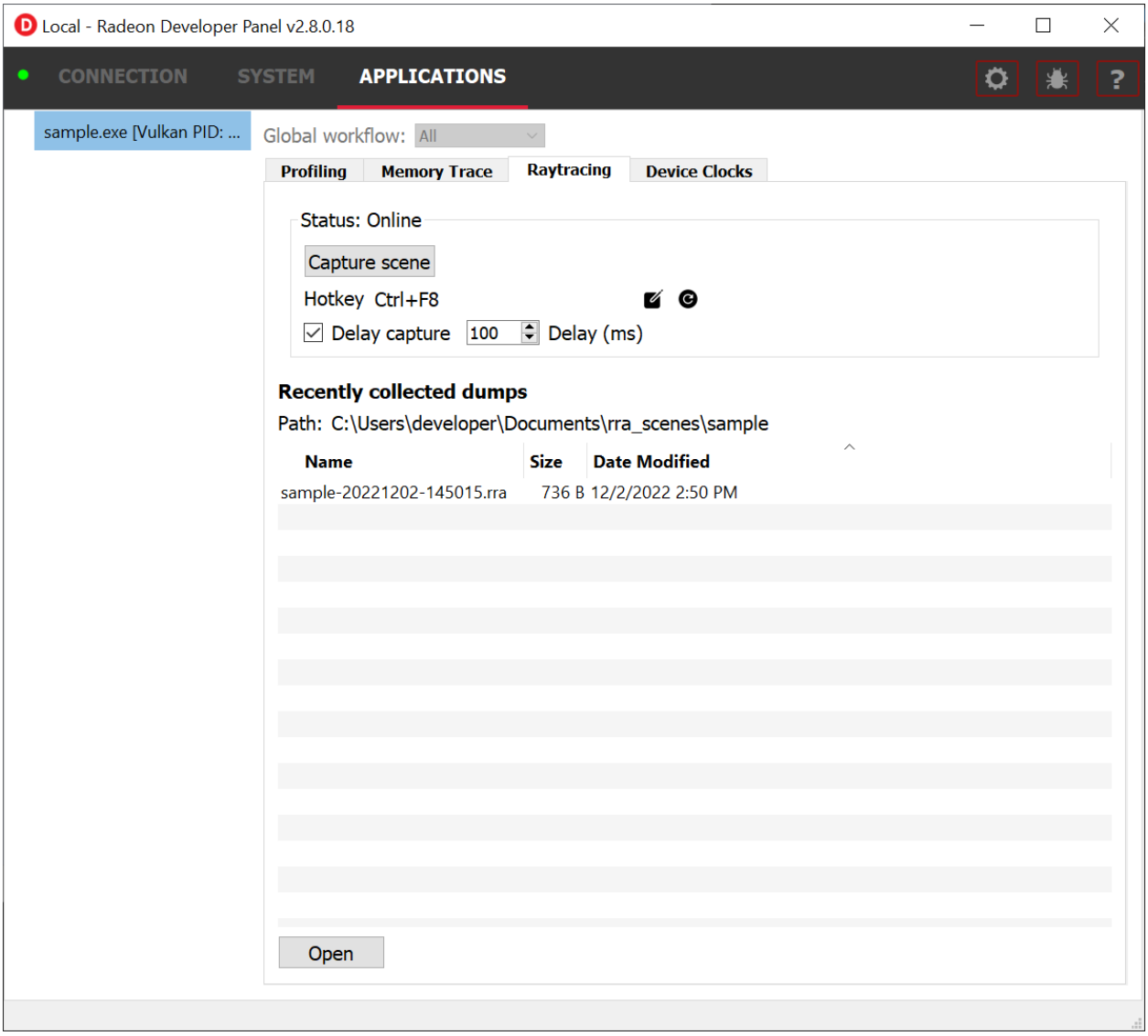
NOTE The trace output directory is specified as part of the associated **workflow** with this application entry in the **My applications** list

IMPORTANT: Once a memory trace has finished either through closing the application or through clicking the **Dump trace** button. The application **MUST** be closed and re-launched to start a new memory trace.

CHAPTER 9

How to capture a raytracing scene from your application

Upon running an application successfully the panel will have switched to the **Applications** tab shown here:



The raytracing UI has the following elements:

- **Capture scene** - Captures a scene and writes to disk
- **Delay capture** - If this is enabled, pressing the capture scene button or triggering the hotkey will first wait the entered number of milliseconds before capturing.
- **Recently collected scenes** - Displays any recently collected scenes found in the output directory

Capturing a scene can be achieved by the following:

- **Click the Capture scene button**

  Clicking the **Capture scene** button from the Raytracing UI will capture a raytracing scene and write the results to disk.

- **Use the Ctrl-F8 hotkey**

  Using Ctrl-F8 default hotkey on Windows or Linux® will capture a raytracing scene and write the results to disk.

  This can be configured **before launching an application** by clicking the edit button to the right of

the hotkey label and then entering a series of key presses.

Example output:

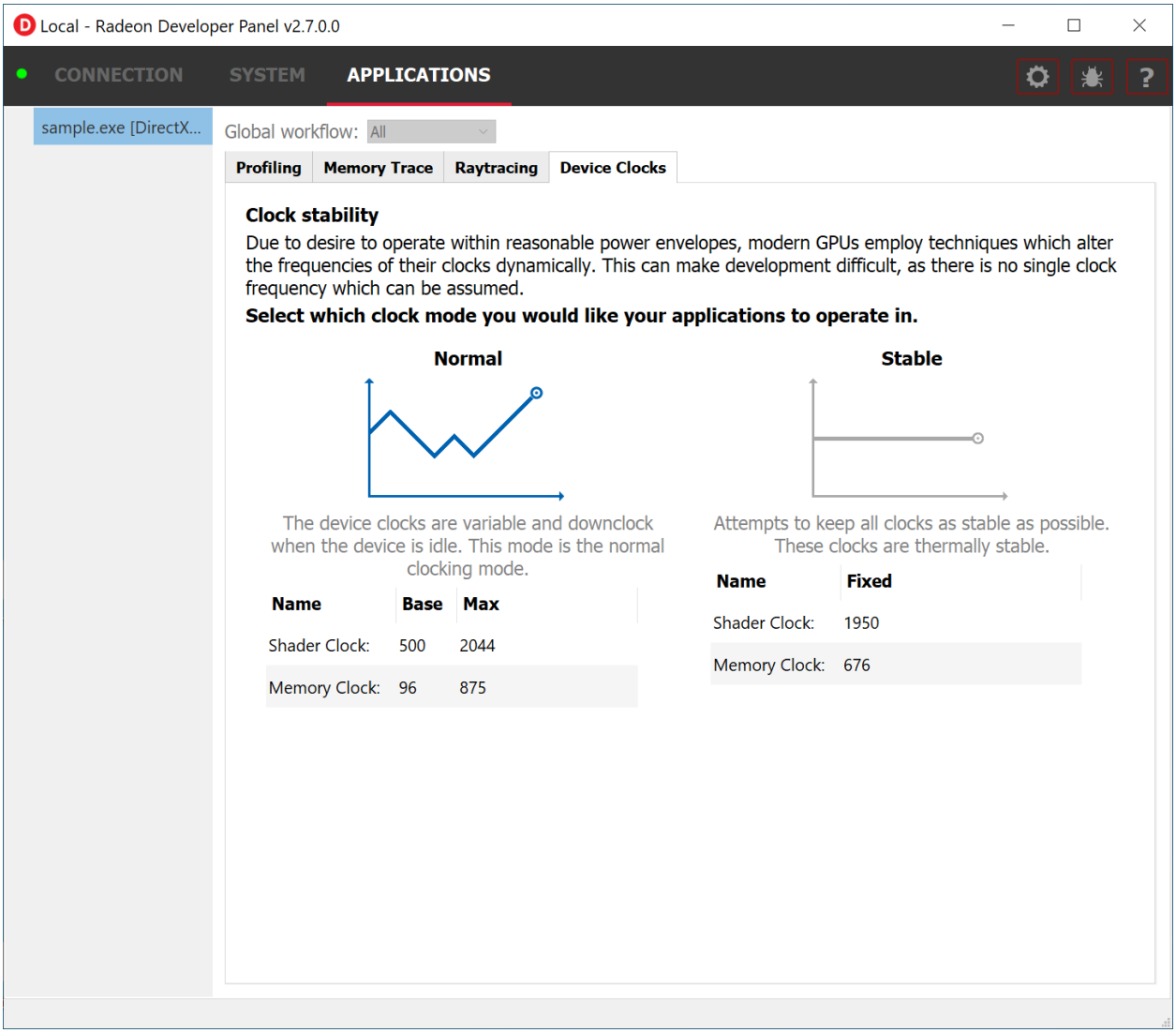
sample-20220705-104021.rra

NOTE The scene output directory is specified as part of the associated **workflow** with this application entry in the **My applications** list

CHAPTER 10

Using the Clock settings

The Radeon Developer Panel (RDP) allows the developer to select from a number of clock modes.



Normal clock mode will run the GPU as it would normally run your application. To ensure that the GPU runs within its designed power and temperature envelopes, it dynamically adjusts the internal clock frequency. This means that profiles taken of the same application may differ significantly, making side-by-side comparisons impossible.

Stable clock mode will run the GPU at a lower, fixed clock rate. Even though the application may run slower than normal, it will be much easier to compare profiles of the same application.

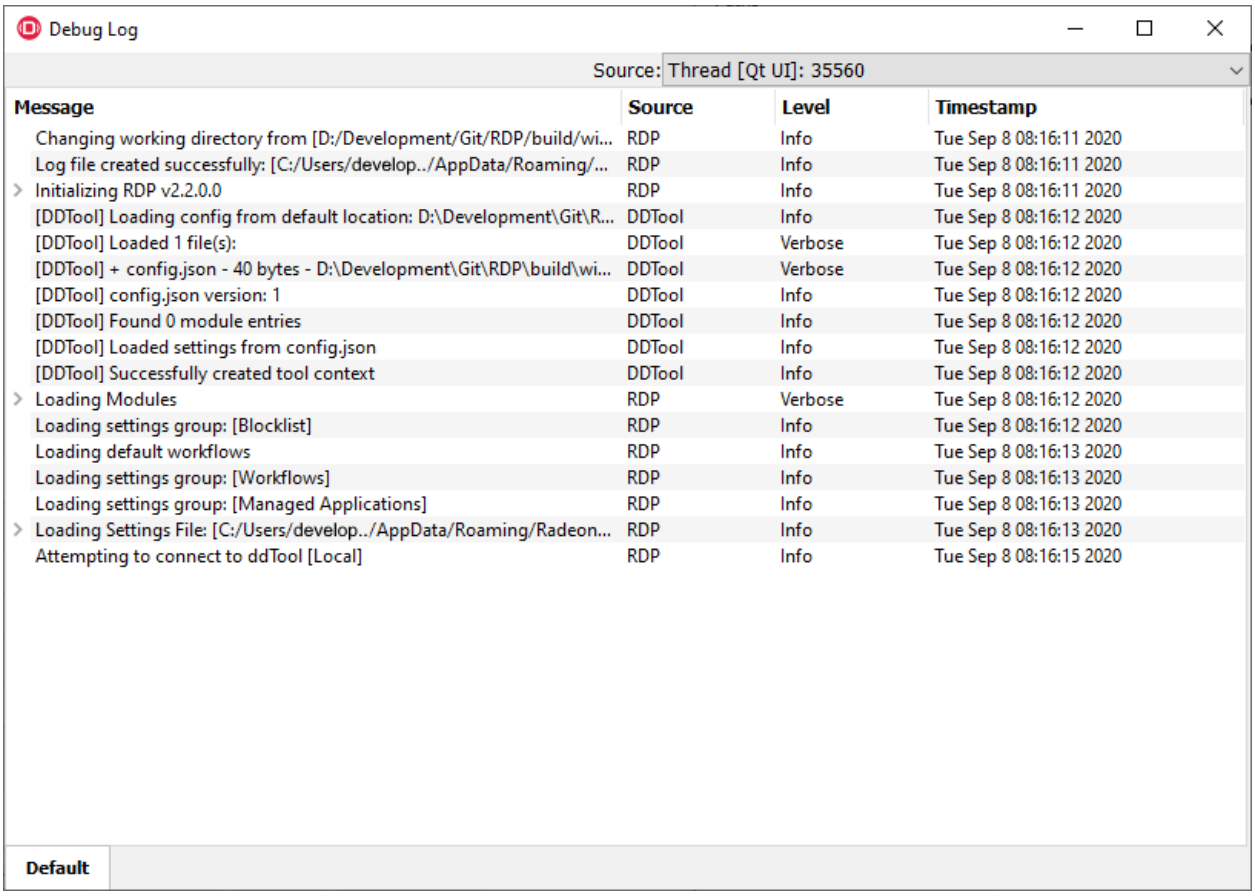
When capturing a profile, the clock settings here are not used since the driver forces a profile to take place using peak clocks.

NOTE A running, connected application is required in order to change the GPU clock modes

CHAPTER 11

Connection Log

Use the keyboard shortcut Ctrl-L to bring up the connection log. Additional information about the connection and any errors encountered by Radeon Developer Panel and the Radeon Developer Service are displayed here. Connection log messages are logged by thread and the log view only displays one thread's log messages at a time. Log messages from other threads can be viewed using the source dropdown. Below is an example of typical output from a session that captured a profile.



This log is also saved in a log file located at:
“C:\Users\your_name\AppData\Roaming\RadeonDeveloperPanel\log.txt”

On Linux, this log is located at:
“~/.local/share/RadeonDeveloperPanel/log.txt”

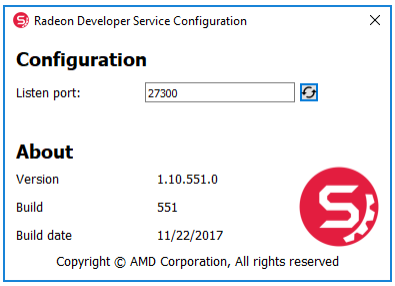
CHAPTER 12

The Radeon Developer Service

Two versions of the Radeon developer service are provided, one with a configuration UI and system tray icon, and one designed for use with headless GPU system where no UI can be supported.

12.1 Radeon Developer Service for desktop developer system

RadeonDeveloperService(.exe) – Can be used for general use where the system has a monitor and UI (e.g. desktop development machines). The Radeon Developer Service includes a configuration window containing basic service configuration settings and software info. **Double click the Radeon Developer Service system tray icon** to open the configuration window, or right-click on the system tray icon and select 'configure' from the context menu.



- **Listen port** – The port that the Radeon Developer Service uses to listen for incoming connections from a remote Radeon Developer Panel. **The default port is 27300**. Altering the port will disconnect all existing sessions. The circular arrows icon to the right of the Listen port field can be clicked to reset the port to the default value.
- **Version info** – Software version information for the Radeon Developer Service.

Double click the Radeon Developer Service system tray icon again or right-click on the system tray icon and select 'configure' from the context menu to close the configuration window.

Please note that when running both the Radeon Developer Panel and the Radeon Developer Service on the same system the communication between the two uses pipes, not sockets and ports, so setting the port has no effect.

12.2 Radeon Developer Service for headless GPU systems

RadeonDeveloperServiceCLI(.exe) – Command line version for use with headless GPU systems where no UI can be provided. NOTE: This version can also run on a system that has a monitor and UI.

The following command line options are available for RadeonDeveloperServiceCLI:

1) **– port <port number>** *Overrides the default listener port used by the service (27300 is the default).*

Please note that the service will need to be explicitly started before starting the Radeon Developer Panel. If the service isn't running, the Radeon Developer Panel will automatically start the UI version of the Radeon Developer Service, which may not be what is required.

CHAPTER 13

Bug Report

At any time, a bug report template can be generated by clicking the bug button in the upper right corner. This will copy a template to your clipboard with relevant information such as the graphics cards and operating system of the connected system.

When reporting bugs, please use the generated template and fill in the description and proper steps to reproduce the issue marked by the **“(fill me in)”** sections.

CHAPTER 14

Known Issues

14.1 Cleanup After a RadeonDeveloperServiceCLI Crash

If the RadeonDeveloperServiceCLI executable crashes on Linux, shared memory may need to be cleaned up by running the remove_shared_memory.sh script located in the script folder of the RGP release kit. Run the script with elevated privileges using sudo. If this fails to work, try starting the panel with elevated privileges.

14.2 Windows Firewall Blocking Incoming Connections

1) **Deleting the settings file**. If problems arise with connection or application histories, these can be resolved by deleting the Radeon Developer Panel's settings file at: "C:\Users\your_name\AppData\Roaming\RadeonDeveloperPanel\settings.ini"

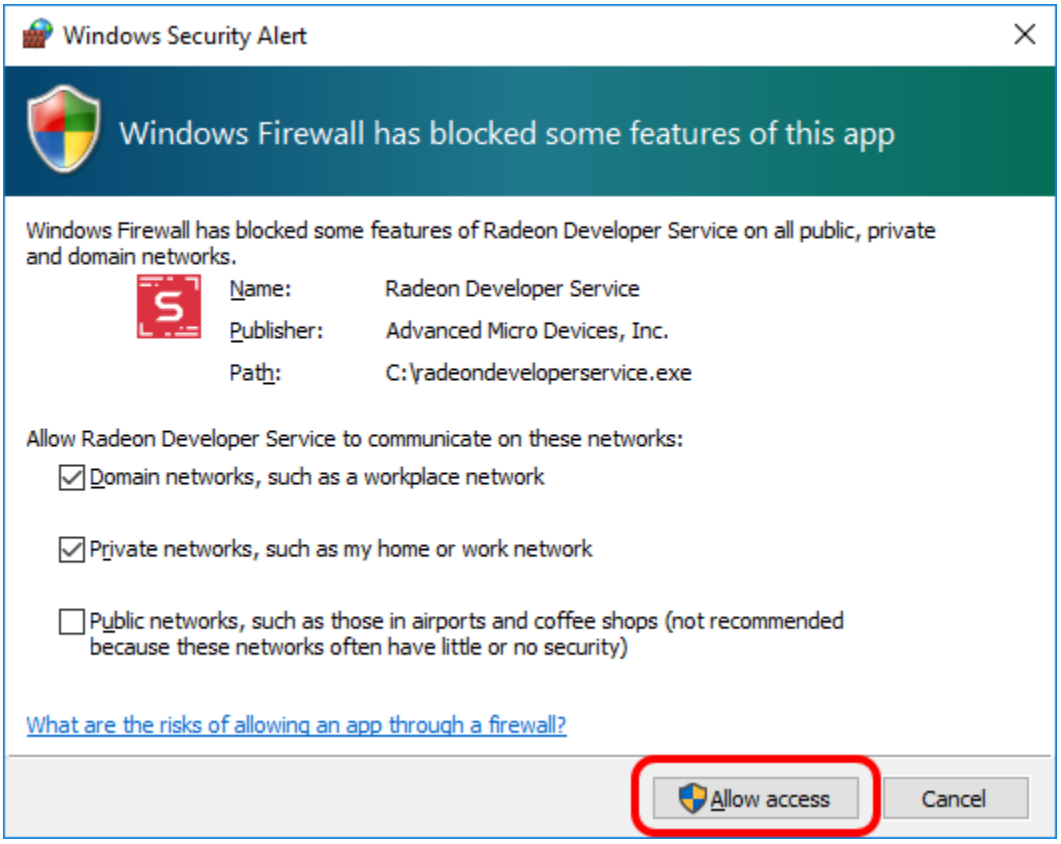
   on Windows. On Linux, the corresponding file is located at:

   "~/.local/share/RadeonDeveloperPanel/settings.ini"

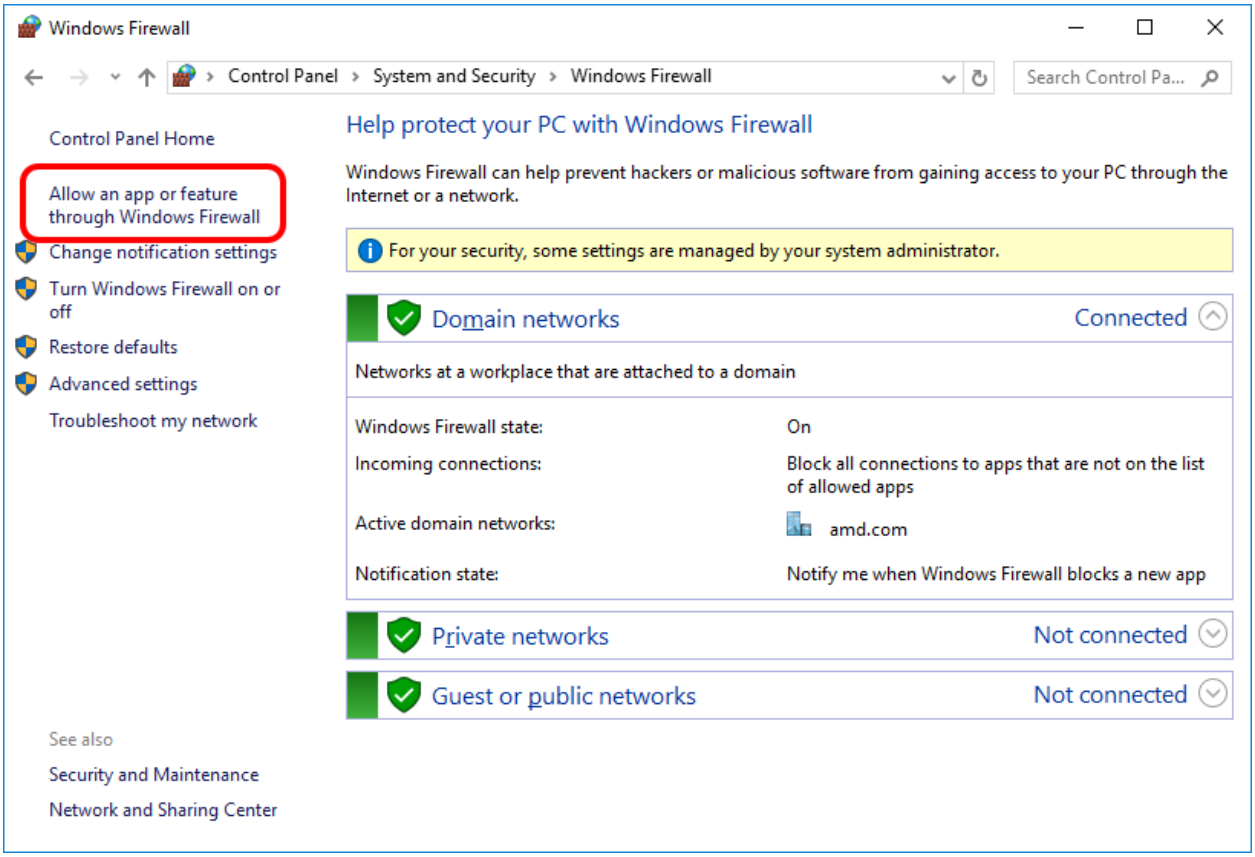
2) **"Connection Failure"** error message. This issue is sometimes seen when running the panel for the very first time. The panel tries to start the service automatically for local connections and this can fail. If you see this message try manually starting the "RadeonDeveloperService(.exe)" and connect again.

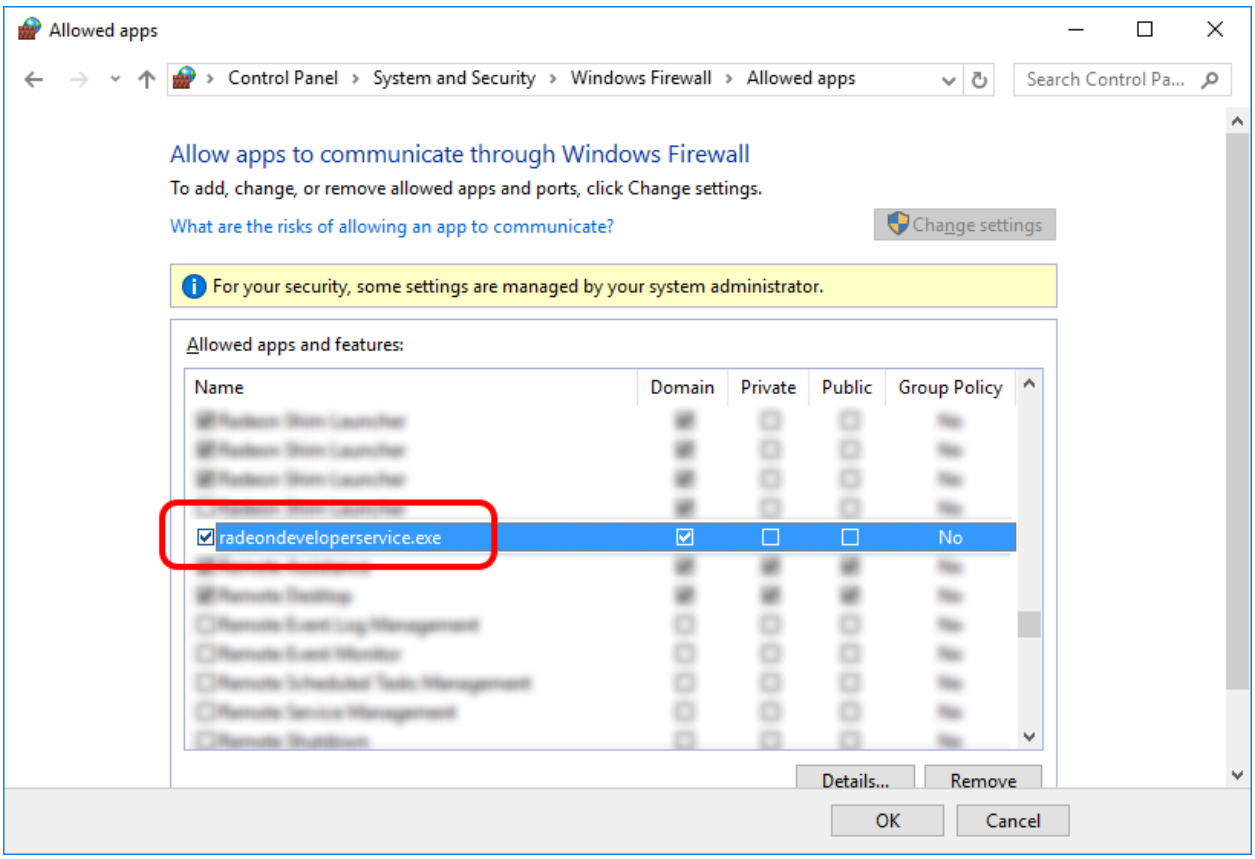
3) **Remote connection attempts timing out.** When running the Radeon Developer Service on Windows, the Windows Firewall may attempt to block incoming connection attempts from other machines. The best methods of ensuring that remote connections are established correctly are:

   a. Allow the RDS firewall exception to be created within the Windows Firewall when RDS is first started. Within the Windows Security Alert popup, enable the checkboxes that apply for your network configuration, and click "Allow access".



a. If "Cancel" was previously clicked in the above step during the first run, the exception for RDS can still be enabled by allowing it within the Windows Control Panel firewall settings. Navigate to the "Allow an app or feature" section, and ensure that the checkbox next to the RadeonDeveloperService(.exe) entry is checked:


Windows Firewall
Control Panel › System and Security › Windows Firewall
Search Control Pa...
Control Panel Home
Allow an app or feature through Windows Firewall
Change notification settings
Turn Windows Firewall on or off
Restore defaults
Advanced settings
Troubleshoot my network
Help protect your PC with Windows Firewall
Windows Firewall can help prevent hackers or malicious software from gaining access to your PC through the Internet or a network.
For your security, some settings are managed by your system administrator.
Domain networks
Connected
Networks at a workplace that are attached to a domain
Windows Firewall state:
On
Incoming connections:
Block all connections to apps that are not on the list of allowed apps
Active domain networks:
amd.com
Notification state:
Notify me when Windows Firewall blocks a new app
Private networks
Not connected
Guest or public networks
Not connected
See also
Security and Maintenance
Network and Sharing Center


Allowed apps
Control Panel › System and Security › Windows Firewall › Allowed apps
Search Control Pa...
Allow apps to communicate through Windows Firewall
To add, change, or remove allowed apps and ports, click Change settings.
What are the risks of allowing an app to communicate?
Change settings
For your security, some settings are managed by your system administrator.
Allowed apps and features:
Name
Domain
Private
Public
Group Policy
radeondeveloperservice.exe
No
Details...
Remove
OK
Cancel

a. Alternatively, disable the Windows Firewall entirely will also allow RDS to be connected to.

   NOTE The Windows firewall alert in no way indicates that the Radeon Developer tools are trying to communicate to an AMD server over the internet. The Radeon Developer tools do not attempt to connect to a remote AMD server of any description and do not send personal or system information over remote connections. The Radeon Developer Panel needs to communicate with the Radeon Developer Service, which may or may not be on the same machine, and a connection needs to be made between the two (normally via a socket).

14.3 Disabling Linux Firewall

If the remote machine is running Linux and the **"Connection Failure"** error message is displayed, the Linux firewall may need to be disabled. This is done by typing "**sudo ufw disable**" in a terminal. The firewall can be re-enabled after capturing by typing "**sudo ufw enable**".

14.4 Setting GPU clock modes on Linux

Adjusting the GPU clock mode on Linux is accomplished by writing to /sys/class/drm/card<n>/device/power_dpm_force_performance_level, where <n> is the index of the card in question. By default this file is only modifiable by root, so the application being profiled would have to be run as root in order for it to modify the clock mode. It is possible to modify the permissions for the file instead so that it can be written by unprivileged users. The Radeon GPU Profiler package includes the "**scripts/setup.sh**" script which when run as root will set the GPU clock mode. **Execute this script before running the Radeon Developer Service and target application,** and the GPU clock mode will be updated correctly at runtime.

NOTE This script needs to be run each time you reboot your machine; the file permissions do not survive system reboots.

14.5 Enabling support for RMV tracing on Linux

RMV tracing on Linux requires specific kernel tracing features to be enabled. The **scripts/setup.sh** script file when run as root will setup the necessary kernel tracing components to support RMV capture. Please run this script prior to launching **Radeon Developer Service** or **Radeon Developer Panel**.

14.6 Radeon Developer Panel connection issues on Linux

The Radeon Developer Panel may fail to start the Radeon Developer Service when the Connect button is clicked. If this occurs, manually start the Radeon Developer Service, select localhost from the Recent connections list and click the Connect button again.

14.7 Missing Timing Data for DirectX 12 Applications

To collect complete profile datasets for DirectX 12 applications, two additional actions must be performed:

1) The user account in Windows needs to be associated with the "Performance Log Users" group.

2) The following REG_DWORD registry key must be set: **HKEY_LOCAL_MACHINE\Software\AMD\RadeonTools\RgpEnableEt**

If these two privileges aren't configured properly, profiles collected under the user's account may not include all timing data for GPU Sync objects.

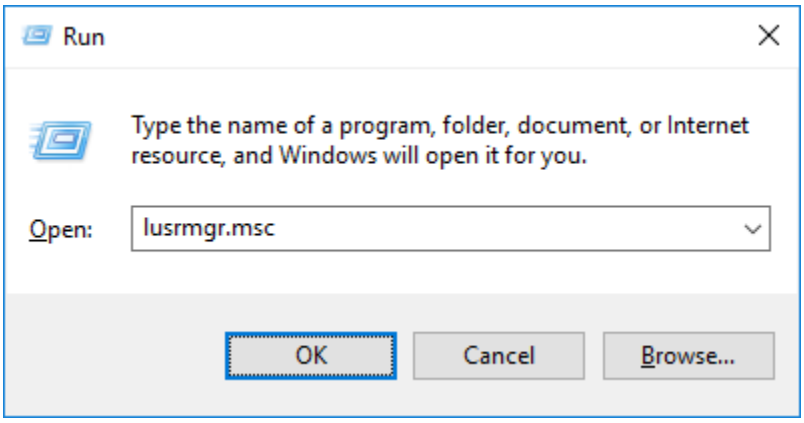
A batch file is provided to perform the above two actions (scripts\AddUserToGroup.bat). The batch file should be run as administrator (Right click on file and select "Run as Administrator"). The script's output is shown below:

```
C:\tools\RGP>AddUserToGroup.bat
Copyright (c) 2017-2021 Advanced Micro Devices, Inc. All rights reserved.

This script will add the current user to the "Performance Log Users" group.
This script will also add a required registry key.
Run this script with "--cleanup" to delete the registry key and remove the
current user from the group.

Domain:        AMD
Username:      developer
Computer Name: BDCL10-DEVELOPER

Running Script as Administrator.

**** Attempting to add user developer to "Performance Log Users" group ****

The command completed successfully.

Please reboot your system for these changes to take effect.

**** Attempting to add HKLM\Software\AMD\RadeonTools\RgpEnableEtw registry key ****

Registry updated successfully.

Exiting...

C:\tools\RGP>
```

The actions performed by the batch fie can be undone by running the batch file with a **–cleanup** command line switch.

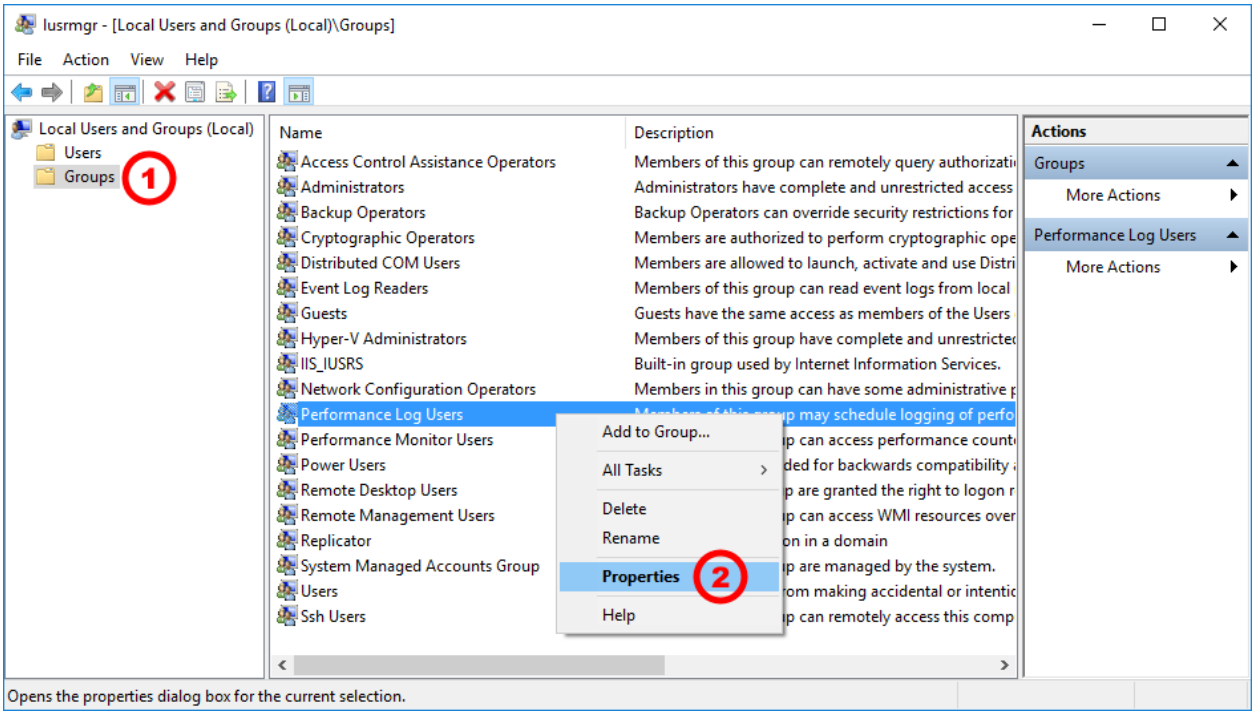
Alternatively, to manually add the active user to the proper group, follow these steps:

1) **Open the Run dialog** by using the Windows Start menu, or through the Windows + R shortcut.
    a. **Type** "**lusrmgr.msc**" into the Run window, and **click OK**.

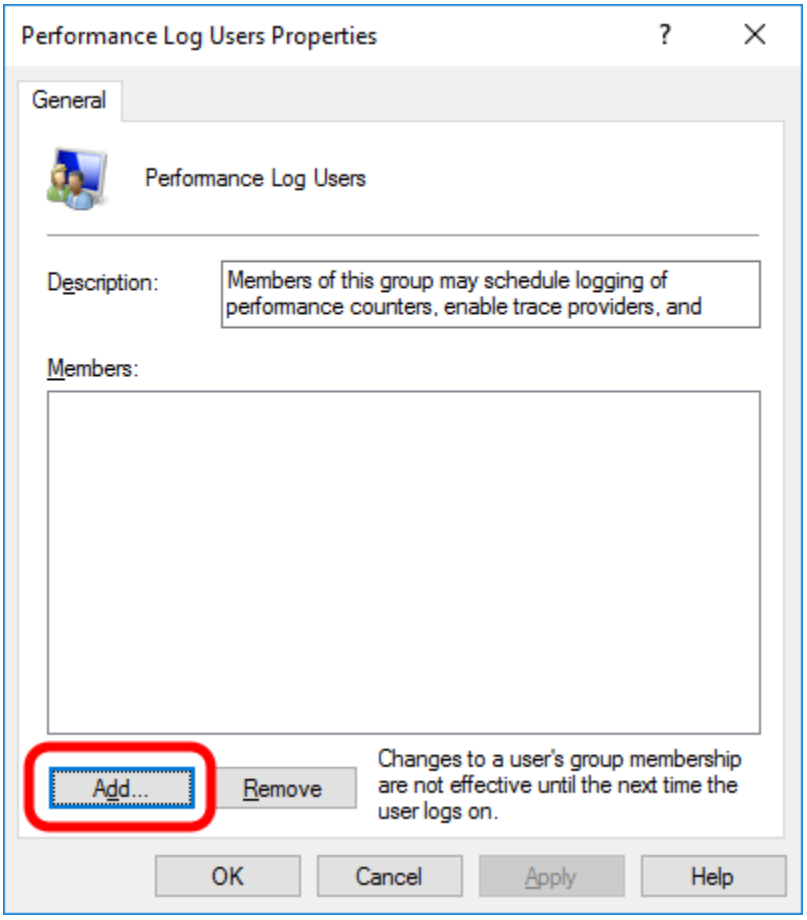


2) Within the "Local Users and Groups" configuration window that opens, **select the Groups node**.

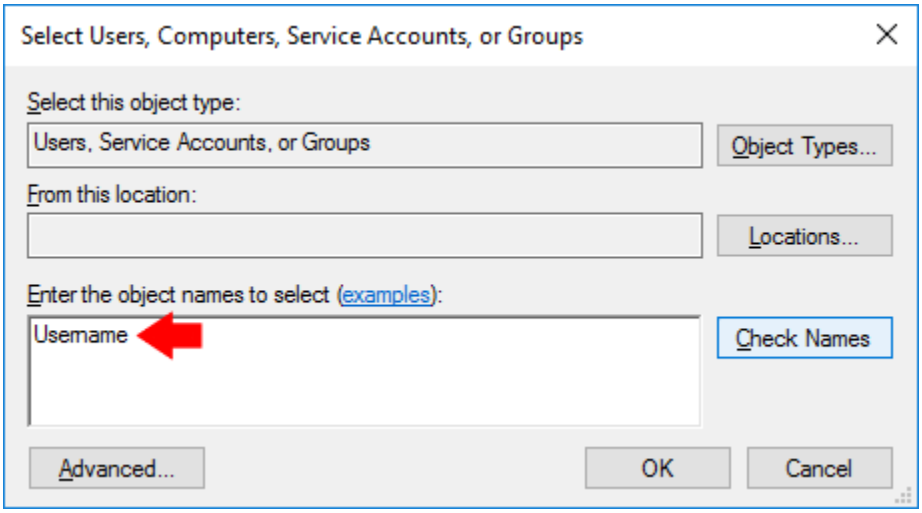
a. **Select the Performance Log Users entry. Right-click and select Properties.**



1) To add the active user to the group, **click the Add... button**. (If the active user appears within this list, the account is already configured properly.)



2) **Type the active user's account name** into the Select Users, Computers, Service Accounts, or Groups dialog, and **click OK**.



3) When the user has been added to the group, **restart the machine** and log back in. RDS should now be configured to collect full timing information for DirectX 12 applications.

14.8 Radeon Developer Service Port numbers

Please note that when running both the Radeon Developer Panel and the Radeon Developer Service on the same system the communication between the two uses pipes, not sockets and ports, so setting the port has no effect. In this scenario, it is possible to set the service to listen on a non-default port. Leave the panel on the default port, and connecting will work fine.

14.9 Problems caused by existing installation of RADV Linux Vulkan driver

Installations of Ubuntu 20.04 or newer may have the RADV open source Vulkan driver installed by default on the system. As a result, after an amdgpu-pro driver install, the default Vulkan ICD may be the RADV ICD.

In order to capture a profile, Vulkan applications must be using the amdgpu-pro Vulkan ICD. The default Vulkan ICD can be overridden by setting the following environment variable before launching a Vulkan application: VK_ICD_FILENAMES=/etc/vulkan/icd.d/amd_icd64.json

14.10 Problems caused by the presence of non-AMD GPUs and non-AMD CPUs with integrated graphics

The presence of non-AMD GPU's and CPU's on your system can cause the failure to generate a profile or apps to not run at all.

These problems typically occur with Vulkan apps in systems that have:

1) A non-AMD CPU with in integrated non-AMD GPU
2) A non-AMD discrete GPU

Vulkan applications, by default, use GPU 0 which usually maps to the integrated GPU, or in some cases, the non-AMD discrete GPU. In both cases Vulkan apps will either fail to run, or RGP profiling will not work (no RGP overlay will be present in these cases).

To avoid these issues:

1) Disable any non-AMD integrated GPU's in the device manager
2) Disable any non-AMD discrete GPU's in the device manager, and/or physically remove from the system.