API Debugger Tutorial

1 INTRODUCTION

OpenCL™ API Debugger is a Microsoft Visual Studio* plug-in that enables debugging of OpenCL applications by allowing developers to monitor and understand their OpenCL environment. API Debugger is a new feature of the Compute Code Builder found in the Intel® Integrated Native Developer Experience (Intel® INDE) and the Intel® SDK for OpenCL™ Applications 2014 Beta.

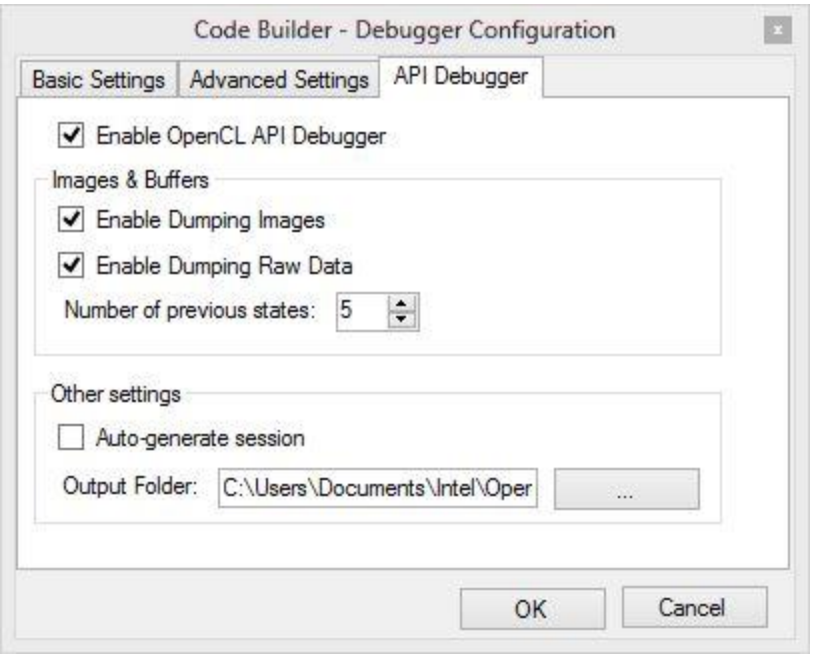
The API Debugger lets developers view the list of all OpenCL API calls that their applications call with an ability to view information about the function parameters, return values, and execution times. The plug-in also allows for viewing various OpenCL objects that exist in memory while the application is executing. You can read more about API Debugger and its features in the Intel® SDK for OpenCL™ Applications 2014 Beta – User's Guide.

In this tutorial we will show one important usage of API Debugger and that is to debug and find the root cause of an application failure in the presence of only the kernel source code and the application binary (no host side source code). Note that a very nice feature of the API Debugger is the ability to show the kernel source code even if it is embedded in the application being debugged. In the latter case, it is possible to identify the source of the kernel bug but not fix it in the absence of full source code for the application.

For demonstration purposes we will introduce a bug in the sobel filter kernel from the JumpStart tutorial and show you how to identify the bug without looking at the host source code.

2 ENABLE API DEBUGGER IN VISUAL STUDIO*

The first step is to make sure the API Debugger is enabled in Visual Studio. To do this, click **Tools→Code Builder – Options** in Visual Studio or press **Ctrl+1** to open the Debugger Configuration dialog as shown below.



Make sure you have at least the **Enable OpenCL API Debugger** option checked and click OK. Once the plug-in is enabled you can start debugging your application.

3 LAUNCHING THE APPLICATION

As mentioned previously, we introduced a bug in the sobel application by changing the kernel argument types for width and height as shown below.

```
kernel void
sobel(__global uchar4* src, __global uchar4* dst, ushort width, ushort height)
{
```

When we try to run the application with this new kernel, the application fails at the point the kernel arguments are set. Assuming no proper error handling, this is very difficult to detect without full source code. Now let's see how this issue can be debugged.

Since we have only the application binary and the OpenCL kernel, we need to somehow launch the application to be debugged from Visual Studio. Start the Visual Studio command prompt from the Start menu or alternately add %VSINSTALLDIR%\Common7\IDE\ (where %VSINSTALLDIR% refers to the directory where Visual Studio is installed) to the PATH environment variable and start a command prompt. Launch the application from the command prompt like so:

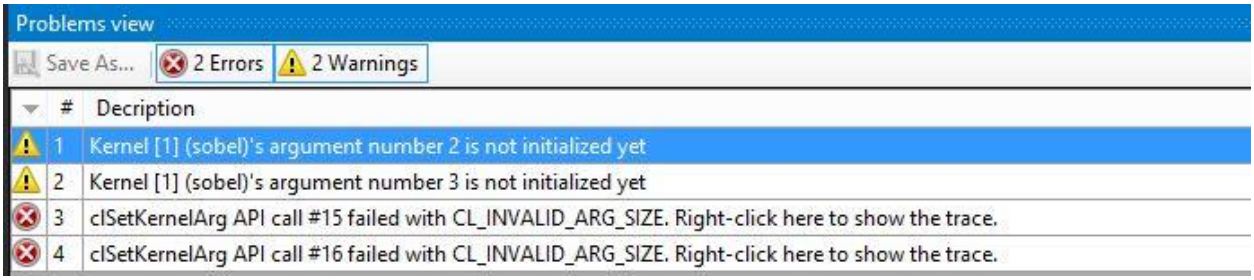
devenv <application>

where <application> is the full or relative path of the application we are trying to debug, say sobel.exe.

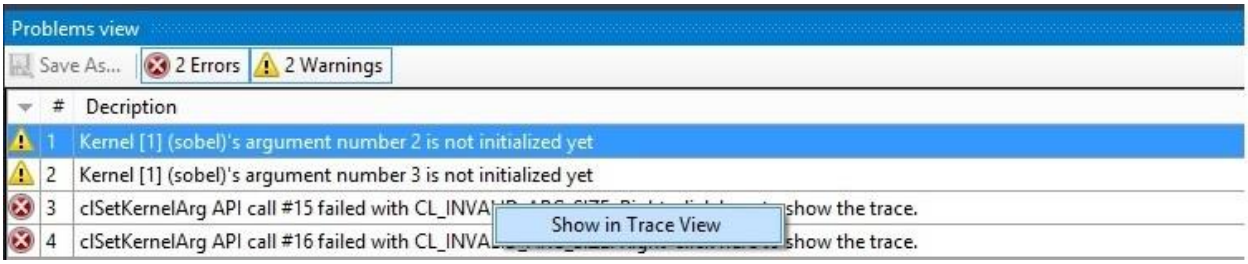
At this point, Visual Studio should launch with the application to be debugged. Make sure the API Debugger is enabled as shown in the previous section. Start debugging by pressing **F5**. The application will launch but will fail without executing the OpenCL kernel.

4 PROBLEMS VIEW AND TRACE VIEW

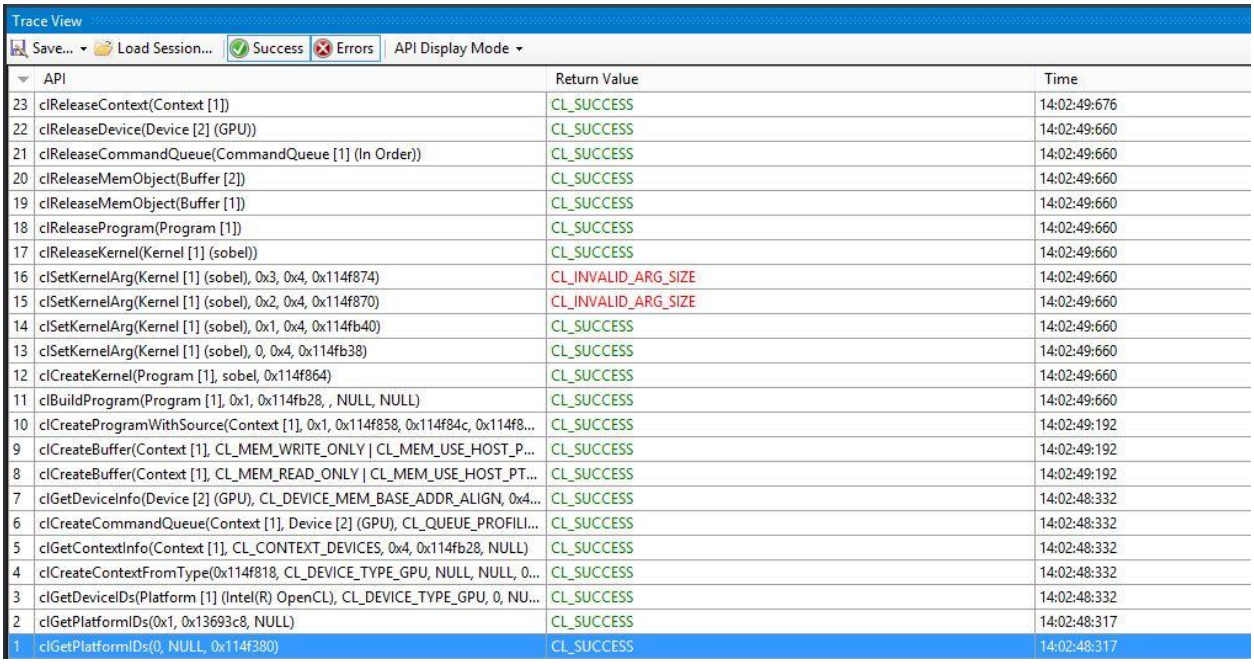
Open the Problems View by going to **Tools→Code Builder→OpenCL Debugger→Problems View**. The **Problems View** window will list all the warnings and errors resulting from running the OpenCL application. We are particularly interested in the errors (but it's good practice to check the warnings too). You will see two errors as a result of calling clSetKernelArg() as shown below.



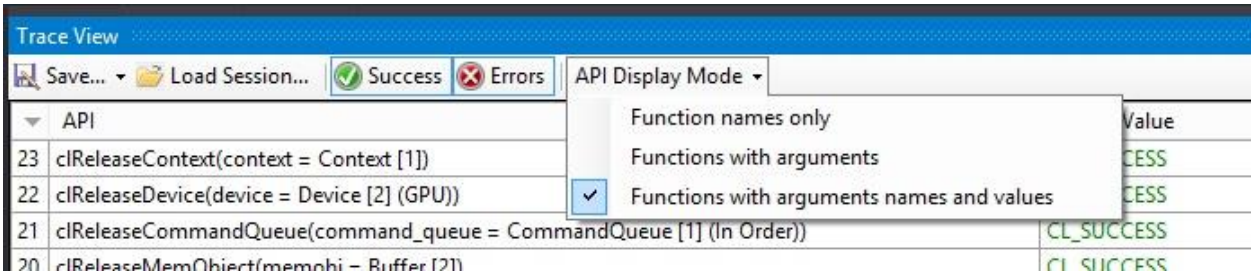
The above view only tells us half the story—that the call failed with the CL_INVALID_ARG_SIZE. Now right-click one of the errors and select **Show in Trace View**.



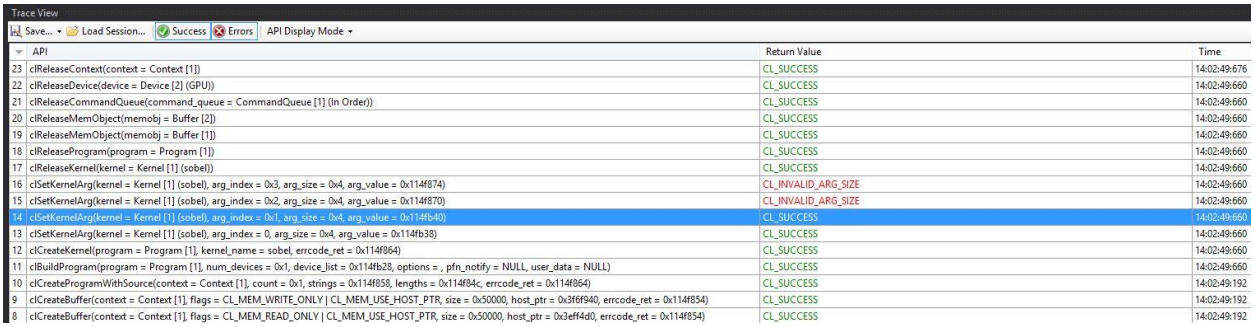
This will open the **Trace View** window with the OpenCL API trace.



In the Trace View window select **Functions with arguments names and values** from the list of **API Display Mode** as shown below.



This verbose display mode gives additional API details including the number of arguments and their sizes.



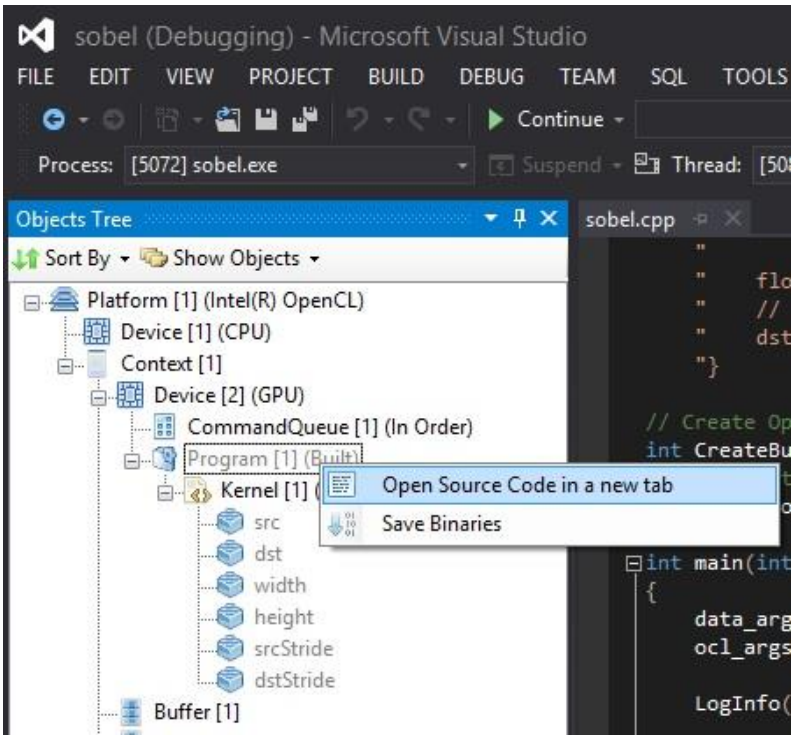
Looking at the offending calls and the return value of clSetKernelArg(), it is obvious there is some problem with setting kernel arguments #2 and #3. Let's look at the kernel signature again.

```
kernel void
sobel(__global uchar4* src, __global uchar4* dst, ushort width, ushort height)
{
```

You can see from the **Trace View** details that the host code set the kernel arguments whose size is 4 (from arg_size = 0x4 for the two offending calls), whereas the OpenCL kernel is accepting ushort whose size is 2. Now we know that either the host code or the kernel needs to be fixed so that both types match.

4.1 VIEWING EMBEDDED KERNEL SOURCE CODE

As mentioned previously, API Debugger can show kernel source code even if it is embedded in the host binary. As before, make sure API Debugger is enabled and launch your application from Visual Studio. After breaking into the debugger open the Objects Tree View (Tools→Code Builder – OpenCL Debugger→Objects Tree View) and look for the built program node (something like "Program [1] (Built)"). Then right-click on the program node and choose "Open Source Code in a new tab" to view the kernel source code.



Note that the above binary was built without debugging information. So it is possible to inspect the kernel source code even when the binary is built in release mode.

5 SUMMARY

This tutorial shows how **API Debugger** can help determine the root cause of certain errors in OpenCL applications, particularly when source code is not available. API Debugger has much more functionality and nicely extends the debugging capabilities of Visual Studio by providing capabilities for debugging OpenCL host applications from the IDE.

Notices