Dell SDK for Monitors
Whitepaper

for SDK version 3.x

2020 – 09 Rev. A00

Contents

Introduction

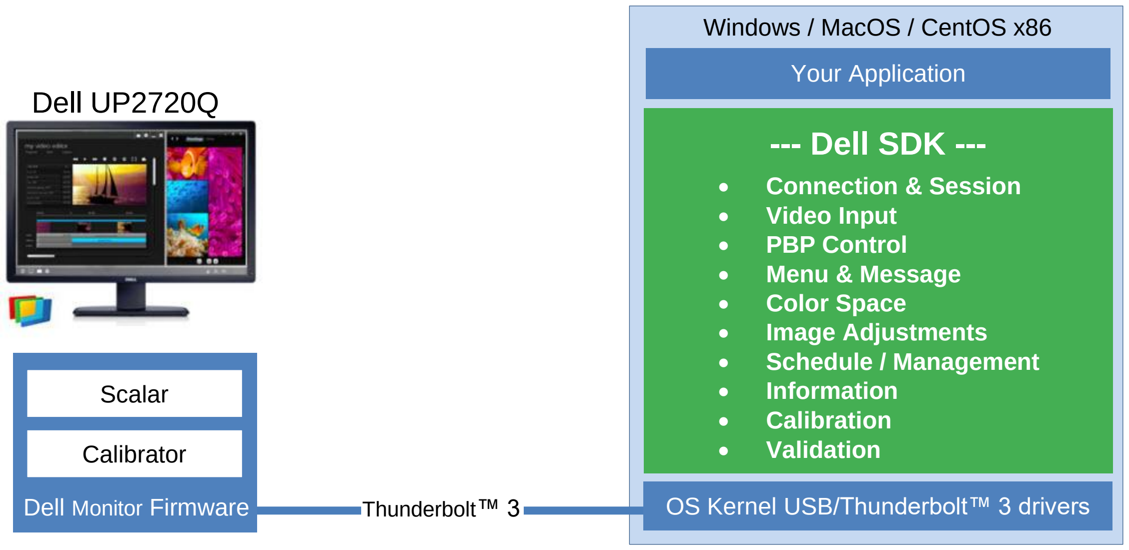
The Dell Software Development Kit (SDK) for Monitors will enable you to manage, control and create custom applications specifically for Dell Ultra Sharp Premier Monitor UP2720Q and UP3221Q across multiple operating systems via a Thunderbolt™ 3 connection.

The SDK gives you access to control just about everything, if not more, on what you can do manually on the monitor’s OSD menu. This allows you to automate certain settings within your application without the need to go through the OSD menu structure manually.

With the SDK, all the low-level integration with the various supported OS and their differences have been done. This allows you to focus your resources on developing your application without worrying about how to talk to different OS API to connect to the monitor.

Porting your application is also made easier as the API is the same across all supported OS. The supported OS are Windows, MacOS and CentOS x86.

Architecture



The figure above shows the architecture of the Dell SDK for Monitors. The SDK can interact with the monitor via the supplied USB-C to A cable or the supplied Thunderbolt cable.

The SDK is supplied as a

- DLL and LIB for Windows (x86 32bits)
- DYLIB for MacOS x86
- SO for CentOS (x86 32bits, 64bits)

SDK 3.1 is backwards compatible with past Dell UltraSharp Premium monitors supporting Dell Monitor SDK.

Please refer to SDK API Guide released with your product for the exact APIs implemented with the product.

Key SDK Features

The SDK is organized into various sets of APIs for different set of functionalities. In essence, the SDK enables you to control various aspects of the monitor that you could normally perform manually with the OSD, and more. The list below shows the key sets of APIs.

Monitor Management – This module provides the initialization, connecting to the shutdown of the SDK within your application. It also provides informational APIs like getting and setting the asset tag.

Power Management – Control the power settings of the USB and TBT ports as well as turning the monitor to standby or back on using these APIs.

Image Management – Control everything from sharpness to DCI Masking to Markers, toggling blue channel and more with these set of APIs.

Color Management – This set of APIs gives you extensive control of the Color Space. From simple functionalities like switching between different color space, to more advanced functionalities like changing the gamut, hue, saturation and white point.

Video Input Management – Switch between different video inputs using these APIs.

PBP Management – Allows your application to control the PBP functionalities on the monitor easily, including swapping video, changing white point etc.

OSD Management – Set OSD language or lock/unlock the OSD buttons via this module.

System Management – This set of APIs will enable you to set the date and time easily from your application, perform LCD conditioning or perform a factory reset.

Calibration/Validation – The APIs in this module allows you to control the settings of the Calibration and Validation of the monitor.

Scheduler – With this set of APIs, you can set the calibration and validation schedules via your own application.

Development with the SDK

Implementation with the Dell SDK is easy. We provide you with a well commented sample code below on discovering and connecting to the monitor in a simple C program, so that you can get started quickly and focus on your core application functionalities.

A simple application flow on initialization and connecting to a monitor is as follow:

1. Initialize the SDK: **Initialize()**
2. Get connected monitors
    a. Use **GetAvailableMonitors** to just get a count, *OR*
    b. Use **GetAvailableMonitorsDetail** to get count and the associated array of monitor model name
3. Connect to monitor using index: **ConnectMonitor(index)** (or **ConnectMonitorByServiceTag**) where index is 0 to (count-1) returned in step 2.
4. Perform your application processes…
5. Disconnect monitor: **DisconnectMonitor()**
6. Shutdown the SDK: **Shutdown()**

The corresponding code in C will be as below. For simplicity, we shall exit the program on any error.

```c
int main(int argc, char *argv[])
{
  int res;
  BYTE b8Val;

  /** Initialize the SDK **/
  res = Initialize();
  if (res != MONITOR_SUCCESS)
  {
    fprintf( stderr, "Error initializing SDK\n" );
    return 1;
  }

  /** Get Available Monitors **/
  MonitorDetailStructType *arrMonitorDetail = NULL;
  res = GetAvailableMonitorsDetail(&b8Val, &arrMonitorDetail);
  if (res != MONITOR_SUCCESS)
  {
    fprintf( stderr, "Error discovering supported monitors\n" );
    Shutdown();
    return 1;
  }

  /** Check if any supported monitor found **/
  if (b8Val == 0)
  {
    fprintf( stderr, "No supported monitor found\n" );
    free(arrMonitorDetail);
    Shutdown();
    return 1;
  }

  /** OPTIONAL - Let's print out **/
  for (int m = 0; m < (int)b8Val; m++) {
    fprintf( stdout, "[%d] Monitor: %s / ServiceTag %s ", m,
        arrMonitorDetail[m].MonitorName, arrMonitorDetail[m].ServiceTag);
  }

  free(arrMonitorDetail);

  /** Connect to the first monitor **/
  res = ConnectMonitor(0);
  if (res != MONITOR_SUCCESS)
  {
    fprintf( stderr, "Error connecting to monitor[0]\n" );
    Shutdown();
    return 1;
  }
  /***********************************************************/
  /** FROM THIS POINT ONWARDS, DO YOUR APPLICATION PROCESS **/
  /**                                                      **/
  /**                                                      **/
  /** In this example, we switch to NATIVE color space and **/
  /** lower the luminance level to the lowest level.       **/
  /** You should see the monitor reflecting this change.   **/
  /**                                                      **/
  /***********************************************************/

  /** Switch to Color Space Native **/
  res = SetColorSpaceState(COLOR_SPACE2_NATIVE);
  if (res != MONITOR_SUCCESS)
  {
    fprintf( stderr, "Error switching to color space NATIVE\n" );
    Shutdown();
    return 1;
  }

  /** Set the luminance to lowest **/
  res = SetLuminance(45);
  if (res != MONITOR_SUCCESS)
  {
    fprintf( stderr, "Error setting luminance level to 45\n" );
    Shutdown();
    return 1;
  }

  /***********************************************************/
  /** END OF OUR SIMPLE APPLICATION PROCESS                **/
  /***********************************************************/

  /** Disconnect Monitor **/
  res = DisconnectMonitor();

  /** Shutdown SDK **/
  res = Shutdown();

  return 0;
}
```