Vulkan 1.1 Compatibility Statement
SDK Strategy

Does Vulkan 1.1 support my Vulkan 1.0 application?

Did you know that since Vulkan 1.1 has been released by Khronos, there is no longer a need for a LunarG SDK based on 1.0 headers? Vulkan 1.1 is a minor release, and as such adds additional functionality to the API without modifying the behavior of the already existing Vulkan 1.0 functionality. Therefore, it guarantees compatibility for 1.0 applications:

1. Application binaries built from a 1.0 Vulkan header will work with Vulkan 1.1 versions of the validation layers, Vulkan runtime (loader), and drivers.
2. Application source written to use the 1.0 Vulkan header can be built with the Vulkan 1.1 SDK.

LunarG no longer plans to release Vulkan SDKs based on a 1.0 Vulkan header. To continue to get the latest validation layers and other SDK components, you can use the Vulkan 1.1 versioned SDKs and be confident that the validation layers, tools, and Vulkan runtime will be compatible with your Vulkan 1.0 applications.

As well, IHV Windows driver updates will continue to bundle the Vulkan Runtime and they will be moving to a Vulkan 1.1 version of the Vulkan Runtime.

How to migrate your applications to Vulkan 1.1

Starting with Vulkan 1.1, the VkApplicationInfo substructure of VkInstanceCreateInfo is no longer optional if you want to create a Vulkan 1.1 (or newer) application. There is now a process for properly creating a Vulkan application for Vulkan 1.1 and newer:

1. First, find out what API version the runtime supports for Instances:

```
uint32 apiVersion = 0;
vkEnumerateInstanceVersion(&apiVersion);
```

   This is the maximum API version an instance can be created for on a system.
   If the returned version is greater-than or equal-to Vulkan 1.0 you may create a Vulkan 1.1 Instance.

```
if (VK_MAKE_VERSION(1, 1, 0) <= apiVersion) {
      // 1.1 or newer is available
}
```

2. Next, you must create a VkApplicationInfo structure, and set the “apiVersion” field to the following:

```
VkApplicationInfo myApplicationInfo = {};
...
myApplicationInfo.apiVersion = VK_MAKE_VERSION(1,1,0);
```

3. Then, you must set the VkInstanceCreateInfo “pApplicationInfo” to point to the above application info struct:

```
VkInstanceCreateInfo myInstanceCreateInfo = {};
…
myInstanceCreateInfo.pApplicationInfo = &myApplicationInfo;
```

4. Finally, call vkCreateInstance as you normally would with your VkInstanceCreateInfo structure.

Ensuring that your physical devices support Vulkan 1.1

An additional process is necessary to ensure that your physical devices support Vulkan 1.1:

1. Once you have a Vulkan 1.1 instance available, you must check which physical devices support Vulkan 1.1. To do this on any physical device, call:

```
VkPhysicalDeviceProperties properties = {};
…
vkGetPhysicalDeviceProperties(physicalDevice, &properties);
```

2. The supported API version of this physical device will be given in the “apiVersion” field of the VkPhysicalDeviceProperties structure and will take the same format as the “apiVersion” field in vkEnumerateInstanceVersion.

In Summary

If you intend to support Vulkan 1.0 loaders, you can't link directly to any Vulkan 1.1 commands.

You cannot legally use Vulkan 1.1 functionality if you do any of the following:

a. Fail to call vkEnumerateInstanceVersion
b. Set myInstanceCreateInfo.pApplicationInfo = NULL
c. Set myApplicationInfo.apiVersion = 0
d. Set myApplicationInfo.apiVersion = VK_MAKE_VERSION(1, 0, <anything>);

To legally use Vulkan 1.1 functionality in your application you **should**:

1. Query vkEnumerateInstanceVersion by calling vkGetInstanceProcAddr

2. Call vkEnumerateInstanceVersion and determine that the API version returned is greater than or equal to VK_MAKE_VERSION(1, 1, 0)
3. Create an instance with the application info defined and the “apiVersion” set to VK_MAKE_VERSION(1, 1, 0)
4. Select a physical device that supports Vulkan 1.1

All these steps are required to properly initialize Vulkan 1.1. If you don’t perform all of these steps, you can only be guaranteed that Vulkan 1.0 functionality is present.