Vulkanised 2024

The 6th Vulkan Developer Conference
Sunnyvale, California | February 5-7, 2024

Using Vulkan Validation Effectively

Jeremy Gebben, LunarG

Presentation:

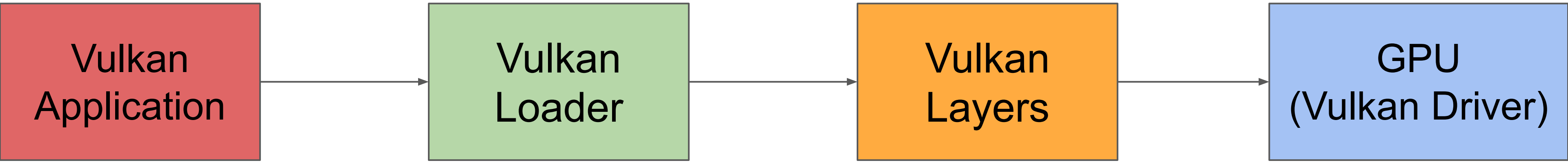
https://bit.ly/48Wb5sL

Agenda

- Valid Usage and VUIDs
- Example error walkthrough
- Debug utilities extension
- Enabling and configuring validation

What is a Vulkan Layer

- A shared library that intercepts Vulkan commands from an application
- The Loader is responsible for managing layers and drivers



Why the Vulkan Validation Layer?

- OpenGL had many error code checks that drivers had to implement
- Checks always enabled in drivers (useless CPU overhead)
- Most checking was similar in all drivers (duplicated effort)
- Vulkan moved error checking to the Validation Layer
    - Enabled only during development, no overhead in released applications

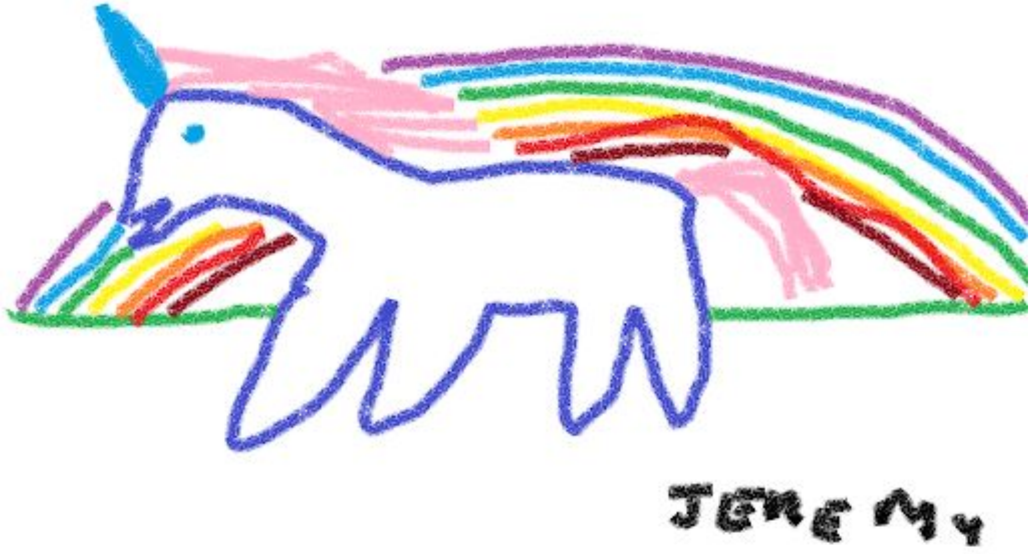
What is Valid Usage

"set of conditions that must be met in order to achieve well-defined run-time behavior in an application."

- The driver assumes the application provides valid data
- If a Valid Usage is broken, the result is **undefined behavior**
    - For the current command and everything following it.
- **Advice**: Fix the first error message first

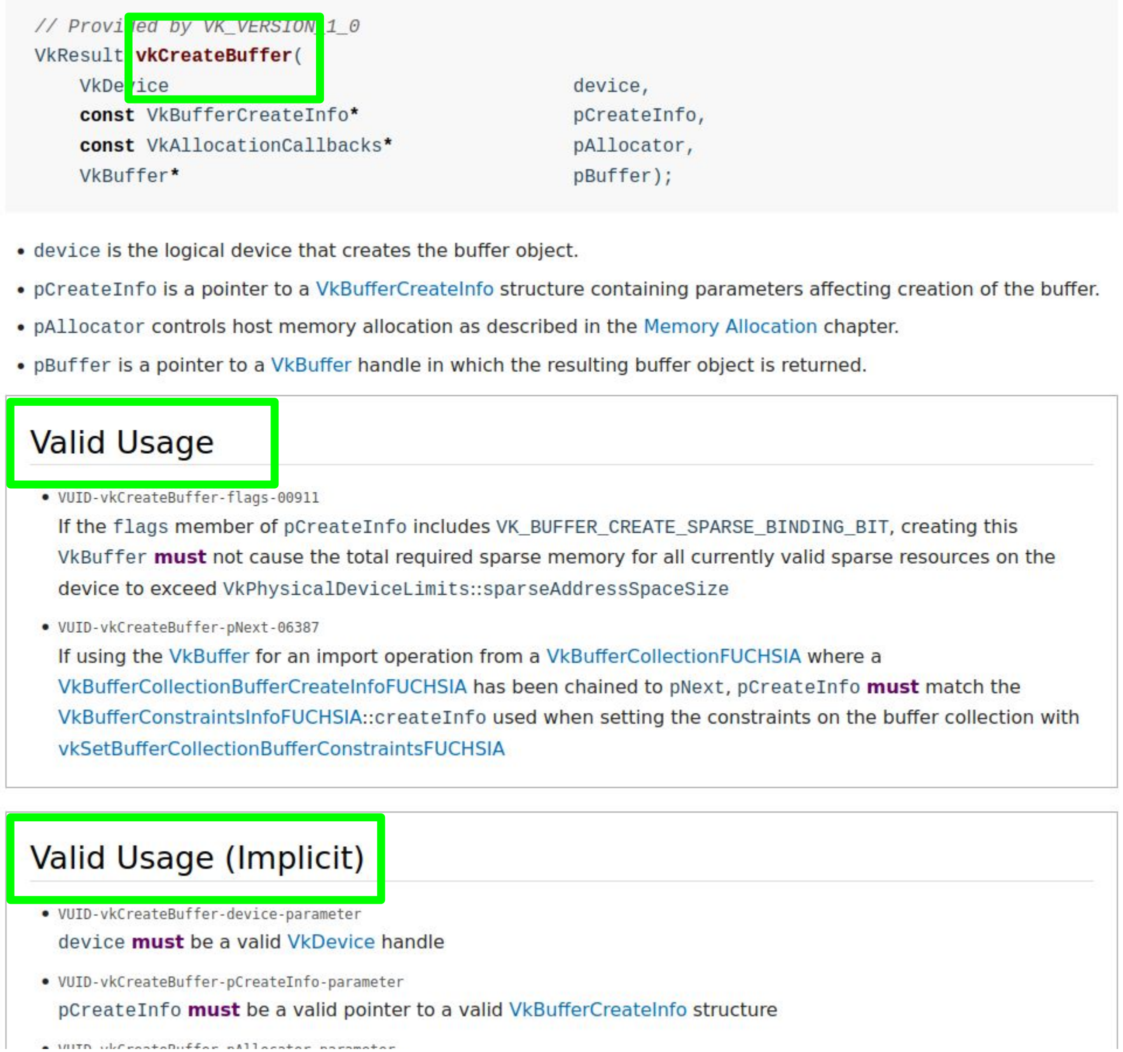
Undefined Behavior

- … App might work fine
- … output might be corrupt
- … GPU might hang
- … Computer might blow up!
- Anything is possible!



VUID

- **V**alid **U**sage **ID**
- Automatically generated number when spec is released
- Unique ID to map each error back to the spec
  - Format: VUID-{command or structure}-{parameter, field or None}-{Number}
  - Example: VUID-vkCmdDraw-None-07850
- Number is unique per Valid Usage, but could apply to multiple commands:
  - VUID-vkCmdDrawMultiEXT-None-07850
  - VUID-vkCmdDrawIndexed-None-07850
  - VUID-vkCmdDrawMultiIndexedEXT-None-07850



```
// Provided by VK_VERSION_1_0
VkResult vkCreateBuffer(
    VkDevice                                    device,
    const VkBufferCreateInfo*                   pCreateInfo,
    const VkAllocationCallbacks*                pAllocator,
    VkBuffer*                                   pBuffer);
```

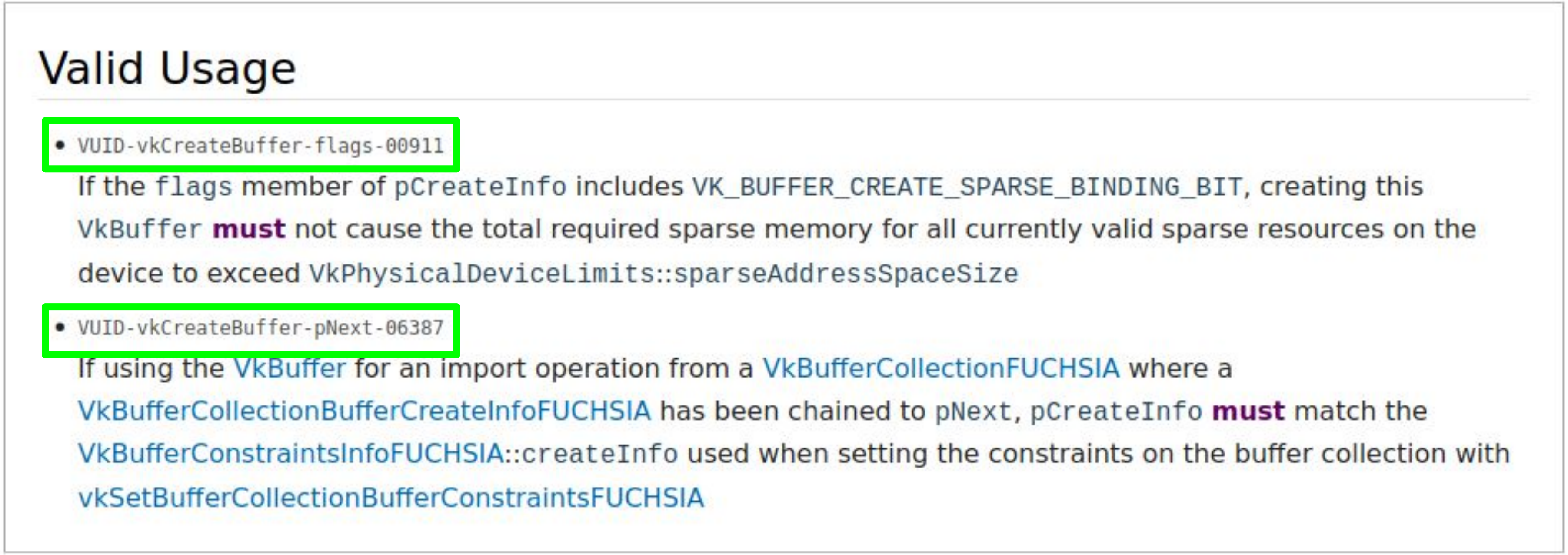
- `device` is the logical device that creates the buffer object.
- `pCreateInfo` is a pointer to a VkBufferCreateInfo structure containing parameters affecting creation of the buffer.
- `pAllocator` controls host memory allocation as described in the Memory Allocation chapter.
- `pBuffer` is a pointer to a VkBuffer handle in which the resulting buffer object is returned.

Valid Usage

- `VUID-vkCreateBuffer-flags-00911`
  If the `flags` member of `pCreateInfo` includes `VK_BUFFER_CREATE_SPARSE_BINDING_BIT`, creating this `VkBuffer` **must** not cause the total required sparse memory for all currently valid sparse resources on the device to exceed `VkPhysicalDeviceLimits::sparseAddressSpaceSize`
- `VUID-vkCreateBuffer-pNext-06387`
  If using the VkBuffer for an import operation from a VkBufferCollectionFUCHSIA where a VkBufferCollectionBufferCreateInfoFUCHSIA has been chained to `pNext`, `pCreateInfo` **must** match the VkBufferConstraintsInfoFUCHSIA::`createInfo` used when setting the constraints on the buffer collection with vkSetBufferCollectionBufferConstraintsFUCHSIA

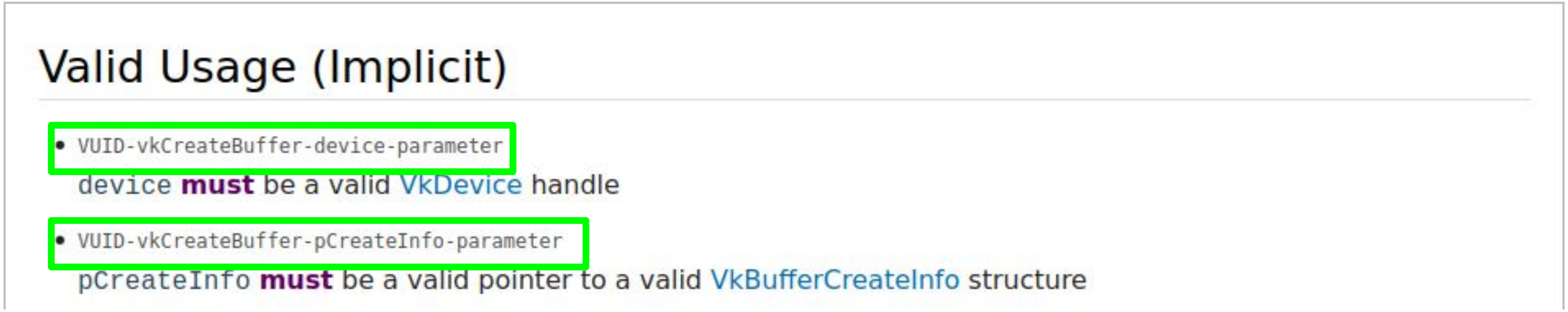
Valid Usage (Implicit)

- `VUID-vkCreateBuffer-device-parameter`
  `device` **must** be a valid VkDevice handle
- `VUID-vkCreateBuffer-pCreateInfo-parameter`
  `pCreateInfo` **must** be a valid pointer to a valid VkBufferCreateInfo structure
- `VUID-vkCreateBuffer-pAllocator-parameter`



Valid Usage

- `VUID-vkCreateBuffer-flags-00911`
  If the `flags` member of `pCreateInfo` includes `VK_BUFFER_CREATE_SPARSE_BINDING_BIT`, creating this `VkBuffer` **must** not cause the total required sparse memory for all currently valid sparse resources on the device to exceed `VkPhysicalDeviceLimits::sparseAddressSpaceSize`
- `VUID-vkCreateBuffer-pNext-06387`
  If using the VkBuffer for an import operation from a VkBufferCollectionFUCHSIA where a VkBufferCollectionBufferCreateInfoFUCHSIA has been chained to `pNext`, `pCreateInfo` **must** match the VkBufferConstraintsInfoFUCHSIA::`createInfo` used when setting the constraints on the buffer collection with vkSetBufferCollectionBufferConstraintsFUCHSIA



Valid Usage (Implicit)

- `VUID-vkCreateBuffer-device-parameter`
  `device` **must** be a valid VkDevice handle
- `VUID-vkCreateBuffer-pCreateInfo-parameter`
  `pCreateInfo` **must** be a valid pointer to a valid VkBufferCreateInfo structure

Advice: Read the spec!

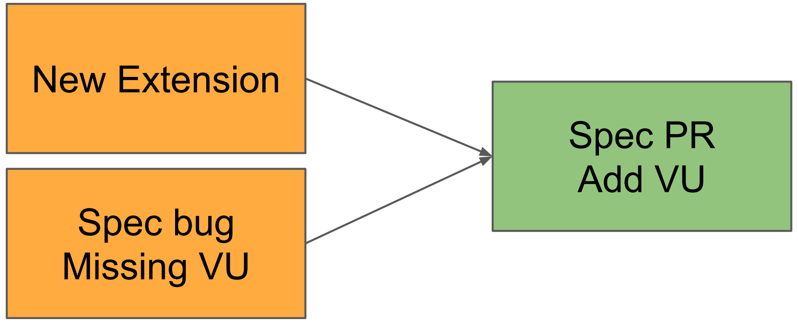
- “Read the spec early and often”
- Has most of the answers!
- Tips for efficient spec reading:
    - Read the section where the VUID is defined
    - Search for words / phrases from the VUID text in the rest of the spec
    - Read VUIDs for the command(s) you’re using and any associated structures

Life cycle of a VU

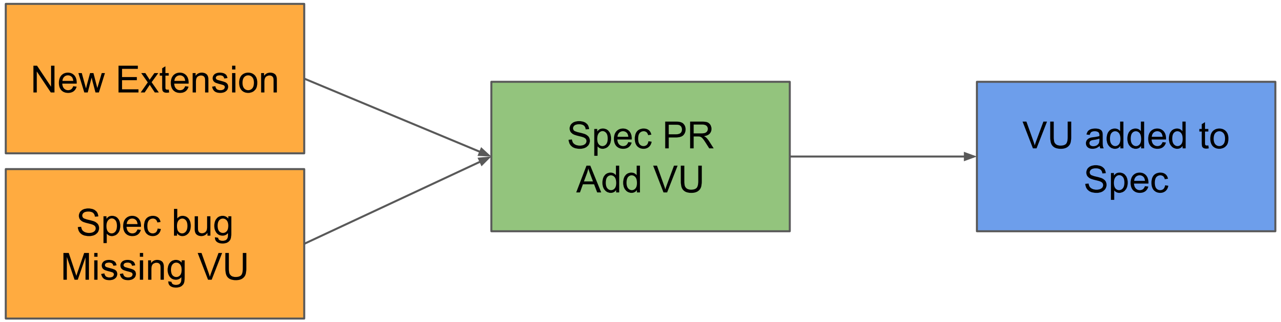
New Extension

Spec bug
Missing VU

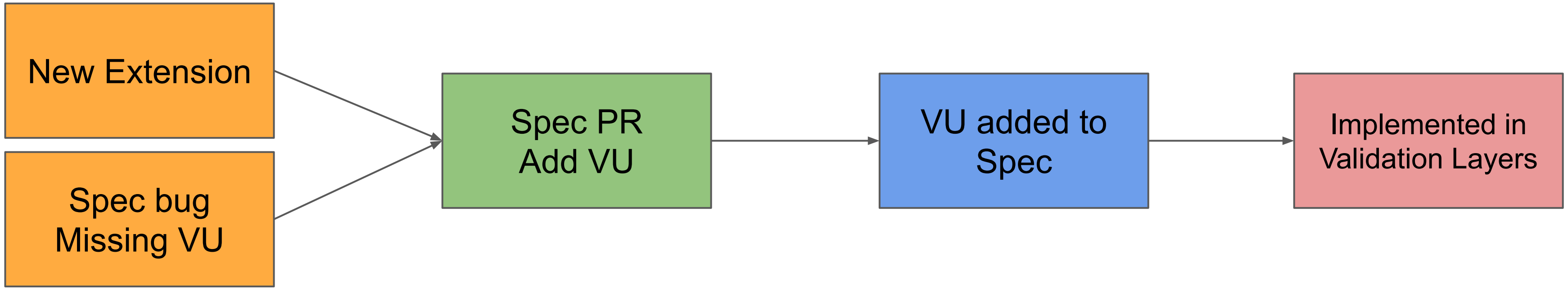
Life cycle of a VU



Life cycle of a VU



Life cycle of a VU



Types of validation - API Usage

- Developer is using an API incorrectly
    - `vkCreateImage(VK_IMAGE_TYPE_2D, extent.depth = 8);`
- Setting depth, but using a 2D image (not 3D)

Types of validation - Device Features

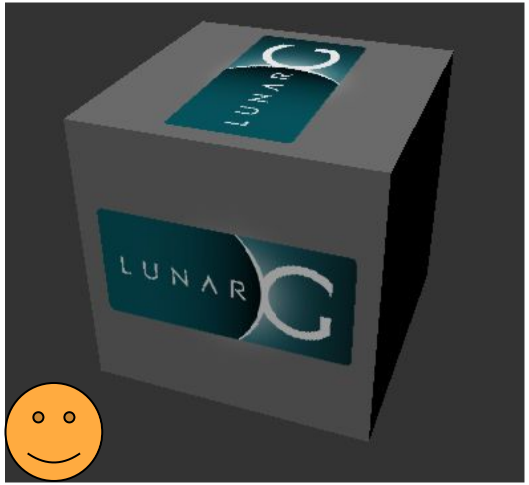
- Unsuccessful interaction between application and system features
- `VkSubpassDescription::colorAttachmentCount = 5;`
- This *might* succeed or fail, it will depend on the system
  - **maxColorAttachments**
  - Minimum required is only 4

Types of validation - Resource constraints

- Unsuccessful interaction between application and the current system state.
- Memory Allocation is the classic example
  - `VkMemoryAllocateInfo::allocationSize = HUGE_SIZE;`
  - `VkResult vkAllocateMemory(..., VkDeviceMemory *pMemory);`
- This *might* fail depending on the what else is happening on the system
- **Advice**: Always handle VkResult return values
  - These errors can happen in a correct application!

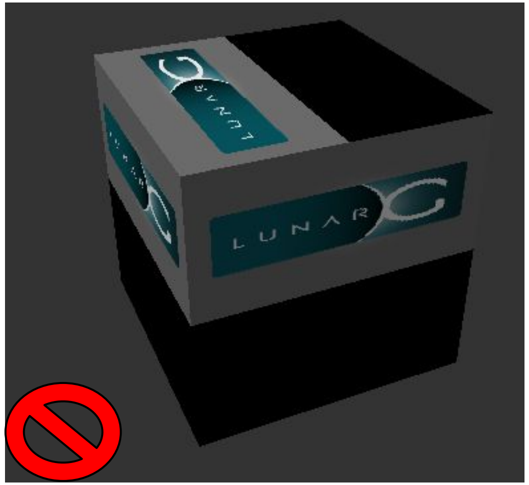
An example error: vkcube

```
VkBufferImageCopy copy_region = {
    .bufferOffset = 0,
    .bufferRowLength = demo->staging_texture.tex_width,
    .bufferImageHeight = demo->staging_texture.tex_height,
    .imageSubresource = {VK_IMAGE_ASPECT_COLOR_BIT, 0, 0, 1},
    .imageOffset = {0, 0, 0},
    .imageExtent = {demo->staging_texture.tex_width, demo->staging_texture.tex_height, 1},
};
vkCmdCopyBufferToImage(demo->cmd, demo->staging_texture.buffer, demo->textures[i].image,
                       VK_IMAGE_LAYOUT_TRANSFER_DST_OPTIMAL, 1, &copy_region)
```



An example error: vkcube

```
VkBufferImageCopy copy_region = {
    .bufferOffset = 0,
    .bufferRowLength = demo->staging_texture.tex_width * 2, // ERROR!
    .bufferImageHeight = demo->staging_texture.tex_height,
    .imageSubresource = {VK_IMAGE_ASPECT_COLOR_BIT, 0, 0, 1},
    .imageOffset = {0, 0, 0},
    .imageExtent = {demo->staging_texture.tex_width, demo->staging_texture.tex_height, 1},
};
vkCmdCopyBufferToImage(demo->cmd, demo->staging_texture.buffer, demo->textures[i].image,
                       VK_IMAGE_LAYOUT_TRANSFER_DST_OPTIMAL, 1, &copy_region)
```



Validation Output: Error Message

```
VUID-vkCmdCopyBufferToImage-pRegions-00171(ERROR / SPEC): msgNum: 1867332608 - Validation Error: [
VUID-vkCmdCopyBufferToImage-pRegions-00171 ] Object 0: handle = 0x56313fd28a00, type =
VK_OBJECT_TYPE_COMMAND_BUFFER; Object 1: handle = 0xd175b40000000013, type = VK_OBJECT_TYPE_BUFFER; |
MessageID = 0x6f4d3c00 | vkCmdCopyBufferToImage: pRegion[0] is trying to copy 523264 bytes plus 0 offset
to/from the VkBuffer (VkBuffer 0xd175b40000000013[]) which exceeds the VkBuffer total size of 262144 bytes.
The Vulkan spec states: srcBuffer must be large enough to contain all buffer locations that are accessed
according to Buffer and Image Addressing, for each element of pRegions
(https://vulkan.lunarg.com/doc/view/1.3.243.0/windows/1.3-extensions/html/vkspec.html#VUID-vkCmdCopyBufferToI
mage-pRegions-00171)
    Objects: 2
        [0] 0x56313fd28a00, type: 6, name: NULL
        [1] 0xd175b40000000013, type: 9, name: NULL
```

Error Message - Basic Info

```
VUID-vkCmdCopyBufferToImage-pRegions-00171(ERROR / SPEC): msgNum: 1867332608 - Validation Error: [
VUID-vkCmdCopyBufferToImage-pRegions-00171 ] Object 0: handle = 0x56313fd28a00, type =
VK_OBJECT_TYPE_COMMAND_BUFFER; Object 1: handle = 0xd175b40000000013, type = VK_OBJECT_TYPE_BUFFER; |
MessageID = 0x6f4d3c00 | vkCmdCopyBufferToImage: pRegion[0] is trying to copy 523264 bytes plus 0 offset
to/from the VkBuffer (VkBuffer 0xd175b40000000013[]) which exceeds the VkBuffer total size of 262144 bytes.
The Vulkan spec states: srcBuffer must be large enough to contain all buffer locations that are accessed
according to Buffer and Image Addressing, for each element of pRegions
(https://vulkan.lunarg.com/doc/view/1.3.243.0/windows/1.3-extensions/html/vkspec.html#VUID-VkCmdCopyBufferToI
mage-pRegions-00171)
    Objects: 2
        [0] 0x56313fd28a00, type: 6, name: NULL
        [1] 0xd175b40000000013, type: 9, name: NULL
```

Error Message - Basic Info

```
VUID-vkCmdCopyBufferToImage-pRegions-00171(ERROR / SPEC): msgNum: 1867332608 - Validation Error: [
VUID-vkCmdCopyBufferToImage-pRegions-00171 ] Object 0: handle = 0x56313fd28a00, type =
VK_OBJECT_TYPE_COMMAND_BUFFER; Object 1: handle = 0xd175b40000000013, type = VK_OBJECT_TYPE_BUFFER; |
MessageID = 0x6f4d3c00 | vkCmdCopyBufferToImage: pRegion[0] is trying to copy 523264 bytes plus 0 offset
to/from the VkBuffer (VkBuffer 0xd175b40000000013[]) which exceeds the VkBuffer total size of 262144 bytes.
The Vulkan spec states: srcBuffer must be large enough to contain all buffer locations that are accessed
according to Buffer and Image Addressing, for each element of pRegions
(https://vulkan.lunarg.com/doc/view/1.3.243.0/windows/1.3-extensions/html/vkspec.html#VUID-VkCmdCopyBufferToI
mage-pRegions-00171)
    Objects: 2
        [0] 0x56313fd28a00, type: 6, name: NULL
        [1] 0xd175b40000000013, type: 9, name: NULL
```

Error Message - Basic Info

```
VUID-vkCmdCopyBufferToImage-pRegions-00171(ERROR / SPEC): msgNum: 1867332608 - Validation Error: [
VUID-vkCmdCopyBufferToImage-pRegions-00171 ] Object 0: handle = 0x56313fd28a00, type =
VK_OBJECT_TYPE_COMMAND_BUFFER; Object 1: handle = 0xd175b40000000013, type = VK_OBJECT_TYPE_BUFFER; |
MessageID = 0x6f4d3c00 | vkCmdCopyBufferToImage: pRegion[0] is trying to copy 523264 bytes plus 0 offset
to/from the VkBuffer (VkBuffer 0xd175b40000000013[]) which exceeds the VkBuffer total size of 262144 bytes.
The Vulkan spec states: srcBuffer must be large enough to contain all buffer locations that are accessed
according to Buffer and Image Addressing, for each element of pRegions
(https://vulkan.lunarg.com/doc/view/1.3.243.0/windows/1.3-extensions/html/vkspec.html#VUID-VkCmdCopyBufferToI
mage-pRegions-00171)
    Objects: 2
        [0] 0x56313fd28a00, type: 6, name: NULL
        [1] 0xd175b40000000013, type: 9, name: NULL
```

- msgNum / MessageID is a hash of the VUID string, used for handling duplicate messages

Error Message - Main message

```
VUID-vkCmdCopyBufferToImage-pRegions-00171(ERROR / SPEC): msgNum: 1867332608 - Validation Error: [
VUID-vkCmdCopyBufferToImage-pRegions-00171 ] Object 0: handle = 0x56313fd28a00, type =
VK_OBJECT_TYPE_COMMAND_BUFFER; Object 1: handle = 0xd175b40000000013, type = VK_OBJECT_TYPE_BUFFER; |
MessageID = 0x6f4d3c00 | vkCmdCopyBufferToImage: pRegion[0] is trying to copy 523264 bytes plus 0 offset
to/from the VkBuffer (VkBuffer 0xd175b40000000013[]) which exceeds the VkBuffer total size of 262144 bytes.
The Vulkan spec states: srcBuffer must be large enough to contain all buffer locations that are accessed
according to Buffer and Image Addressing, for each element of pRegions
(https://vulkan.lunarg.com/doc/view/1.3.243.0/windows/1.3-extensions/html/vkspec.html#VUID-vkCmdCopyBufferToI
mage-pRegions-00171)
    Objects: 2
        [0] 0x56313fd28a00, type: 6, name: NULL
        [1] 0xd175b40000000013, type: 9, name: NULL
```

Error Message - Main message

```
VUID-vkCmdCopyBufferToImage-pRegions-00171(ERROR / SPEC): msgNum: 1867332608 - Validation Error: [
VUID-vkCmdCopyBufferToImage-pRegions-00171 ] Object 0: handle = 0x56313fd28a00, type =
VK_OBJECT_TYPE_COMMAND_BUFFER; Object 1: handle = 0xd175b40000000013, type = VK_OBJECT_TYPE_BUFFER; |
MessageID = 0x6f4d3c00 | vkCmdCopyBufferToImage: pRegion[0] is trying to copy 523264 bytes plus 0 offset
to/from the VkBuffer (VkBuffer 0xd175b40000000013[]) which exceeds the VkBuffer total size of 262144 bytes.
The Vulkan spec states: srcBuffer must be large enough to contain all buffer locations that are accessed
according to Buffer and Image Addressing, for each element of pRegions
(https://vulkan.lunarg.com/doc/view/1.3.243.0/windows/1.3-extensions/html/vkspec.html#VUID-vkCmdCopyBufferToI
mage-pRegions-00171)
    Objects: 2
        [0] 0x56313fd28a00, type: 6, name: NULL
        [1] 0xd175b40000000013, type: 9, name: NULL
```

Error Message - Main message

```
VUID-vkCmdCopyBufferToImage-pRegions-00171(ERROR / SPEC): msgNum: 1867332608 - Validation Error: [
VUID-vkCmdCopyBufferToImage-pRegions-00171 ] Object 0: handle = 0x56313fd28a00, type =
VK_OBJECT_TYPE_COMMAND_BUFFER; Object 1: handle = 0xd175b40000000013, type = VK_OBJECT_TYPE_BUFFER; |
MessageID = 0x6f4d3c00 | vkCmdCopyBufferToImage: pRegion[0] is trying to copy 523264 bytes plus 0 offset
to/from the VkBuffer (VkBuffer 0xd175b40000000013[]) which exceeds the VkBuffer total size of 262144 bytes.
The Vulkan spec states: srcBuffer must be large enough to contain all buffer locations that are accessed
according to Buffer and Image Addressing, for each element of pRegions
(https://vulkan.lunarg.com/doc/view/1.3.243.0/windows/1.3-extensions/html/vkspec.html#VUID-vkCmdCopyBufferToI
mage-pRegions-00171)
    Objects: 2
        [0] 0x56313fd28a00, type: 6, name: NULL
        [1] 0xd175b40000000013, type: 9, name: NULL
```

Error Message - Spec Reference

```
VUID-vkCmdCopyBufferToImage-pRegions-00171(ERROR / SPEC): msgNum: 1867332608 - Validation Error: [
VUID-vkCmdCopyBufferToImage-pRegions-00171 ] Object 0: handle = 0x56313fd28a00, type =
VK_OBJECT_TYPE_COMMAND_BUFFER; Object 1: handle = 0xd175b40000000013, type = VK_OBJECT_TYPE_BUFFER; |
MessageID = 0x6f4d3c00 | vkCmdCopyBufferToImage: pRegion[0] is trying to copy  523264 bytes plus 0 offset
to/from the VkBuffer (VkBuffer 0xd175b40000000013[]) which exceeds the VkBuffer total size of 262144 bytes.
The Vulkan spec states: srcBuffer must be large enough to contain all buffer locations that are accessed
according to Buffer and Image Addressing, for each element of pRegions
(https://vulkan.lunarg.com/doc/view/1.3.243.0/windows/1.3-extensions/html/vkspec.html#VUID-vkCmdCopyBufferToI
mage-pRegions-00171)
    Objects: 2
        [0] 0x56313fd28a00, type: 6, name: NULL
        [1] 0xd175b40000000013, type: 9, name: NULL
```

Error Message - Object Handles

```
VUID-vkCmdCopyBufferToImage-pRegions-00171(ERROR / SPEC): msgNum: 1867332608 - Validation Error: [
VUID-vkCmdCopyBufferToImage-pRegions-00171 ] Object 0: handle = 0x56313fd28a00, type =
VK_OBJECT_TYPE_COMMAND_BUFFER; Object 1: handle = 0xd175b40000000013, type = VK_OBJECT_TYPE_BUFFER; |
MessageID = 0x6f4d3c00 | vkCmdCopyBufferToImage: pRegion[0] is trying to copy 523264 bytes plus 0 offset
to/from the VkBuffer (VkBuffer 0xd175b40000000013[]) which exceeds the VkBuffer total size of 262144 bytes.
The Vulkan spec states: srcBuffer must be large enough to contain all buffer locations that are accessed
according to Buffer and Image Addressing, for each element of pRegions
(https://vulkan.lunarg.com/doc/view/1.3.243.0/windows/1.3-extensions/html/vkspec.html#VUID-vkCmdCopyBufferToI
mage-pRegions-00171)
    Objects: 2
        [0] 0x56313fd28a00, type: 6, name: NULL
        [1] 0xd175b40000000013, type: 9, name: NULL
```

Debug Utilities Extension

- VK_EXT_debug_utils
  - Replaced original VK_EXT_debug_report/VK_EXT_debug_marker
- Implemented by Vulkan-ValidationLayers (and other tools)
- Provides the ability to attach user-defined names to
  - Vulkan Objects
  - Sequences of commands recorded in Command Buffers
  - Queue submissions
- Names show up in validation error messages
  - Also used by other tools such as RenderDoc
- Allows applications to register their own validation error handling callback

```
typedef struct VkDebugUtilsObjectNameInfoEXT {
    VkStructureType    sType;
    const void*        pNext;
    VkObjectType       objectType;
    uint64_t           objectHandle;
    const char*        pObjectName;
} VkDebugUtilsObjectNameInfoEXT;
```

```
VkResult vkSetDebugUtilsObjectNameEXT(
    VkDevice                                    device,
    const VkDebugUtilsObjectNameInfoEXT*        pNameInfo);
```

Debug Utilities Extension: Object naming

```
        err = vkCreateBuffer(demo->device, &buffer_create_info, NULL, &tex_obj->buffer);
        assert(!err);
        demo_name_object(demo, VK_OBJECT_TYPE_BUFFER, (uint64_t)tex_obj->buffer, "TexBuffer(%s)", filename);
```

- The demo_name_object() function
    - vsnprintf()'s the name into a buffer
    - Calls vkSetDebugUtilsObjectNameEXT()
    - Each object's name is stored in internal storage

```
Objects - 2
        Object[0] - VK_OBJECT_TYPE_COMMAND_BUFFER, Handle 0x5566702c9f60, Name "PrepareCB"
        Object[1] - VK_OBJECT_TYPE_BUFFER, Handle 0x9fde6b0000000014, Name "TexBuffer(lunarg.ppm)"
```

```
typedef struct VkDebugUtilsLabelEXT {
    VkStructureType    sType;
    const void*        pNext;
    const char*        pLabelName;
    float              color[4];
} VkDebugUtilsLabelEXT;
```

```
void vkCmdBeginDebugUtilsLabelEXT(
    VkCommandBuffer                                 commandBuffer,
    const VkDebugUtilsLabelEXT*                     pLabelInfo);
```

Debug Utilities extension: Command buffer labels

- Allows a name to be attached to a sequence of commands in a command buffer
- Stack-like, multiple labels can be present at once
  - `vkCmdBeginDebugUtilsLabelEXT()` pushes
  - `vkCmdEndDebugUtilsLabelEXT()` pops
- See also `vkQueueBeginDebugUtilsLabelEXT()`
- **Not printed by default error handler!**

```
Command Buffer Labels - 3
     Label[0] - StagingBufferCopy(0) { 0.000000, 0.000000, 0.000000, 0.000000}
     Label[1] - StagingTexture(0) { 0.000000, 0.000000, 0.000000, 0.000000}
     Label[2] - Prepare { 0.000000, 0.000000, 0.000000, 0.000000}
```

Debug Utilities extension: vkcube error callback

```
ERROR : VALIDATION - Message Id Number: 1867332608 | Message Id Name:
VUID-vkCmdCopyBufferToImage-pRegions-00171
      Validation Error: [ VUID-vkCmdCopyBufferToImage-pRegions-00171 ] Object 0: handle = 0x562780095ca0,
name = PrepareCB, type = VK_OBJECT_TYPE_COMMAND_BUFFER; Object 1: handle = 0x9fde6b0000000014, name =
TexBuffer type = VK_OBJECT_TYPE_BUFFER; | MessageID = 0x6f4d3c00 | vkCmdCopyBufferToImage: pRegion[0] is
trying to copy  523264 bytes plus 0 offset to/from the VkBuffer (VkBuffer
0x9fde6b0000000014[TexBuffer(lunarg.ppm)]) which exceeds the VkBuffer total size of 262144 bytes. The Vulkan
spec states: srcBuffer must be large enough to contain all buffer locations that are accessed according to
Buffer and Image Addressing, for each element of pRegions
(https://vulkan.lunarg.com/doc/view/1.3.243.0/windows/1.3-extensions/html/vkspec.html#VUID-vkCmdCopyBufferToI
mage-pRegions-00171)

      Objects - 2
            Object[0] - VK_OBJECT_TYPE_COMMAND_BUFFER, Handle 0x562780095ca0, Name "PrepareCB"
            Object[1] - VK_OBJECT_TYPE_BUFFER, Handle 0x9fde6b0000000014, Name "TexBuffer(lunarg.ppm)"

      Command Buffer Labels - 3
            Label[0] - StagingBufferCopy(0) { 0.000000, 0.000000, 0.000000, 0.000000}
            Label[1] - StagingTexture(0) { 0.000000, 0.000000, 0.000000, 0.000000}
            Label[2] - Prepare { 0.000000, 0.000000, 0.000000, 0.000000}
```

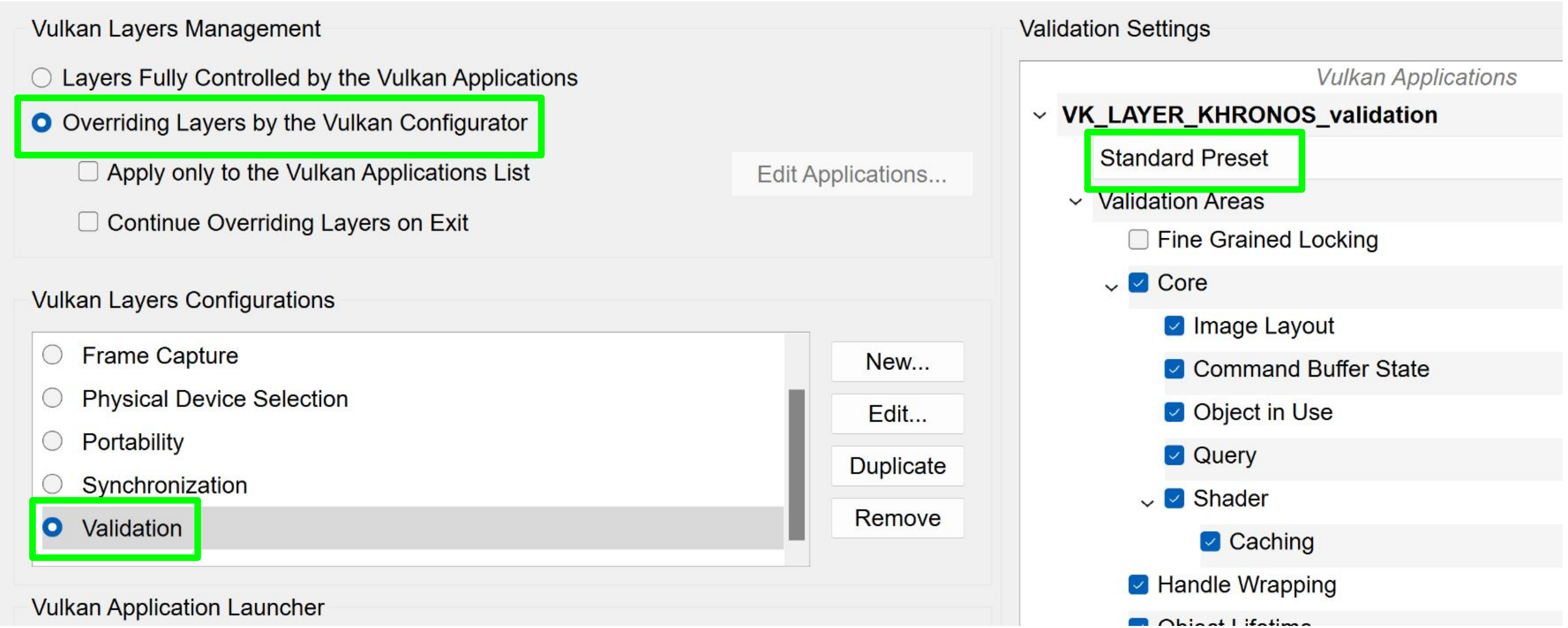
Debug Utilities extension: Custom message callback

- Set up by calling `vkCreateDebugUtilsMessengerEXT()`
  - Your callback receives a complex struct for each error
  - Same mechanism used for default error logging
- Possible uses
  - Make your own message format
  - Add messages to application logging stream
  - Send messages to somewhere other than the console
  - Trigger failures in your unit test framework
- **Don’t use it to filter messages**, it is faster to use Validation Layer’s the built in filtering

Validation Quick Start - Enable

- Run the Vulkan Configurator (Simplest)
  - With SDK installed you should have a **Vulkan Configurator** program under the start menu
  - Or run **vkconfig** from the command line
- At **vkCreateInstance()** time
  - Add the layer name to `VkInstanceCreateInfo::ppEnabledLayerNames`
- From the terminal
  - `export VK_INSTANCE_LAYERS=VK_LAYER_KHRONOS_validation ./your-application`

Vulkan Configurator


Vulkan Layers Management

( ) Layers Fully Controlled by the Vulkan Applications

(●) Overriding Layers by the Vulkan Configurator

☐ Apply only to the Vulkan Applications List

☐ Continue Overriding Layers on Exit

Vulkan Layers Configurations

( ) Frame Capture
( ) Physical Device Selection
( ) Portability
( ) Synchronization
(●) Validation

Vulkan Application Launcher

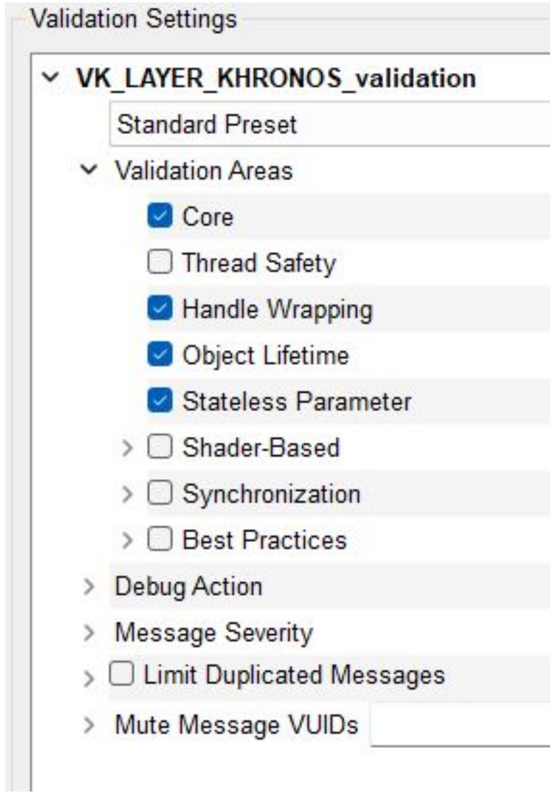
Validation Settings

Vulkan Applications

VK_LAYER_KHRONOS_validation
- Standard Preset
- Validation Areas
  - ☐ Fine Grained Locking
  - ☑ Core
    - ☑ Image Layout
    - ☑ Command Buffer State
    - ☑ Object in Use
    - ☑ Query
    - ☑ Shader
      - ☑ Caching
  - ☑ Handle Wrapping

Configuration - How to set

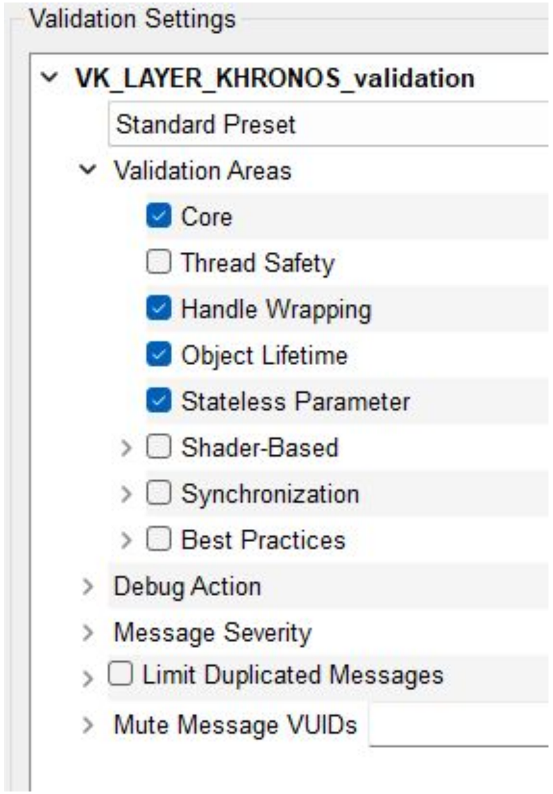
- Right pane in vkconfig
- Can use vk_layer_settings.txt
    - khronos_validation.enables
    - khronos_validation.disables
- Environment variables
    - VK_LAYER_ENABLES
    - VK_LAYER_DISABLES
- VK_EXT_validation_features
    - Set at VkDevice creation time
- https://vulkan.lunarg.com/doc/sdk/latest/windows/khronos_validation_layer.html


Validation Settings

VK_LAYER_KHRONOS_validation
- Standard Preset
- Validation Areas
    - [✓] Core
    - [ ] Thread Safety
    - [✓] Handle Wrapping
    - [✓] Object Lifetime
    - [✓] Stateless Parameter
    - [ ] Shader-Based
    - [ ] Synchronization
    - [ ] Best Practices
- Debug Action
- Message Severity
- [ ] Limit Duplicated Messages
- Mute Message VUIDs

Configuration - presets and areas

- Validation is split up into several areas to reduce performance overhead
- Don’t enable all areas at once (it will be slow!)
- Use the available presets!
- Fix errors from each preset,
  - Then run Standard preset again

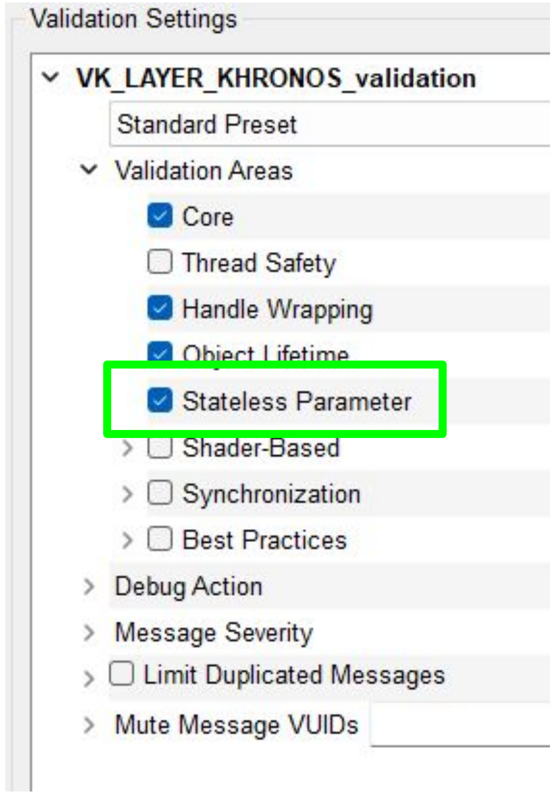


Validation Settings

- VK_LAYER_KHRONOS_validation
  - Standard Preset
  - Validation Areas
    - [✓] Core
    - [ ] Thread Safety
    - [✓] Handle Wrapping
    - [✓] Object Lifetime
    - [✓] Stateless Parameter
    - [ ] Shader-Based
    - [ ] Synchronization
    - [ ] Best Practices
  - Debug Action
  - Message Severity
  - [ ] Limit Duplicated Messages
  - Mute Message VUIDs

Configuration: Stateless

- Checks implicit and other simple VUIDs
- Lots of generated checks
- Doesn't require expensive state tracking - fast

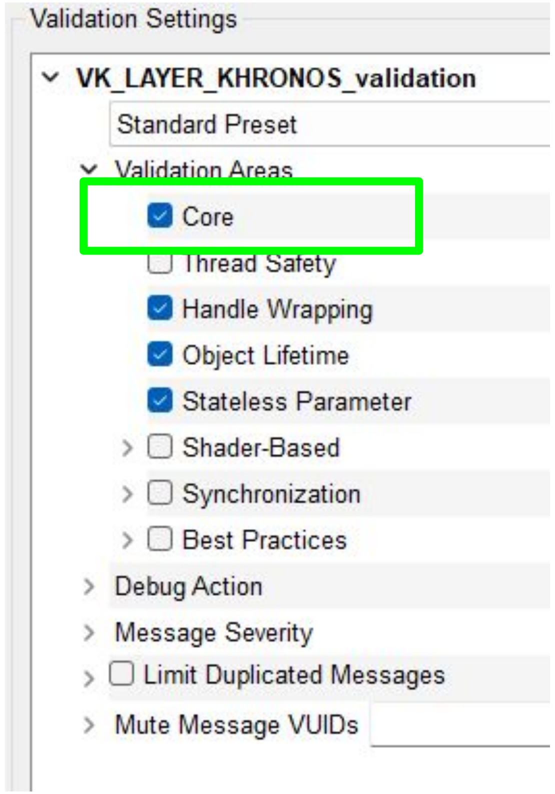


Validation Settings

- VK_LAYER_KHRONOS_validation
  - Standard Preset
  - Validation Areas
    - [✓] Core
    - [ ] Thread Safety
    - [✓] Handle Wrapping
    - [✓] Object Lifetime
    - [✓] Stateless Parameter
    - [ ] Shader-Based
    - [ ] Synchronization
    - [ ] Best Practices
  - Debug Action
  - Message Severity
  - [ ] Limit Duplicated Messages
  - Mute Message VUIDs

Configuration: Core

- Most VUIDs checked here
- Requires state tracking - slower



Validation Settings

VK_LAYER_KHRONOS_validation

Standard Preset

Validation Areas

- [x] Core
- [ ] Thread Safety
- [x] Handle Wrapping
- [x] Object Lifetime
- [x] Stateless Parameter
- [ ] Shader-Based
- [ ] Synchronization
- [ ] Best Practices

Debug Action

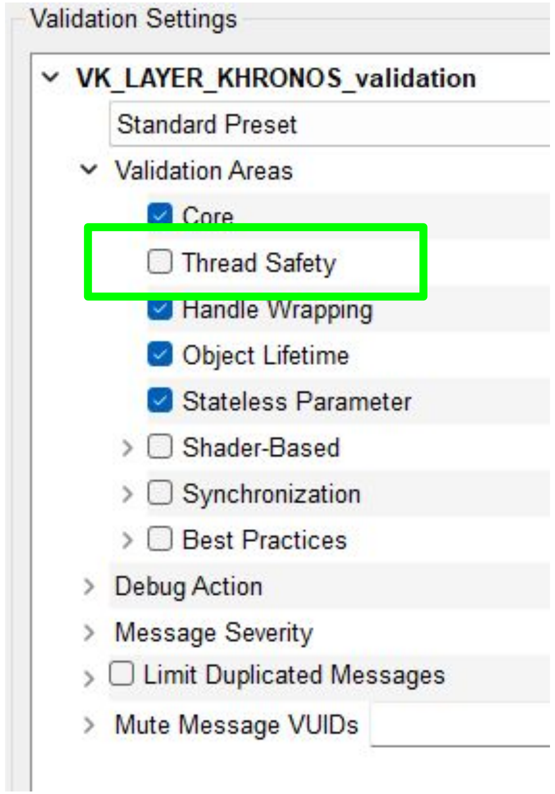
Message Severity

- [ ] Limit Duplicated Messages

Mute Message VUIDs

Configuration: Thread Safety

- Checks external synchronization requirements
- Accessing a vulkan object from multiple threads concurrently



Validation Settings

VK_LAYER_KHRONOS_validation

Standard Preset

Validation Areas

- [x] Core
- [ ] Thread Safety
- [x] Handle Wrapping
- [x] Object Lifetime
- [x] Stateless Parameter
- [ ] Shader-Based
- [ ] Synchronization
- [ ] Best Practices

Debug Action

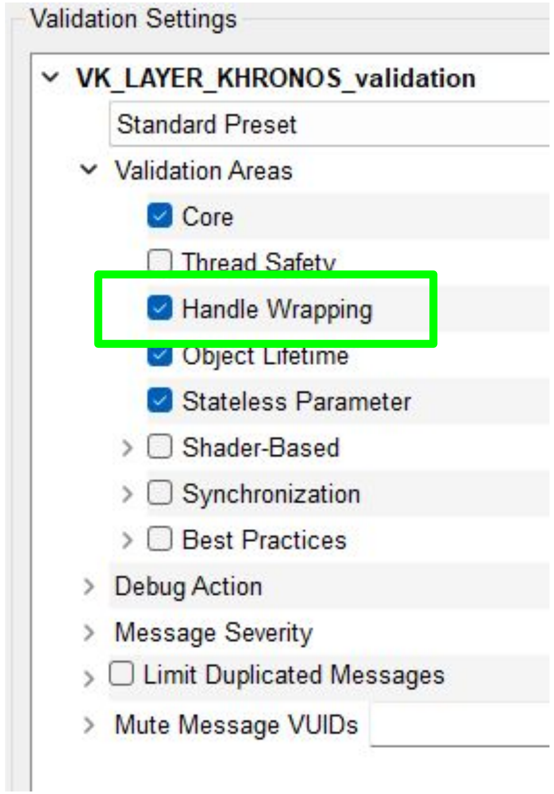
Message Severity

- [ ] Limit Duplicated Messages

Mute Message VUIDs

Configuration: Handle Wrapping

- Prevents handle reuse bugs

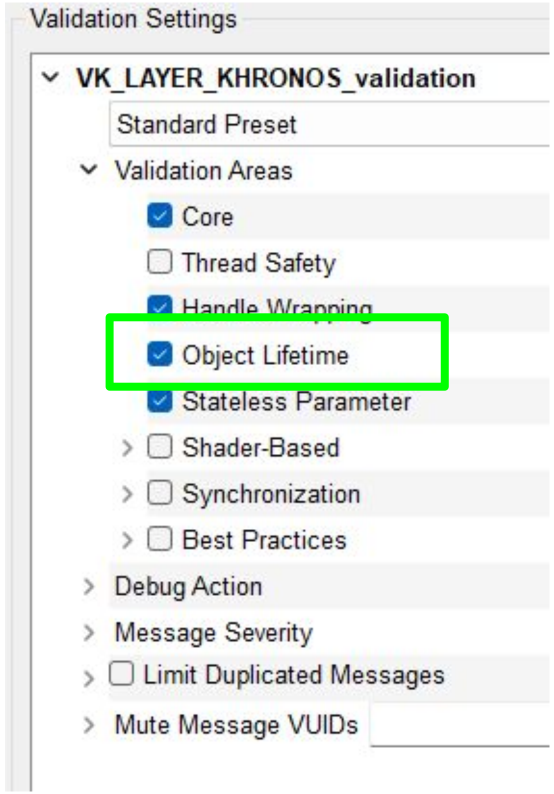


Validation Settings

- VK_LAYER_KHRONOS_validation
  - Standard Preset
  - Validation Areas
    - [✓] Core
    - [ ] Thread Safety
    - [✓] Handle Wrapping
    - [✓] Object Lifetime
    - [✓] Stateless Parameter
    - [ ] Shader-Based
    - [ ] Synchronization
    - [ ] Best Practices
  - Debug Action
  - Message Severity
  - [ ] Limit Duplicated Messages
  - Mute Message VUIDs

Configuration: Object Lifetime

- Detects use of destroyed objects

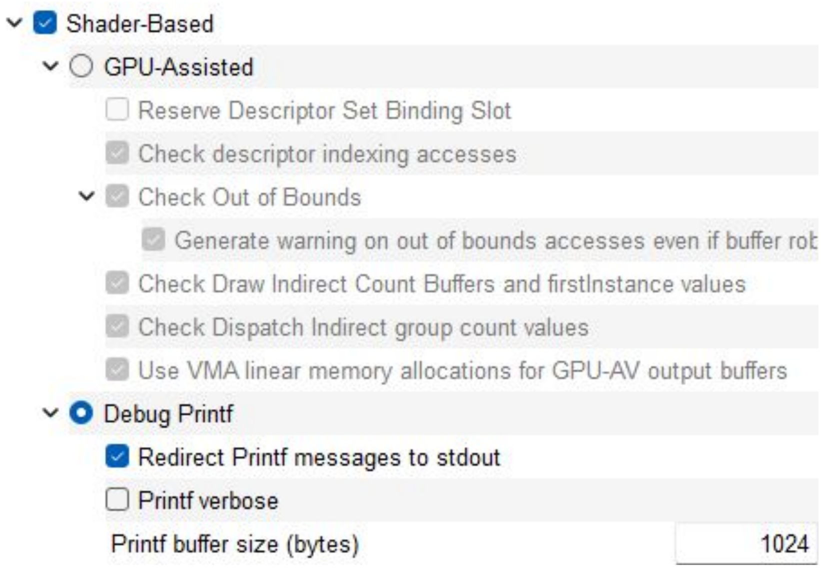


Validation Settings

- VK_LAYER_KHRONOS_validation
  - Standard Preset
  - Validation Areas
    - [✓] Core
    - [ ] Thread Safety
    - [✓] Handle Wrapping
    - [✓] Object Lifetime
    - [✓] Stateless Parameter
    - [ ] Shader-Based
    - [ ] Synchronization
    - [ ] Best Practices
  - Debug Action
  - Message Severity
  - [ ] Limit Duplicated Messages
  - Mute Message VUIDs

Configuration: Shader Based

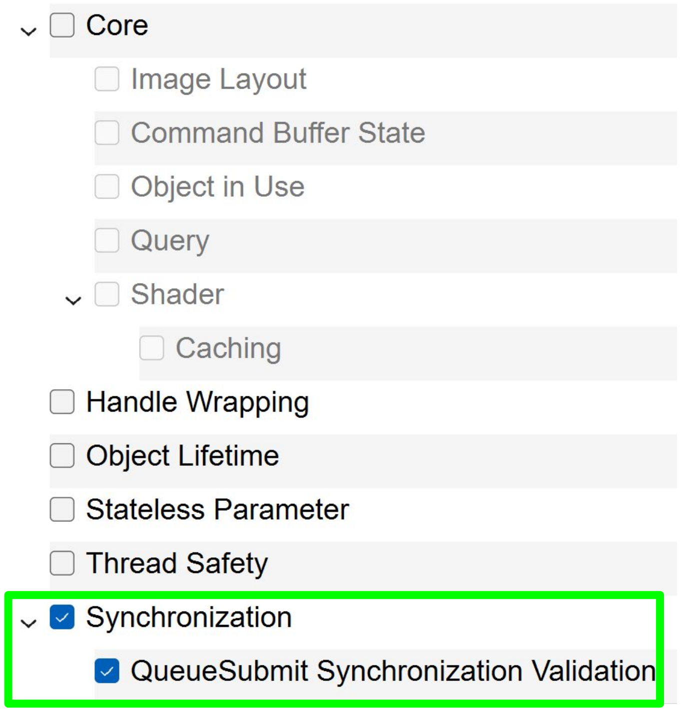
- GPU-Assisted
  - AKA: GPU-AV
  - Instruments SPIR-V to detect problems in shaders
  - Descriptor indexing
  - Buffer Device Address
  - Not supported on Mac
- DebugPrintf
  - Adds printf() functionality to shaders
  - Not supported on Mac



- [✓] Shader-Based
  - ○ GPU-Assisted
    - [ ] Reserve Descriptor Set Binding Slot
    - [✓] Check descriptor indexing accesses
    - [✓] Check Out of Bounds
      - [✓] Generate warning on out of bounds accesses even if buffer rot
    - [✓] Check Draw Indirect Count Buffers and firstInstance values
    - [✓] Check Dispatch Indirect group count values
    - [✓] Use VMA linear memory allocations for GPU-AV output buffers
  - ● Debug Printf
    - [✓] Redirect Printf messages to stdout
    - [ ] Printf verbose
    - Printf buffer size (bytes): 1024

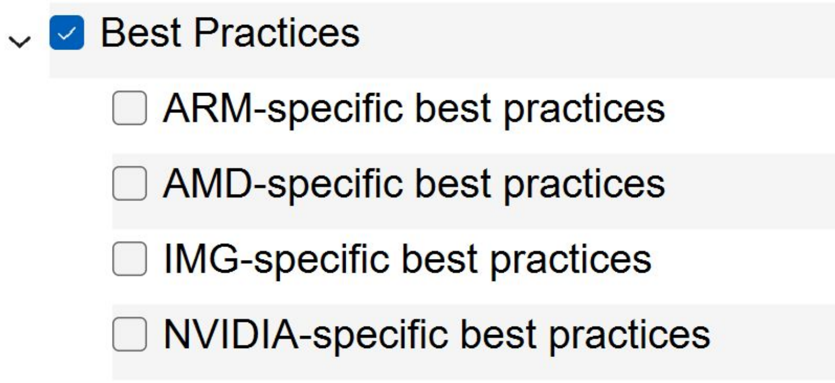
Configuration: Synchronization

- Checks for correct Execution and Memory Dependencies
- vkCmdPipelineBarrier(), VkEvents, etc.


- [ ] Core
  - [ ] Image Layout
  - [ ] Command Buffer State
  - [ ] Object in Use
  - [ ] Query
  - [ ] Shader
    - [ ] Caching
- [ ] Handle Wrapping
- [ ] Object Lifetime
- [ ] Stateless Parameter
- [ ] Thread Safety
- [✓] Synchronization
  - [✓] QueueSubmit Synchronization Validation

Configuration: Best Practice

- Detects Valid but dubious behavior
  - Performance warnings
  - Undefined values
  - Non-success return values
- Mixture of common and vendor-specific checks

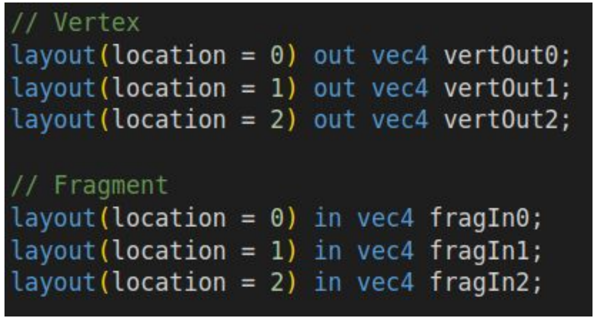


- [✓] Best Practices
  - [ ] ARM-specific best practices
  - [ ] AMD-specific best practices
  - [ ] IMG-specific best practices
  - [ ] NVIDIA-specific best practices

Best Practices example: Undefined Value

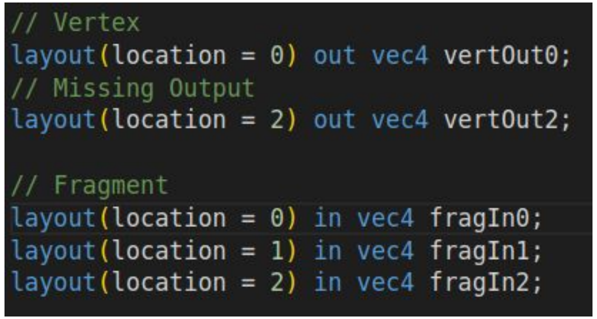
- Undefined **Value** != Undefined **Behavior**
- The app will never crash
- Your data might be garbage
- Great use of Best Practices layers

Undefined Behavior vs Best Practice



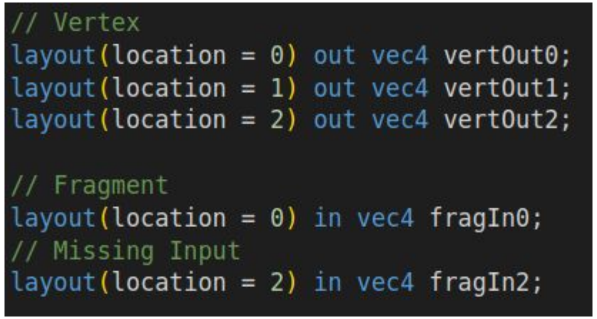
```
// Vertex
layout(location = 0) out vec4 vertOut0;
layout(location = 1) out vec4 vertOut1;
layout(location = 2) out vec4 vertOut2;

// Fragment
layout(location = 0) in vec4 fragIn0;
layout(location = 1) in vec4 fragIn1;
layout(location = 2) in vec4 fragIn2;
```



```
// Vertex
layout(location = 0) out vec4 vertOut0;
// Missing Output
layout(location = 2) out vec4 vertOut2;

// Fragment
layout(location = 0) in vec4 fragIn0;
layout(location = 1) in vec4 fragIn1;
layout(location = 2) in vec4 fragIn2;
```



```
// Vertex
layout(location = 0) out vec4 vertOut0;
layout(location = 1) out vec4 vertOut1;
layout(location = 2) out vec4 vertOut2;

// Fragment
layout(location = 0) in vec4 fragIn0;
// Missing Input
layout(location = 2) in vec4 fragIn2;
```

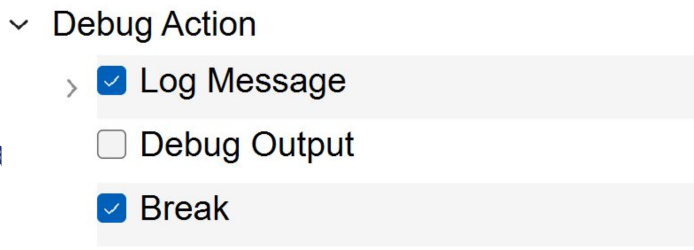
Normal

Error

Valid
But is this what you wanted?

Configuration: Break on error

- Will stop program when an error is detected
  - Calls `DebugBreak();` or `raise(SIGTRAP);`
- Stack trace will **usually** take you to the part of your code ca the error
- But some errors are not detected until queue submission time
  - Examples: Image Layout, Sync Validation, Timeline Semaphores
  - Stack trace will take you to the queue submission code

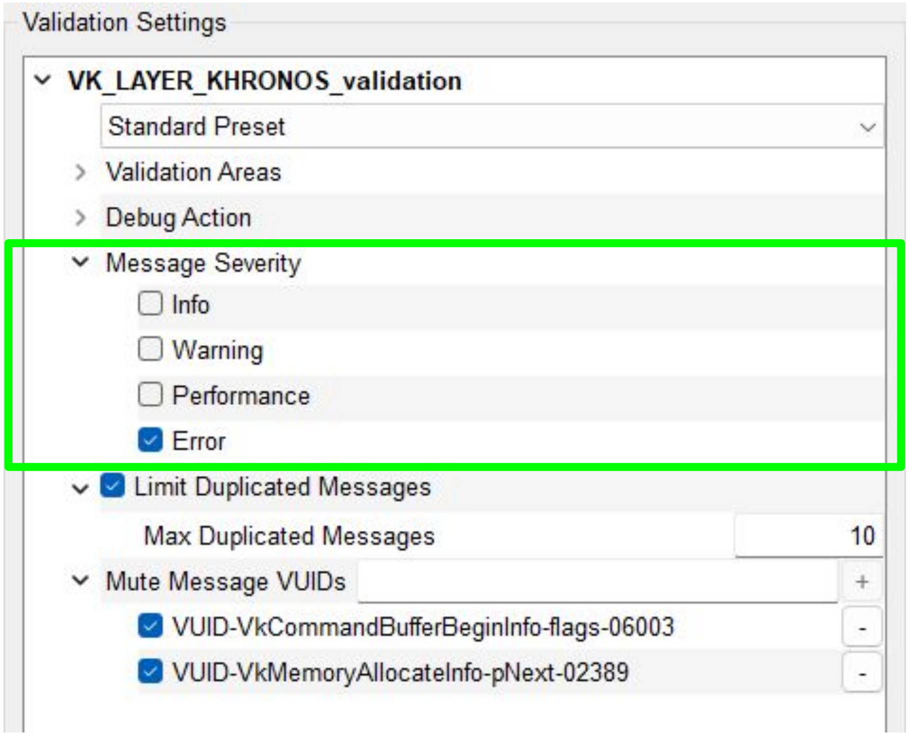


Debug Action

- [x] Log Message
- [ ] Debug Output
- [x] Break

Configuration: Limit message severity

- Almost all messages are ‘Error’
- Except Best Practices, which is ‘Performance’ and ‘Warning’

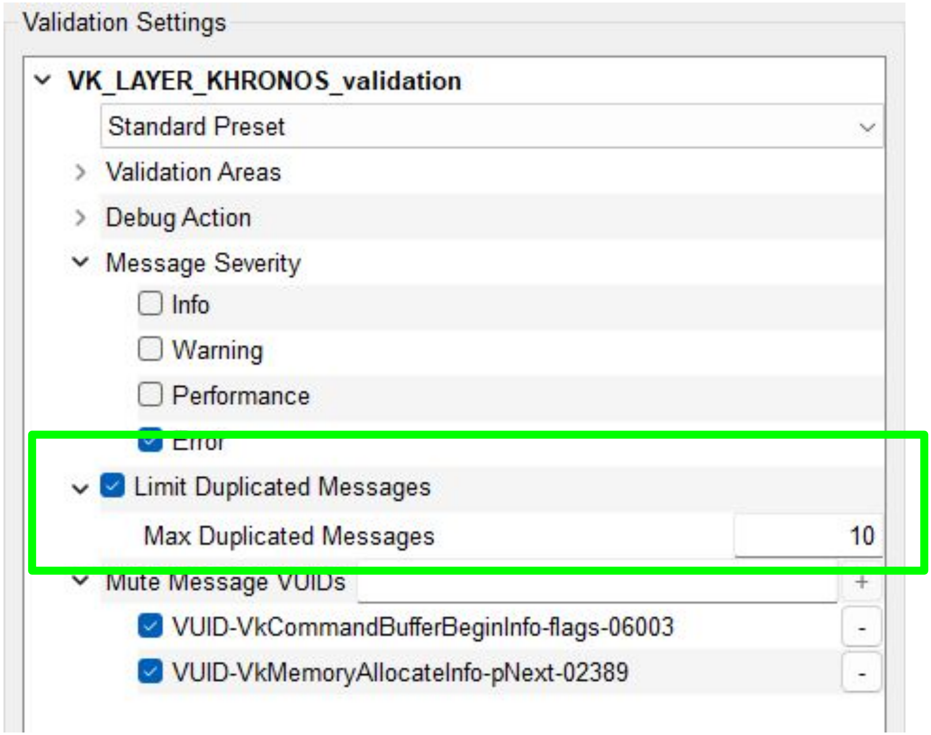


Validation Settings

- VK_LAYER_KHRONOS_validation
  - Standard Preset
  - Validation Areas
  - Debug Action
  - Message Severity
    - [ ] Info
    - [ ] Warning
    - [ ] Performance
    - [✓] Error
  - [✓] Limit Duplicated Messages
    - Max Duplicated Messages: 10
  - Mute Message VUIDs
    - [✓] VUID-VkCommandBufferBeginInfo-flags-06003
    - [✓] VUID-VkMemoryAllocateInfo-pNext-02389

Configuration: Limit repeated messages

- Limit times a message is repeated
  - Exact VUID string must match to count as a repeat



Validation Settings

VK_LAYER_KHRONOS_validation

Standard Preset

Validation Areas

Debug Action

Message Severity

- [ ] Info
- [ ] Warning
- [ ] Performance
- [✓] Error

[✓] Limit Duplicated Messages

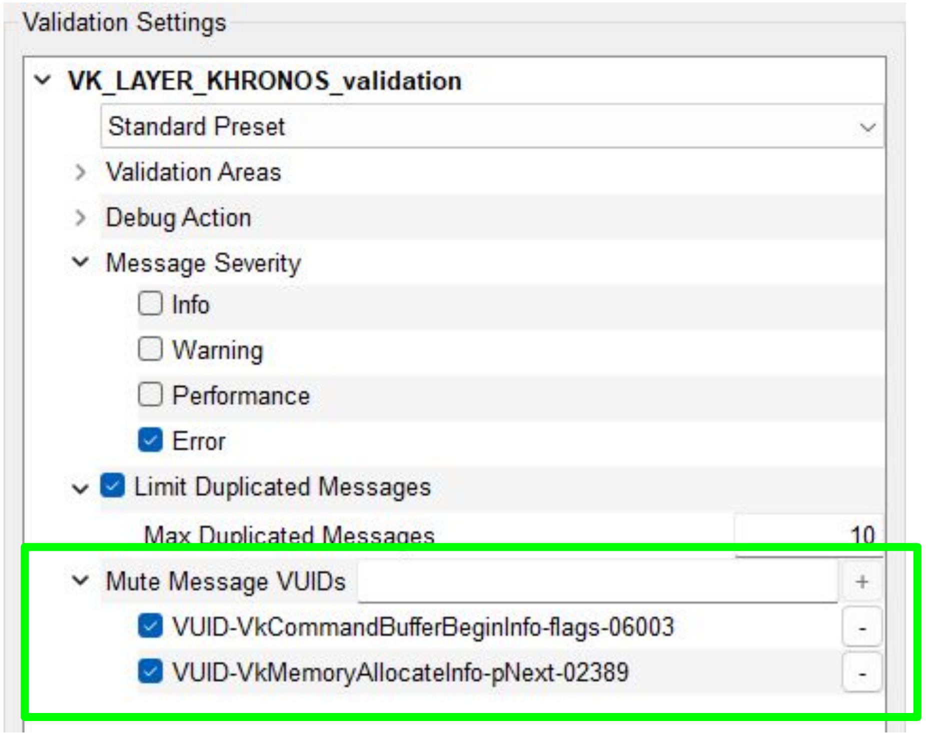
Max Duplicated Messages: 10

Mute Message VUIDs

- [✓] VUID-VkCommandBufferBeginInfo-flags-06003
- [✓] VUID-VkMemoryAllocateInfo-pNext-02389

Configuration: Mute message

- Sometimes undefined behaviour works
- Sometimes the Validation Layers have bugs
- Sometimes the Vulkan Spec has bugs

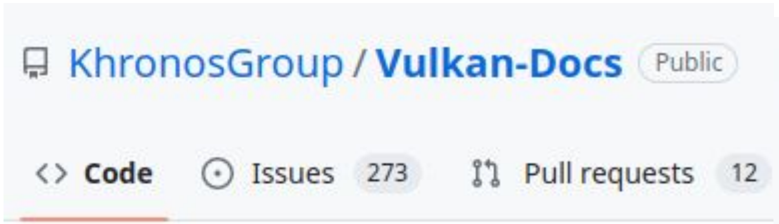


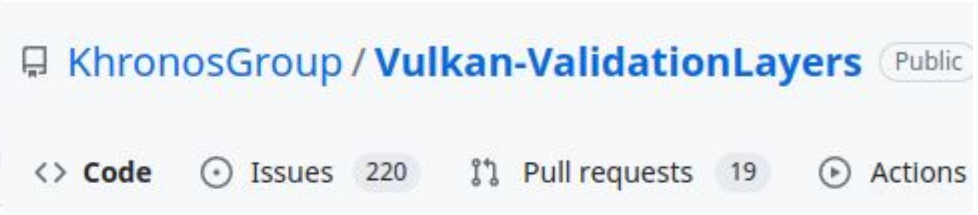
Validation Settings

- VK_LAYER_KHRONOS_validation
  - Standard Preset
  - Validation Areas
  - Debug Action
  - Message Severity
    - [ ] Info
    - [ ] Warning
    - [ ] Performance
    - [✓] Error
  - [✓] Limit Duplicated Messages
    - Max Duplicated Messages: 10
  - Mute Message VUIDs
    - [✓] VUID-VkCommandBufferBeginInfo-flags-06003
    - [✓] VUID-VkMemoryAllocateInfo-pNext-02389

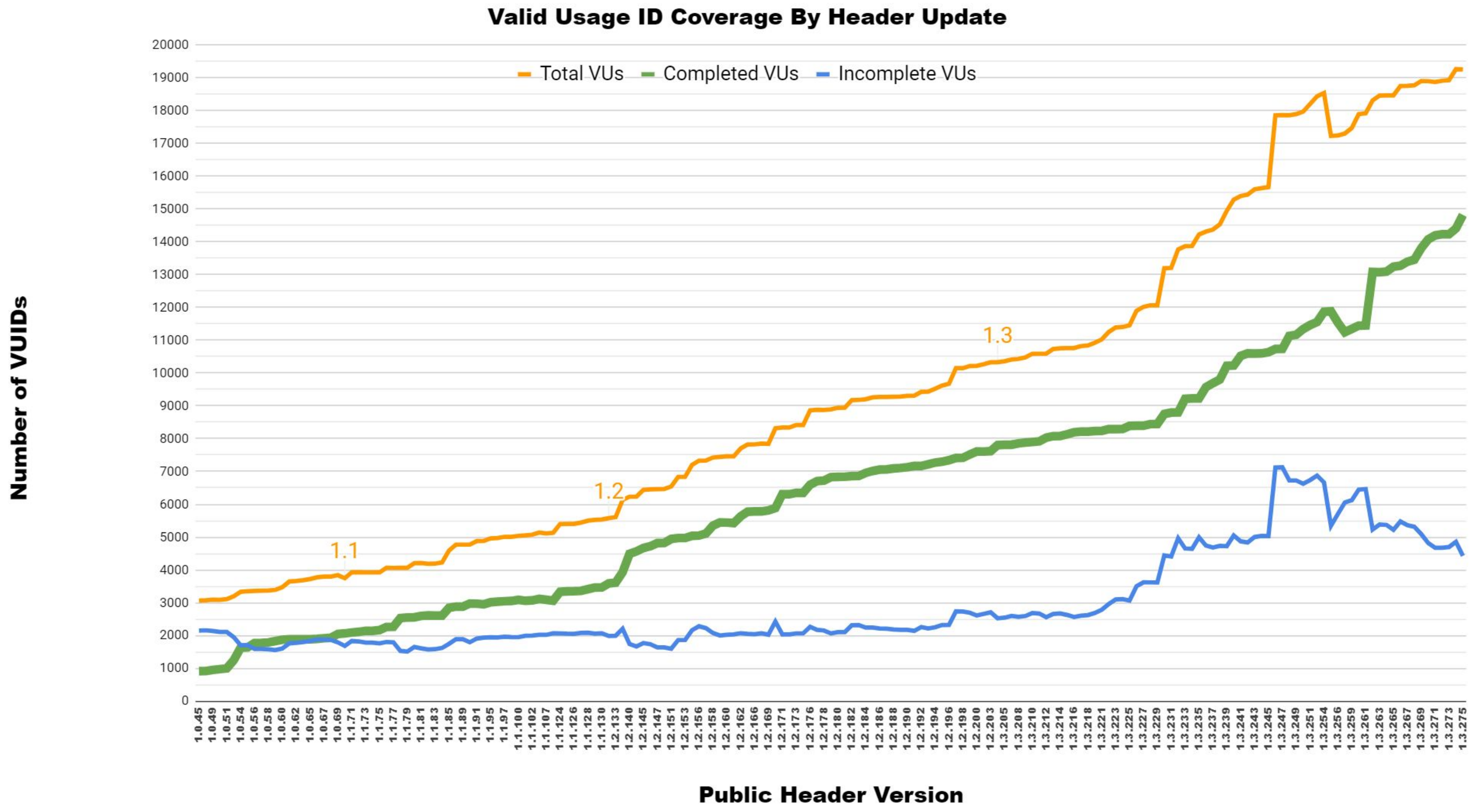
Is this really an error?

- **Advice:**
  - Search in the ValidationLayer source for the VUID string to see how it is validated
  - Check Khronos Slack, Discord, Reddit, etc.
  - Disable implicit layers, which could cause errors
- Could be a bug in validation or the spec, please report it!
- If not sure which to choose, feel free to put in Validation repo



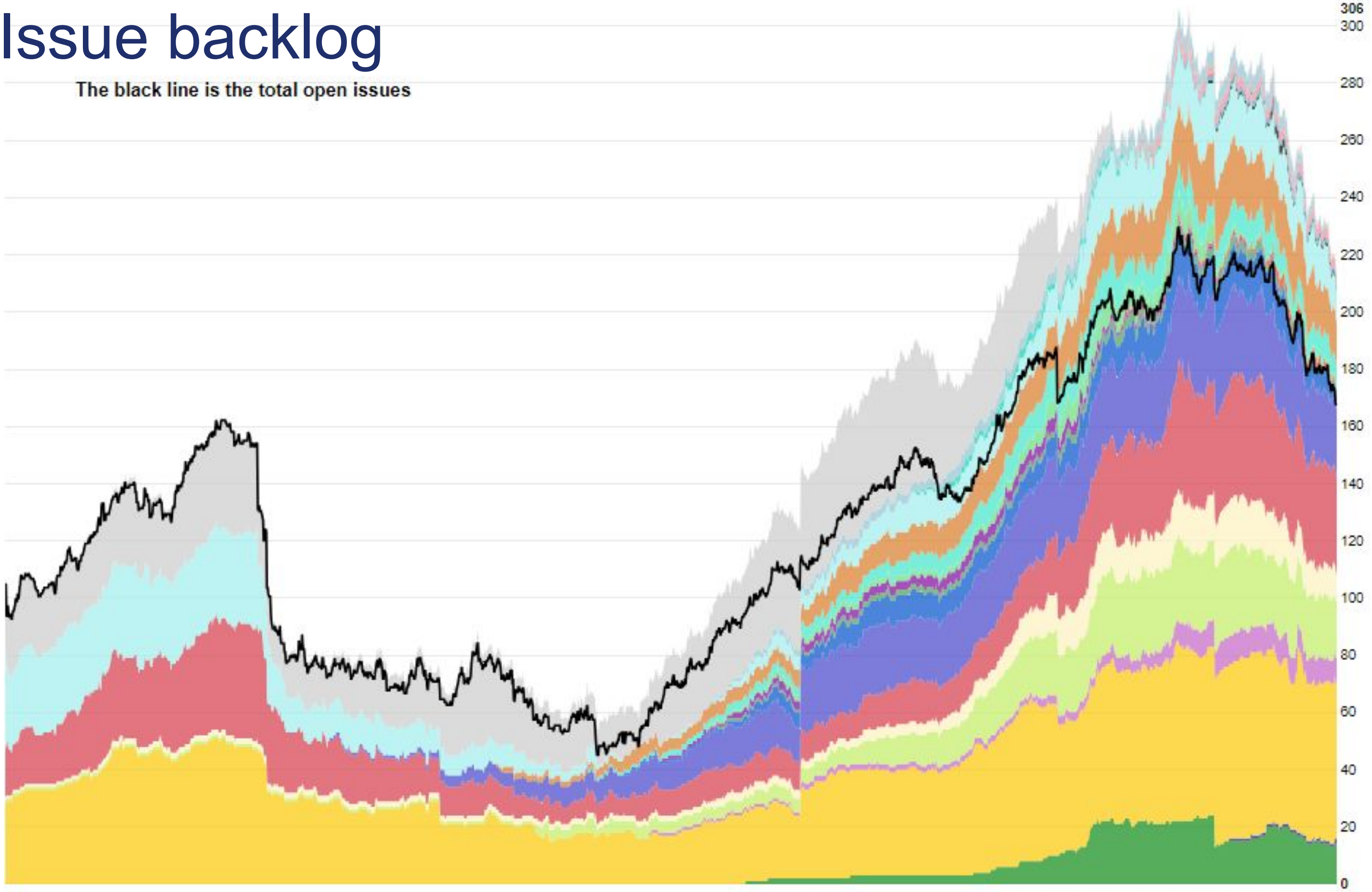


Not all VUIDs checked



Issue backlog

The black line is the total open issues



Recent Improvements (last 12 months)

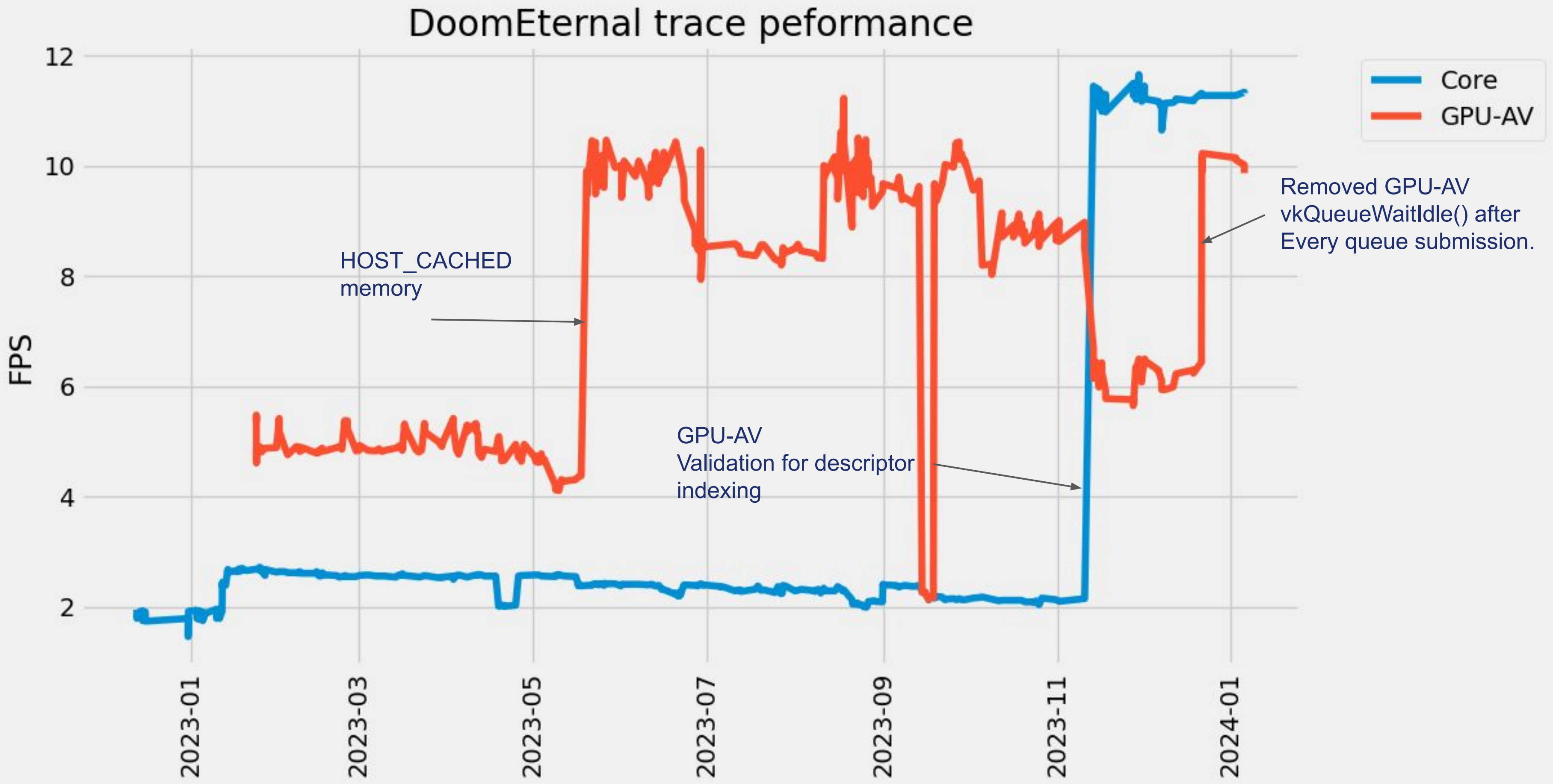
- Improved consistency and detail of all existing error messages
- GPU-AV descriptor indexing validation
- Sync Validation at Queue submission time
- Improved support for timeline semaphores, queue present operations, external memory
- Vulkan Utilities Libraries (commonly used parts of VVL codebase)
  - Utility headers such as vk_format_utils.h
  - Layer Settings library

GPU-AV descriptor indexing validation

"A descriptor is dynamically used if any shader invocation executes an instruction that performs any memory access using the descriptor. If a descriptor is not dynamically used, any resource referenced by the descriptor is not considered to be referenced during command execution."

- Bindless applications have huge arrays of descriptors
  - But… only a few descriptors are used by each shader invocation
- GPU-AV has instrumentation to track which descriptors are used
  - CPU code then validates only this subset
- Improves performance and removes false positives from unused descriptors

Validation Layer Performance Improvements

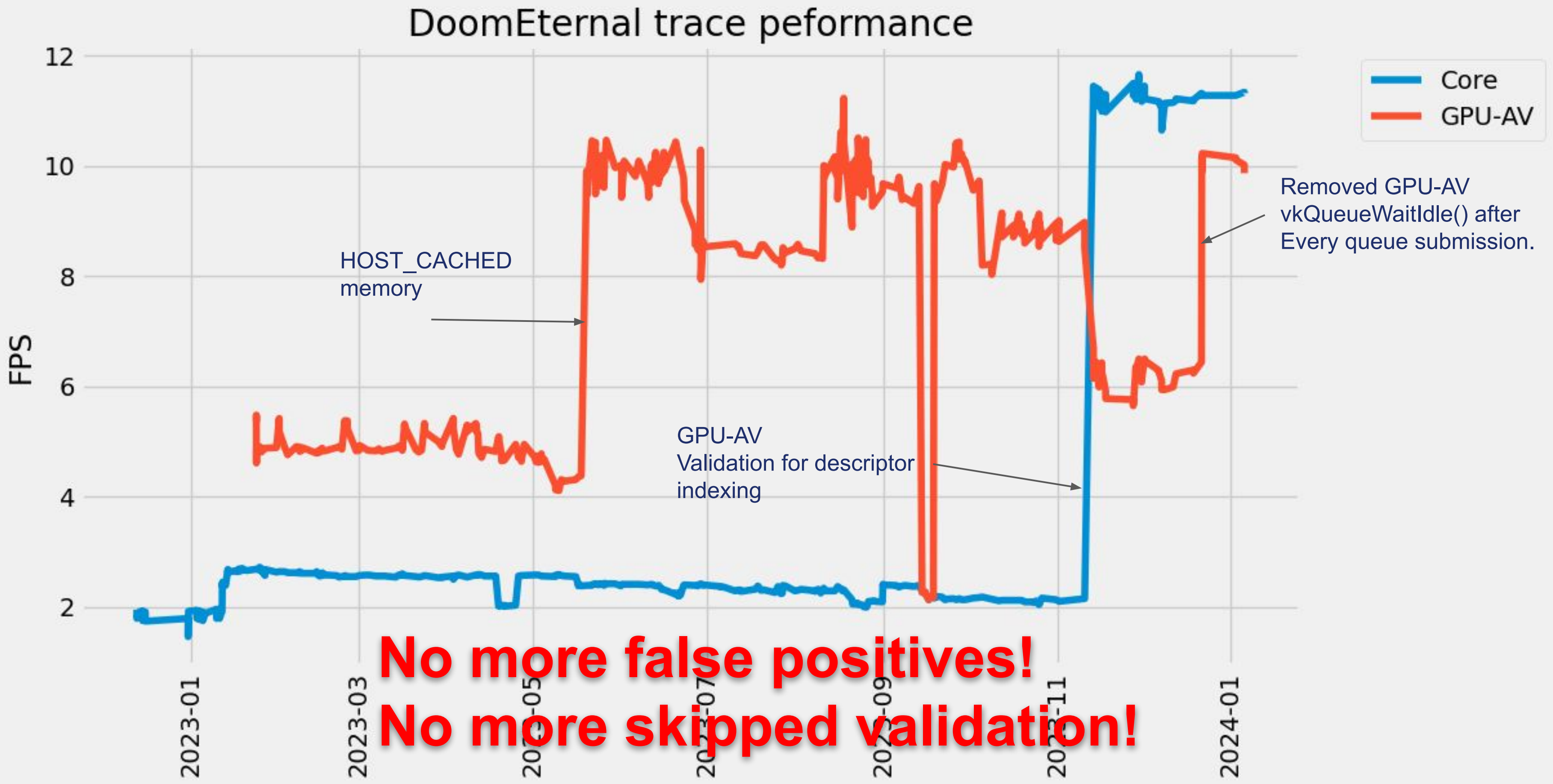


HOST_CACHED memory

GPU-AV Validation for descriptor indexing

Removed GPU-AV vkQueueWaitIdle() after Every queue submission.

Validation Layer Performance Improvements



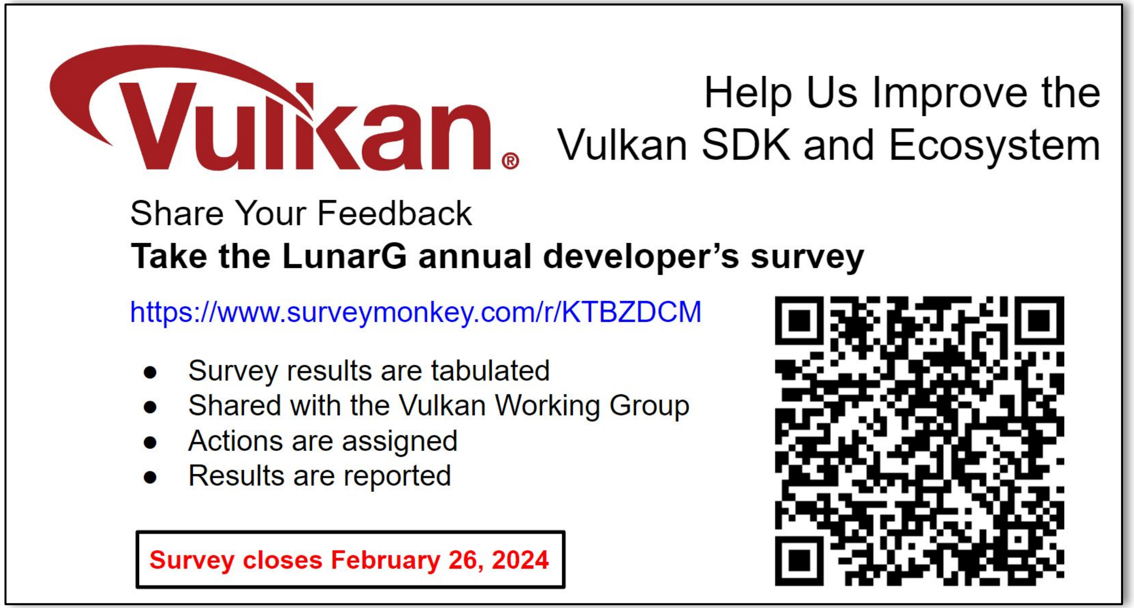
No more false positives!
No more skipped validation!

Upcoming Improvements

- More GPU-AV work
    - Ray tracing
    - Descriptor buffers
- Sync validation performance optimization
- Improve debuggability of errors detected during queue submission
    - Finding which command caused an error of this type can be difficult
- SPIR-V runtime validation improvements
- Further work on error message formatting
- Again, please submit an Issue on github if we're missing something you need!
    - We also accept Pull Requests :)

Summary

- Vulkan is complex and there are many rules for you to follow
- The VUID system and Validation Layer help you deal with these rules
- The Debug Utilties extension can also help you find the source of errors
- The Vulkan Configurator is an easy way to configure validation
- The Validation Layer isn’t perfect but we’re always working to make it better



Vulkan®

Help Us Improve the Vulkan SDK and Ecosystem

Share Your Feedback

Take the LunarG annual developer's survey

https://www.surveymonkey.com/r/KTBZDCM

- Survey results are tabulated
- Shared with the Vulkan Working Group
- Actions are assigned
- Results are reported

Survey closes February 26, 2024

Today's Presentation:

https://bit.ly/48Wb5sL

bitly

Thank you!

QUESTIONS?

Get A FREE Tumbler at the LunarG Sponsor Table!

LUNARG