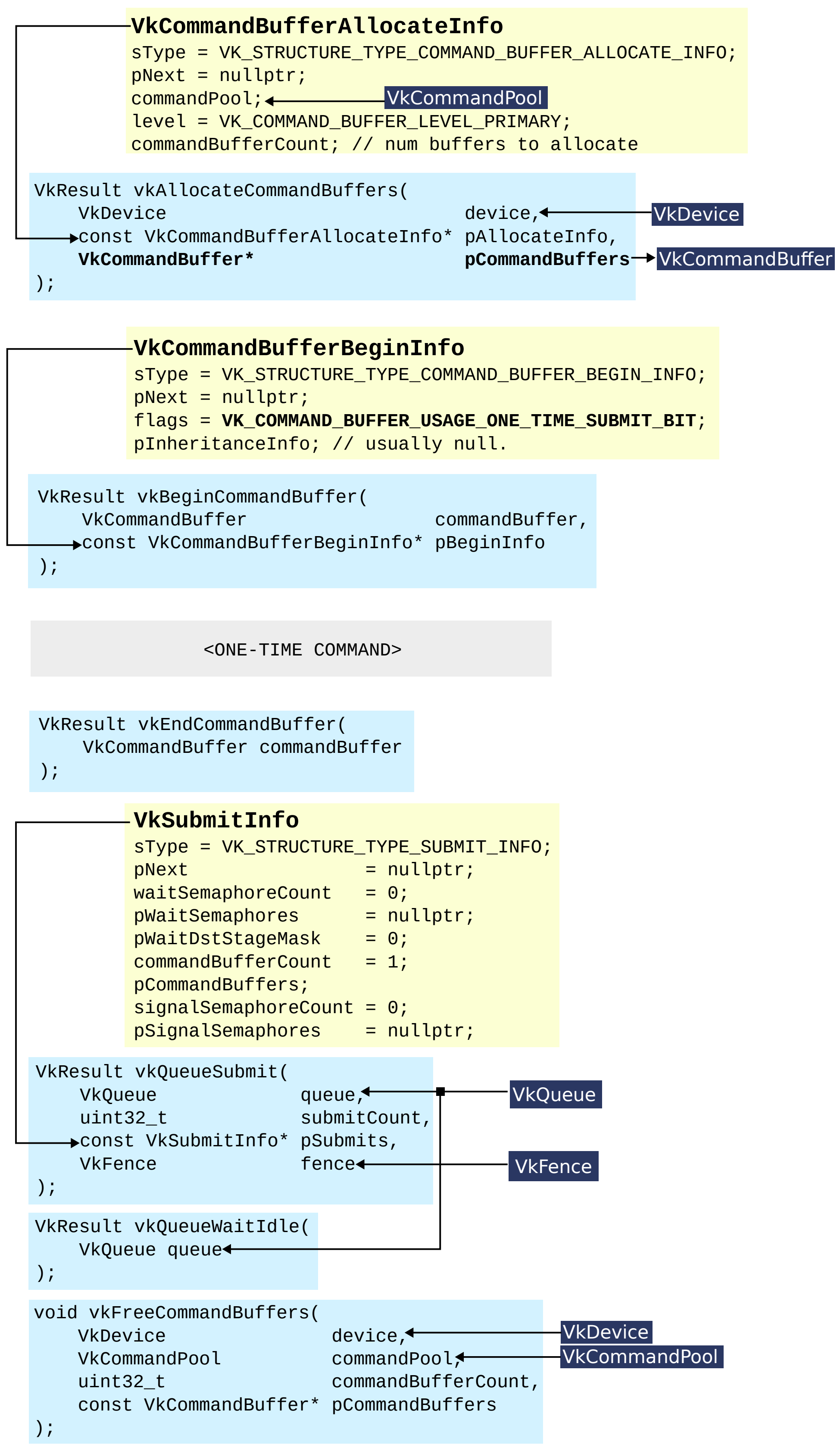
VkCommandBuffer for a One-Time Command



```
VkCommandBufferAllocateInfo
sType = VK_STRUCTURE_TYPE_COMMAND_BUFFER_ALLOCATE_INFO;
pNext = nullptr;
commandPool;
level = VK_COMMAND_BUFFER_LEVEL_PRIMARY;
commandBufferCount; // num buffers to allocate
```

```
VkResult vkAllocateCommandBuffers(
    VkDevice                           device,
    const VkCommandBufferAllocateInfo* pAllocateInfo,
    VkCommandBuffer*                   pCommandBuffers
);
```

```
VkCommandBufferBeginInfo
sType = VK_STRUCTURE_TYPE_COMMAND_BUFFER_BEGIN_INFO;
pNext = nullptr;
flags = VK_COMMAND_BUFFER_USAGE_ONE_TIME_SUBMIT_BIT;
pInheritanceInfo; // usually null.
```

```
VkResult vkBeginCommandBuffer(
    VkCommandBuffer                 commandBuffer,
    const VkCommandBufferBeginInfo* pBeginInfo
);
```

```
<ONE-TIME COMMAND>
```

```
VkResult vkEndCommandBuffer(
    VkCommandBuffer commandBuffer
);
```

```
VkSubmitInfo
sType = VK_STRUCTURE_TYPE_SUBMIT_INFO;
pNext                = nullptr;
waitSemaphoreCount   = 0;
pWaitSemaphores      = nullptr;
pWaitDstStageMask    = 0;
commandBufferCount   = 1;
pCommandBuffers;
signalSemaphoreCount = 0;
pSignalSemaphores    = nullptr;
```

```
VkResult vkQueueSubmit(
    VkQueue             queue,
    uint32_t            submitCount,
    const VkSubmitInfo* pSubmits,
    VkFence             fence
);
```

```
VkResult vkQueueWaitIdle(
    VkQueue queue
);
```

```
void vkFreeCommandBuffers(
    VkDevice               device,
    VkCommandPool          commandPool,
    uint32_t               commandBufferCount,
    const VkCommandBuffer* pCommandBuffers
);
```