vkGetPhysicalDeviceSurfaceSupportKHR()
vkGetPhysicalDeviceSurfaceCapabilitiesKHR()
vkGetPhysicalDeviceSurfacePresentModesKHR()
vkGetPhysicalDeviceSurfaceFormatsKHR()
for VkSwapchainKHR

```
VkResult vkGetPhysicalDeviceSurfaceSupportKHR(
    VkPhysicalDevice physicalDevice,
    uint32_t         queueFamilyIndex,
    VkSurfaceKHR     surface,
    VkBool32*        pSupported
);
```

VkPhysicalDevice

VkSurfaceKHR

```
VkResult vkGetPhysicalDeviceSurfaceCapabilitiesKHR(
    VkPhysicalDevice          physicalDevice,
    VkSurfaceKHR              surface,
    VkSurfaceCapabilitiesKHR* pSurfaceCapabilities
);
```

VkPhysicalDevice

VkSurfaceKHR

```
typedef struct VkSurfaceCapabilitiesKHR {
minImageCount; // Ex. 2
maxImageCount; // Ex. 8
currentExtent;  // Ex. (800, 600)
minImageExtent; // Ex. (800, 600)
maxImageExtent; // Ex. (800, 600)
maxImageArrayLayers; // Ex. 1
supportedTransforms;
currentTransform;
supportedCompositeAlpha;
supportedUsageFlags;
} VkSurfaceCapabilitiesKHR;
```

```
typedef enum VkSurfaceTransformFlagBitsKHR {
    VK_SURFACE_TRANSFORM_IDENTITY_BIT_KHR = 0x00000001,
    VK_SURFACE_TRANSFORM_ROTATE_90_BIT_KHR = 0x00000002,
    VK_SURFACE_TRANSFORM_ROTATE_180_BIT_KHR = 0x00000004,
    VK_SURFACE_TRANSFORM_ROTATE_270_BIT_KHR = 0x00000008,
    VK_SURFACE_TRANSFORM_HORIZONTAL_MIRROR_BIT_KHR = 0x00000010,
    VK_SURFACE_TRANSFORM_HORIZONTAL_MIRROR_ROTATE_90_BIT_KHR,
    VK_SURFACE_TRANSFORM_HORIZONTAL_MIRROR_ROTATE_180_BIT_KHR,
    VK_SURFACE_TRANSFORM_HORIZONTAL_MIRROR_ROTATE_270_BIT_KHR,
    VK_SURFACE_TRANSFORM_INHERIT_BIT_KHR = 0x00000100,
} VkSurfaceTransformFlagBitsKHR;
```

```
typedef enum VkCompositeAlphaFlagBitsKHR {
    VK_COMPOSITE_ALPHA_OPAQUE_BIT_KHR = 0x00000001,
    VK_COMPOSITE_ALPHA_PRE_MULTIPLIED_BIT_KHR = 0x00000002,
    VK_COMPOSITE_ALPHA_POST_MULTIPLIED_BIT_KHR = 0x00000004,
    VK_COMPOSITE_ALPHA_INHERIT_BIT_KHR = 0x00000008,
} VkCompositeAlphaFlagBitsKHR;
```

```
typedef enum VkImageUsageFlagBits {
    VK_IMAGE_USAGE_TRANSFER_SRC_BIT = 0x00000001,
    VK_IMAGE_USAGE_TRANSFER_DST_BIT = 0x00000002,
    VK_IMAGE_USAGE_SAMPLED_BIT = 0x00000004,
    VK_IMAGE_USAGE_STORAGE_BIT = 0x00000008,
    VK_IMAGE_USAGE_COLOR_ATTACHMENT_BIT = 0x00000010,
    VK_IMAGE_USAGE_DEPTH_STENCIL_ATTACHMENT_BIT = 0x00000020,
    VK_IMAGE_USAGE_TRANSIENT_ATTACHMENT_BIT = 0x00000040,
    VK_IMAGE_USAGE_INPUT_ATTACHMENT_BIT = 0x00000080,
    ...
} VkImageUsageFlagBits;
```

```
typedef struct VkExtent2D {
    uint32_t    width;
    uint32_t    height;
} VkExtent2D;
```

```
VkResult vkGetPhysicalDeviceSurfacePresentModesKHR(
    VkPhysicalDevice  physicalDevice,
    VkSurfaceKHR      surface,
    uint32_t*         pPresentModeCount,
    VkPresentModeKHR* PresentModes
);
```

VkPhysicalDevice

VkSurfaceKHR

```
typedef enum VkPresentModeKHR {
    VK_PRESENT_MODE_IMMEDIATE_KHR = 0,
    VK_PRESENT_MODE_MAILBOX_KHR = 1,
    VK_PRESENT_MODE_FIFO_KHR = 2,
    VK_PRESENT_MODE_FIFO_RELAXED_KHR = 3,
  // Provided by VK_KHR_shared_presentable_image
    VK_PRESENT_MODE_SHARED_DEMAND_REFRESH_KHR = 1000111000,
  // Provided by VK_KHR_shared_presentable_image
    VK_PRESENT_MODE_SHARED_CONTINUOUS_REFRESH_KHR = 1000111001,
} VkPresentModeKHR;
```

```
VkResult vkGetPhysicalDeviceSurfaceFormatsKHR(
    VkPhysicalDevice    physicalDevice,
    VkSurfaceKHR        surface,
    uint32_t*           pSurfaceFormatCount,
    VkSurfaceFormatKHR* pSurfaceFormats
);
```

VkPhysicalDevice

VkSurfaceKHR

```
typedef struct VkSurfaceFormatKHR {
    VkFormat           format;
    VkColorSpaceKHR    colorSpace;
} VkSurfaceFormatKHR;
```

```
typedef enum VkFormat {
    ...
    VK_FORMAT_R8G8B8_UNORM = 23,
    ...
    VK_FORMAT_R8G8B8_SRGB = 29,
    ...
} VkFormat;
```

```
typedef enum VkColorSpaceKHR {
    VK_COLOR_SPACE_SRGB_NONLINEAR_KHR,
    VK_COLOR_SPACE_DISPLAY_P3_NONLINEAR_EXT,
    VK_COLOR_SPACE_EXTENDED_SRGB_LINEAR_EXT,
    VK_COLOR_SPACE_DISPLAY_P3_LINEAR_EXT,
    VK_COLOR_SPACE_DCI_P3_NONLINEAR_EXT,
    VK_COLOR_SPACE_BT709_LINEAR_EXT,
    VK_COLOR_SPACE_BT709_NONLINEAR_EXT,
    VK_COLOR_SPACE_BT2020_LINEAR_EXT,
    VK_COLOR_SPACE_HDR10_ST2084_EXT,
    VK_COLOR_SPACE_DOLBYVISION_EXT,
    VK_COLOR_SPACE_HDR10_HLG_EXT,
    VK_COLOR_SPACE_ADOBERGB_LINEAR_EXT,
    VK_COLOR_SPACE_ADOBERGB_NONLINEAR_EXT,
    VK_COLOR_SPACE_PASS_THROUGH_EXT,
    VK_COLOR_SPACE_EXTENDED_SRGB_NONLINEAR_EXT,
    VK_COLOR_SPACE_DISPLAY_NATIVE_AMD,
    VK_COLORSPACE_SRGB_NONLINEAR_KHR,
    VK_COLOR_SPACE_DCI_P3_LINEAR_EXT,
} VkColorSpaceKHR;
```