

Copy Report to Clipboard

## Graphics Feature Status

- Canvas: **Hardware accelerated**
- Canvas out-of-process rasterization: **Disabled**
- Custo Wallpaper Animation: **Enabled**
- Direct Rendering Display Compositor: **Disabled**
- Compositing: **Hardware accelerated**
- Multiple Raster Threads: **Enabled**
- OpenGL: **Enabled**
- Rasterization: **Hardware accelerated**
- Raw Draw: **Disabled**
- Video Decode: **Hardware accelerated**
- Video Encode: **Software only. Hardware acceleration disabled**
- Vulkan: **Enabled**
- WebGL: **Hardware accelerated**
- WebGL2: **Hardware accelerated**
- WebGPU: **Disabled**

## Driver Bug Workarounds

- `adjust_src_dst_region_for_blitframebuffer`
- `clear_uniforms_before_first_program_use`
- `count_all_in_varyings_packing`
- `decode_encode_srgb_for_generatemipmap`
- `enable_webgl_timer_query_extensions`
- `exit_on_context_lost`
- `msaa_is_slow`
- `disabled_extension_GL_KHR_blend_equation_advanced`
- `disabled_extension_GL_KHR_blend_equation_advanced_coherent`

## Problems Detected

- WebGPU has been disabled via blocklist or the command line.  
*Disabled Features:* **`webgpu`**
- Accelerated video encode has been disabled, either via blocklist, `about:flags` or the command line.  
*Disabled Features:* **`video_encode`**
- Clear uniforms before first program use on all platforms: [124764](#), [349137](#)  
*Applied Workarounds:* **`clear_uniforms_before_first_program_use`**
- Mesa drivers in Linux handle varyings without static use incorrectly: [333885](#)  
*Applied Workarounds:* **`count_all_in_varyings_packing`**
- On Intel GPUs MSAA performance is not acceptable for GPU rasterization: [527565](#), [1298585](#)  
*Applied Workarounds:* **`msaa_is_slow`**
- adjust src/dst region if blitting pixels outside framebuffer on Linux Intel: [664740](#)  
*Applied Workarounds:* **`adjust_src_dst_region_for_blitframebuffer`**
- Disable KHR\_blend\_equation\_advanced until cc shaders are updated: [661715](#)  
*Applied Workarounds:* **`disable(GL_KHR_blend_equation_advanced)`**,  
**`disable(GL_KHR_blend_equation_advanced_coherent)`**
- Decode and Encode before generateMipmap for srgb format textures on Linux Mesa ANGLE path: [634519](#)  
*Applied Workarounds:* **`decode_encode_srgb_for_generatemipmap`**
- Expose WebGL's disjoint\_timer\_query extensions on platforms with site isolation: [808744](#), [870491](#)  
*Applied Workarounds:* **`enable_webgl_timer_query_extensions`**
- Some drivers can't recover after OUT\_OF\_MEM and context lost: [893177](#)  
*Applied Workarounds:* **`exit_on_context_lost`**

## ANGLE Features

- **allowCompressedFormats** (Frontend workarounds): **Enabled**: true  
*Allow compressed formats*
- **cacheCompiledShader** (Frontend features) [anglebug:7036](#): **Enabled**: true  
*Enable to cache compiled shaders*
- **disableAnisotropicFiltering** (Frontend workarounds): **Disabled**  
*Disable support for anisotropic filtering*
- **disableDrawBuffersIndexed** (Frontend features) [anglebug:7724](#): **Disabled**  
*Disable support for OES\_draw\_buffers\_indexed and EXT\_draw\_buffers\_indexed*
- **disableProgramBinary** (Frontend features) [anglebug:5007](#): **Disabled**  
*Disable support for GL\_OES\_get\_program\_binary*
- **disableProgramCachingForTransformFeedback** (Frontend workarounds): **Disabled**  
*On some GPUs, program binaries don't contain transform feedback varyings*
- **emulatePixelLocalStorage** (Frontend features) [anglebug:7279](#): **Disabled**: false  
*Emulate ANGLE\_shader\_pixel\_local\_storage using shader images*
- **enableCaptureLimits** (Frontend features) [anglebug:5750](#): **Disabled**  
*Set the context limits like frame capturing was enabled*
- **enableProgramBinaryForCapture** (Frontend features) [anglebug:5658](#): **Disabled**  
*Even if FrameCapture is enabled, enable GL\_OES\_get\_program\_binary*
- **forceDepthAttachmentInitOnClear** (Frontend workarounds) [anglebug:7246](#): **Disabled**  
*Force depth attachment initialization on clear ops*
- **forceGLErrorChecking** (Frontend features) <https://issuetracker.google.com/220069903>: **Disabled**: (IsAndroid() && isSwiftShader)  
*Force GL error checking (i.e. prevent applications from disabling error checking)*
- **forceInitShaderVariables** (Frontend features): **Disabled**  
*Force-enable shader variable initialization*
- **forceRobustResourceInit** (Frontend features) [anglebug:6041](#): **Disabled**  
*Force-enable robust resource init*
- **loseContextOnOutOfMemory** (Frontend workarounds): **Enabled**: true  
*Some users rely on a lost context notification if a GL\_OUT\_OF\_MEMORY error occurs*
- **scalarizeVecAndMatConstructorArgs** (Frontend workarounds) [1165751](#): **Disabled**: false  
*Always rewrite vec/mat constructors to be consistent*
- **singleThreadedTextureDecompression** (Frontend workarounds): **Disabled**  
*Disables multi-threaded decompression of compressed texture formats*
- **allocateNonZeroMemory** (Vulkan features) [anglebug:4384](#): **Disabled**: false  
*Fill new allocations with non-zero values to flush out errors.*
- **allowGenerateMipmapWithCompute** (Vulkan features) [anglebug:4551](#): **Disabled**: supportsSubgroupQuadOpsInComputeShader && mSubgroupExtendedTypesFeatures.shaderSubgroupExtendedTypes && maxComputeWorkGroupInvocations >= 256 && ((isAMD && !isWindows()) || isNvidia || isSamsung)  
*Use the compute path to generate mipmaps on devices that meet the minimum requirements, and the performance is better.*
- **asyncCommandQueue** (Vulkan features) [anglebug:4324](#): **Disabled**: false  
*Use CommandQueue worker thread to dispatch work to GPU.*
- **bottomLeftOriginPresentRegionRectangles** (Vulkan workarounds): **Disabled**: IsAndroid()  
*On some platforms present region rectangles are expected to have a bottom-left origin, instead of top-left origin as from spec*
- **bresenhamLineRasterization** (Vulkan features): **Enabled**: true  
*Enable Bresenham line rasterization via VK\_EXT\_line\_rasterization extension*
- **clampPointSize** (Vulkan workarounds) [anglebug:2970](#): **Disabled**: isNvidia && nvidiaVersion.major < uint32\_t(IsWindows() ? 430 : 421)  
*The point size range reported from the API is inconsistent with the actual behavior*
- **compressVertexData** (Vulkan workarounds): **Disabled**  
*Compress vertex data to smaller data types when possible. Using this feature makes ANGLE non-conformant.*

- **deferFlushUntilEndRenderPass** (Vulkan workarounds) <https://issuetracker.google.com/issues/166475273>: **Enabled**: !isQualcommProprietary  
*Allow glFlush to be deferred until renderpass ends*
- **depthClamping** (Vulkan workarounds) [anglebug:3970](#): **Disabled**: isNvidia && mPhysicalDeviceFeatures.depthClamp && ExtensionFound("VK\_EXT\_depth\_clip\_enable", deviceExtensionNames) && (!isLinux() || nvidiaVersion.major > 418u)  
*The depth value is not clamped to [0,1] for floating point depth buffers.*
- **disableFlippingBlitWithCommand** (Vulkan workarounds) [anglebug:3498](#): **Disabled**: isAndroid() && isQualcommProprietary  
*vkCmdBlitImage with flipped coordinates blits incorrectly.*
- **disallowMixedDepthStencilLoadOpNoneAndLoad** (Vulkan workarounds) [anglebug:7370](#): **Disabled**: isARM && armDriverVersion < ARMDriverVersion(38, 1, 0)  
*Disallow use of LOAD\_OP\_NONE for only one of the depth or stencil aspects of a depth/stencil attachment*
- **eglColorspaceAttributePassthrough** (Vulkan features) [anglebug:7319](#): **Disabled**: isAndroid() && isSamsung  
*Support passthrough of EGL colorspace attribute values*
- **emulateAdvancedBlendEquations** (Vulkan features) [anglebug:3586](#): **Disabled**: !mFeatures.supportsBlendOperationAdvanced.enabled && !isIntel  
*Emulate GL\_KHR\_blend\_equation\_advanced*
- **emulateDithering** (Vulkan features) [anglebug:6755](#): **Disabled**: isAndroid()  
*Emulate OpenGL dithering*
- **emulateR32fImageAtomicExchange** (Vulkan workarounds) [anglebug:5535](#): **Enabled**: true  
*Emulate r32f images with r32ui to support imageAtomicExchange.*
- **emulateTransformFeedback** (Vulkan features) [anglebug:3205](#): **Disabled**: (!mFeatures.supportsTransformFeedbackExtension.enabled && mPhysicalDeviceFeatures.vertexPipelineStoresAndAtomics == 1U)  
*Emulate transform feedback as the VK\_EXT\_transform\_feedback is not present.*
- **emulatedPrerotation180** (Vulkan features) [anglebug:4901](#): **Disabled**  
*Emulate 180-degree prerotation.*
- **emulatedPrerotation270** (Vulkan features) [anglebug:4901](#): **Disabled**  
*Emulate 270-degree prerotation.*
- **emulatedPrerotation90** (Vulkan features) [anglebug:4901](#): **Disabled**  
*Emulate 90-degree prerotation.*
- **enableAsyncPipelineCacheCompression** (Vulkan workarounds) [anglebug:4722](#): **Disabled**: false  
*Enable compressing pipeline cache in a thread.*
- **enableMultisampledRenderToTexture** (Vulkan workarounds) [anglebug:4937](#): **Disabled**: mFeatures.supportsMultisampledRenderToSingleSampled.enabled || mFeatures.supportsMultisampledRenderToSingleSampledGOOGLEX.enabled || (supportsIndependentDepthStencilResolve && (isTileBasedRenderer || isSamsung))  
*Expose EXT\_multisampled\_render\_to\_texture*
- **enablePreRotateSurfaces** (Vulkan features) [anglebug:3502](#): **Disabled**: isAndroid() && supportsNegativeViewport  
*Enable Android pre-rotation for landscape applications*
- **enablePrecisionQualifiers** (Vulkan features) [anglebug:3078](#): **Enabled**: ! (isPixel2(mPhysicalDeviceProperties.vendorID, mPhysicalDeviceProperties.deviceID) && (mPhysicalDeviceProperties.driverVersion < kPixel2DriverWithRelaxedPrecision)) && !isPixel4(mPhysicalDeviceProperties.vendorID, mPhysicalDeviceProperties.deviceID)  
*Enable precision qualifiers in shaders*
- **explicitlyEnablePerSampleShading** (Vulkan workarounds) [anglebug:6876](#): **Disabled**: isARM  
*Explicitly enable per-sample shading if the fragment shader contains the sample qualifier*
- **exposeNonConformantExtensionsAndVersions** (Vulkan workarounds) [anglebug:5375](#): **Disabled**: kExposeNonConformantExtensionsAndVersions  
*Expose GLES versions and extensions that are not conformant.*

- **forceContinuousRefreshOnSharedPresent** (Vulkan features) <https://issuetracker.google.com/229267970>: Disabled: false  
*Force to create vulkan swapchain with continuous refresh on shared present*
- **forceD16TexFilter** (Vulkan workarounds) [anglebug:3452](https://issuetracker.google.com/184850002): Disabled: IsAndroid() && isQualcommProprietary  
*VK\_FORMAT\_D16\_UNORM does not support VK\_FORMAT\_FEATURE\_SAMPLED\_IMAGE\_FILTER\_LINEAR\_BIT, which prevents OES\_depth\_texture from being supported.*
- **forceFallbackFormat** (Vulkan workarounds): Disabled  
*Force a fallback format for angle\_end2end\_tests*
- **forceFragmentShaderPrecisionHighpToMediump** (Vulkan workarounds) [https://issuetracker.google.com/184850002](https://issuetracker.google.com/161903006): Disabled: false  
*Forces highp precision in fragment shader to mediump.*
- **forceMaxUniformBufferSize16KB** (Vulkan workarounds) <https://issuetracker.google.com/161903006>: Disabled: isQualcommProprietary && isAdreno540  
*Force max uniform buffer size to 16K on some device due to bug*
- **forceNearestFiltering** (Vulkan workarounds): Disabled  
*Force nearest filtering when sampling.*
- **forceNearestMipFiltering** (Vulkan workarounds): Disabled  
*Force nearest mip filtering when sampling.*
- **forceStaticVertexStrideState** (Vulkan workarounds) <https://bugs.fuchsia.dev/p/fuchsia/issues/detail?id=107106>: Disabled: mFeatures.supportsExtendedDynamicState.enabled && isARM  
*Force static state for VK\_DYNAMIC\_STATE\_VERTEX\_INPUT\_BINDING\_STRIDE\_EXT due to driver bugs*
- **forceSubmitImmutableTextureUpdates** (Vulkan app workarounds) [anglebug:6929](https://issuetracker.google.com/253522366): Disabled  
*Force submit updates to immutable textures*
- **forceTextureLodOffset1** (Vulkan workarounds): Disabled  
*Increase the minimum texture level-of-detail by 1 when sampling.*
- **forceTextureLodOffset2** (Vulkan workarounds): Disabled  
*Increase the minimum texture level-of-detail by 2 when sampling.*
- **forceTextureLodOffset3** (Vulkan workarounds): Disabled  
*Increase the minimum texture level-of-detail by 3 when sampling.*
- **forceTextureLodOffset4** (Vulkan workarounds): Disabled  
*Increase the minimum texture level-of-detail by 4 when sampling.*
- **forceWaitForSubmissionToCompleteForQueryResult** (Vulkan workarounds) <https://issuetracker.google.com/253522366>: Disabled: isARM || (isNvidia && nvidiaVersion.major < 470u)  
*Force wait for submission to complete before calling getQueryResult(wait).*
- **hasEffectivePipelineCacheSerialization** (Vulkan features) [anglebug:7369](https://issuetracker.google.com/253522366): Enabled: isSwiftShader  
*Whether the implementation serializes the Vulkan pipeline cache effectively. On some implementations, pipeline cache serialization returns no data, so there is no benefit to serializing it*
- **logMemoryReportCallbacks** (Vulkan features): Disabled: false  
*Log each callback from VK\_EXT\_device\_memory\_report*
- **logMemoryReportStats** (Vulkan features): Disabled: false  
*Log stats from VK\_EXT\_device\_memory\_report each swap*
- **mapUnspecifiedColorSpaceToPassThrough** (Vulkan features): Disabled  
*Use VK\_COLOR\_SPACE\_PASS\_THROUGH\_EXT for EGL\_NONE or unspecified color spaces*
- **mergeProgramPipelineCachesToGlobalCache** (Vulkan workarounds) [anglebug:7369](https://issuetracker.google.com/253522366): Enabled: !mFeatures.supportsGraphicsPipelineLibrary.enabled || (mFeatures.preferMonolithicPipelinesOverLibraries.enabled && libraryBlobsAreReusedByMonolithicPipelines)  
*Whether it's beneficial to merge the pipeline cache for the shaders subset of the pipeline*



into the monolithic pipeline cache. Only useful on platforms where monolithic pipelines can reuse blobs from partial pipelines

- **mutableMipmapTextureUpload** (Vulkan features) [anglebug:7308](#): Enabled: `!(IsWindows()) && isIntel()`  
Enable uploading the previously defined mutable mipmap texture.
- **overrideSurfaceFormatRGB8ToRGBA8** (Vulkan workarounds) [anglebug:6651](#): Enabled: true  
Override surface format GL\_RGB8 to GL\_RGBA8
- **padBuffersToMaxVertexAttribStride** (Vulkan workarounds) [anglebug:4428](#): Disabled: `isAMD || isSamsung`  
Vulkan considers vertex attribute accesses to count up to the last multiple of the stride. This additional access supports AMD's robust buffer access implementation. AMDVLK in particular will return incorrect values when the vertex access extends into the range that would be the stride padding and the buffer is too small. This workaround limits GL\_MAX\_VERTEX\_ATTRIB\_STRIDE to a maximum value and pads up every buffer allocation size to be a multiple of the maximum stride.
- **perFrameWindowSizeQuery** (Vulkan workarounds) [anglebug:3623](#): Enabled: `IsAndroid() || isIntel() || (IsWindows() && isAMD) || IsFuchsia() || isSamsung || displayVk->isWayland()`  
Vulkan swapchain is not returning VK\_ERROR\_OUT\_OF\_DATE when window resizing
- **permanentlySwitchToFramebufferFetchMode** (Vulkan features): Disabled: `isTileBasedRenderer`  
Whether the context should permanently switch to framebuffer fetch mode on first encounter
- **persistentlyMappedBuffers** (Vulkan features) [anglebug:2162](#): Enabled: true  
Persistently map buffer memory to reduce map/unmap IOCTL overhead.
- **precisionSafeDivision** (Vulkan workarounds): Disabled: `isSamsung || isAMD`  
Special case handling for platforms that do not generate 1.0f even when the dividend and divisor have the same value
- **preferAggregateBarrierCalls** (Vulkan workarounds) [anglebug:4633](#): Enabled: `isImmediateModeRenderer`  
Single barrier call is preferred over multiple calls with fine grained pipeline stage dependency information
- **preferCPUForBufferSubData** (Vulkan features) <http://issuetracker.google.com/200067929>: Disabled: `isARM`  
Prefer use CPU to do bufferSubData instead of staged update.
- **preferDeviceLocalMemoryHostVisible** (Vulkan features) [anglebug:7047](#): Enabled: `canPreferDeviceLocalMemoryHostVisible(mPhysicalDeviceProperties.deviceType)`  
Prefer adding HOST\_VISIBLE flag for DEVICE\_LOCAL memory when picking memory types
- **preferDrawClearOverVkCmdClearAttachments** (Vulkan workarounds) <https://issuetracker.google.com/166809097>: Disabled: `isQualcommProprietary`  
On some hardware, clear using a draw call instead of vkCmdClearAttachments in the middle of render pass due to bugs
- **preferDriverUniformOverSpecConst** (Vulkan features) [anglebug:7406](#): Disabled: `(isQualcommProprietary && mPhysicalDeviceProperties.driverVersion < kPixel4DriverWithWorkingSpecConstSupport) || isARM || isPowerVR || isSwiftShader`  
Prefer using driver uniforms instead of specialization constants.
- **preferLinearFilterForYUV** (Vulkan features) [anglebug:7382](#): Disabled  
Prefer to use VK\_FILTER\_LINEAR for VkSamplerYcbcrConversion
- **preferMonolithicPipelinesOverLibraries** (Vulkan workarounds) [anglebug:7369](#): Enabled: `!mGraphicsPipelineLibraryProperties.graphicsPipelineLibraryFastLinking || isSwiftShader`  
Whether monolithic pipelines perform significantly better than libraries
- **preferSkippingInvalidateForEmulatedFormats** (Vulkan workarounds) [anglebug:6860](#): Enabled: `isImmediateModeRenderer`  
Skipping invalidate is preferred for emulated formats that have extra channels over re-clearing the image
- **preferSubmitAtFBOBoundary** (Vulkan workarounds) <https://issuetracker.google.com/187425444>: Disabled: `isARM || isSwiftShader`

*Submit commands to driver at each FBO boundary for performance improvements.*

- **preferSubmitOnAnySamplesPassedQueryEnd** (Vulkan workarounds) <https://issuetracker.google.com/250706693>: **Disabled**: isTileBasedRenderer  
*Submit commands to driver when last GL\_ANY\_SAMPLES\_PASSED query is made for performance improvements.*
- **provokingVertex** (Vulkan features): **Enabled**: true  
*Enable provoking vertex mode via VK\_EXT\_provoking\_vertex extension*
- **retainSPIRVDebugInfo** (Vulkan features) [anglebug:5901](#): **Disabled**:  
getEnableValidationLayers()  
*Retain debug info in SPIR-V blob.*
- **roundOutputAfterDithering** (Vulkan workarounds) [anglebug:6953](#): **Disabled**:  
isQualcomm  
*Round output after dithering to workaround a driver bug that rounds the output up*
- **slowDownMonolithicPipelineCreationForTesting** (Vulkan workarounds) [anglebug:7369](#): **Disabled**  
*Artificially slow down async monolithic pipeline creation for threading testing*
- **supportsAndroidHardwareBuffer** (Vulkan features): **Disabled**  
*VkDevice supports the VK\_ANDROID\_external\_memory\_android\_hardware\_buffer extension*
- **supportsAndroidNativeFenceSync** (Vulkan features) [anglebug:2517](#): **Disabled**  
*VkDevice supports the EGL\_ANDROID\_native\_fence\_sync extension*
- **supportsBlendOperationAdvanced** (Vulkan features) [anglebug:3586](#): **Disabled**:  
ExtensionFound("VK\_EXT\_blend\_operation\_advanced", deviceExtensionNames)  
*VkDevice supports VK\_EXT\_blend\_operation\_advanced extension.*
- **supportsColorWriteEnable** (Vulkan features) [anglebug:7161](#): **Disabled**  
*VkDevice supports VK\_EXT\_color\_write\_enable extension*
- **supportsComputeTranscodeEtcToBc** (Vulkan features): **Disabled**:  
!mPhysicalDeviceFeatures.textureCompressionETC2 && kSupportTranscodeEtcToBc &&  
(mSubgroupProperties.supportedOperations & kRequiredSubgroupOp) ==  
kRequiredSubgroupOp && (limitsVk.maxTexelBufferElements >= kMaxTexelBufferSize)  
*supports compute shader transcode etc format to bc format*
- **supportsCustomBorderColor** (Vulkan features) [anglebug:3577](#): **Enabled**:  
mCustomBorderColorFeatures.customBorderColors == 1U &&  
mCustomBorderColorFeatures.customBorderColorWithoutFormat == 1U  
*VkDevice supports the VK\_EXT\_custom\_border\_color extension*
- **supportsDepthClipControl** (Vulkan features) [anglebug:5421](#): **Enabled**:  
mDepthClipControlFeatures.depthClipControl == 1U  
*VkDevice supports VK\_EXT\_depth\_clip\_control extension.*
- **supportsDepthStencilResolve** (Vulkan features) [anglebug:4836](#): **Enabled**:  
mFeatures.supportsRenderpass2.enabled &&  
mDepthStencilResolveProperties.supportedDepthResolveModes != 0  
*VkDevice supports the VK\_KHR\_depth\_stencil\_resolve extension with the independentResolveNone feature*
- **supportsExtendedDynamicState** (Vulkan features) [anglebug:5906](#): **Enabled**:  
mExtendedDynamicStateFeatures.extendedDynamicState == 1U  
*VkDevice supports VK\_EXT\_extended\_dynamic\_state extension*
- **supportsExtendedDynamicState2** (Vulkan features) [anglebug:5906](#): **Enabled**:  
mExtendedDynamicState2Features.extendedDynamicState2 == 1U  
*VkDevice supports VK\_EXT\_extended\_dynamic\_state2 extension*
- **supportsExternalFenceCapabilities** (Vulkan features): **Enabled**: true  
*VkInstance supports the VK\_KHR\_external\_fence\_capabilities extension*
- **supportsExternalFenceFd** (Vulkan features) [anglebug:2517](#): **Enabled**:  
ExtensionFound("VK\_KHR\_external\_fence\_fd", deviceExtensionNames)  
*VkDevice supports the VK\_KHR\_external\_fence\_fd extension*
- **supportsExternalMemoryDmaBufAndModifiers** (Vulkan features) [anglebug:6248](#):  
**Enabled**: ExtensionFound("VK\_EXT\_external\_memory\_dma\_buf",  
deviceExtensionNames) && ExtensionFound("VK\_EXT\_image\_drm\_format\_modifier",  
deviceExtensionNames)

*VkDevice supports the VK\_EXT\_external\_memory\_dma\_buf and VK\_EXT\_image\_drm\_format\_modifier extensions*

- **supportsExternalMemoryFd** (Vulkan features): **Enabled**:  
*ExtensionFound("VK\_KHR\_external\_memory\_fd", deviceExtensionNames)*  
*VkDevice supports the VK\_KHR\_external\_memory\_fd extension*
- **supportsExternalMemoryFuchsia** (Vulkan features): **Disabled**:  
*ExtensionFound("VK\_FUCHSIA\_external\_memory", deviceExtensionNames)*  
*VkDevice supports the VK\_FUCHSIA\_external\_memory extension*
- **supportsExternalMemoryHost** (Vulkan features): **Enabled**:  
*ExtensionFound("VK\_EXT\_external\_memory\_host", deviceExtensionNames)*  
*VkDevice supports the VK\_EXT\_external\_memory\_host extension*
- **supportsExternalSemaphoreCapabilities** (Vulkan features): **Enabled**: true  
*VkInstance supports the VK\_KHR\_external\_semaphore\_capabilities extension*
- **supportsExternalSemaphoreFd** (Vulkan features): **Enabled**:  
*ExtensionFound("VK\_KHR\_external\_semaphore\_fd", deviceExtensionNames)*  
*VkDevice supports the VK\_KHR\_external\_semaphore\_fd extension*
- **supportsExternalSemaphoreFuchsia** (Vulkan features): **Disabled**:  
*ExtensionFound("VK\_FUCHSIA\_external\_semaphore", deviceExtensionNames)*  
*VkDevice supports the VK\_FUCHSIA\_external\_semaphore extension*
- **supportsFilteringPrecision** (Vulkan features): **Disabled**:  
*ExtensionFound("VK\_GOOGLE\_sampler\_filtering\_precision", deviceExtensionNames)*  
*VkDevice supports the VK\_GOOGLE\_sampler\_filtering\_precision extension*
- **supportsFragmentShaderPixelInterlock** (Vulkan features): **Enabled**:  
*mFragmentShaderInterlockFeatures.fragmentShaderPixelInterlock == 1U*  
*VkDevice supports the VK\_EXT\_fragment\_shader\_interlock extension and has the fragmentShaderPixelInterlock feature*
- **supportsFragmentShadingRate** (Vulkan features) [anglebug:7172](#): **Enabled**:  
*canSupportFragmentShadingRate(deviceExtensionNames)*  
*VkDevice supports VK\_KHR\_fragment\_shading\_rate extension*
- **supportsGGPFrameToken** (Vulkan features): **Disabled**:  
*VkDevice supports the VK\_GGP\_frame\_token extension*
- **supportsGeometryStreamsCapability** (Vulkan features) [anglebug:3206](#): **Enabled**:  
*mTransformFeedbackFeatures.geometryStreams == 1U*  
*Implementation supports the GeometryStreams SPIR-V capability.*
- **supportsGraphicsPipelineLibrary** (Vulkan features) [anglebug:7369](#): **Disabled**:  
*mGraphicsPipelineLibraryFeatures.graphicsPipelineLibrary == 1U*  
*VkDevice supports the VK\_EXT\_graphics\_pipeline\_library extension*
- **supportsHostQueryReset** (Vulkan features) [anglebug:6692](#): **Enabled**:  
*mHostQueryResetFeatures.hostQueryReset == 1U*  
*VkDevice supports VK\_EXT\_host\_query\_reset extension*
- **supportsImage2dViewOf3d** (Vulkan features) [anglebug:7320](#): **Enabled**:  
*mImage2dViewOf3dFeatures.image2DViewOf3D == 1U &&*  
*mImage2dViewOf3dFeatures.sampler2DViewOf3D == 1U*  
*VkDevice supports VK\_EXT\_image\_2d\_view\_of\_3d*
- **supportsImageCubeArray** (Vulkan features) [anglebug:3584](#): **Enabled**:  
*mPhysicalDeviceFeatures.imageCubeArray == 1U*  
*VkDevice supports the imageCubeArray feature properly*
- **supportsImageFormatList** (Vulkan features) [anglebug:5281](#): **Enabled**:  
*ExtensionFound("VK\_KHR\_image\_format\_list", deviceExtensionNames)*  
*Enable VK\_IMAGE\_CREATE\_MUTABLE\_FORMAT\_BIT by default for ICDs that support VK\_KHR\_image\_format\_list*
- **supportsImagelessFramebuffer** (Vulkan features) [anglebug:7553](#): **Enabled**:  
*mImagelessFramebufferFeatures.imagelessFramebuffer == 1U*  
*VkDevice supports VK\_KHR\_imageless\_framebuffer extension*
- **supportsIncrementalPresent** (Vulkan features): **Enabled**:  
*ExtensionFound("VK\_KHR\_incremental\_present", deviceExtensionNames)*  
*VkDevice supports the VK\_KHR\_incremental\_present extension*



- **supportsIndexTypeUint8** (Vulkan features) [anglebug:4405](#): **Enabled**:  
mIndexTypeUint8Features.indexTypeUint8 == 1U  
*VkDevice supports the VK\_EXT\_index\_type\_uint8 extension*
- **supportsLockSurfaceExtension** (Vulkan features): **Disabled**: IsAndroid()  
*Surface supports the EGL\_KHR\_lock\_surface3 extension*
- **supportsLogicOpDynamicState** (Vulkan features) [anglebug:3862](#): **Enabled**:  
mExtendedDynamicState2Features.extendedDynamicState2LogicOp == 1U && (!  
(IsLinux() && isIntel()) || isAtLeastMesa22\_2)  
*VkDevice supports the logicOp feature of VK\_EXT\_extended\_dynamic\_state2 extension*
- **supportsMultiDrawIndirect** (Vulkan features) [anglebug:6439](#): **Enabled**:  
mPhysicalDeviceFeatures.multiDrawIndirect == 1U  
*VkDevice supports the multiDrawIndirect extension*
- **supportsMultisampledRenderToSingleSampled** (Vulkan features) [anglebug:4836](#):  
**Disabled**: mFeatures.supportsRenderpass2.enabled &&  
mFeatures.supportsDepthStencilResolve.enabled &&  
mMultisampledRenderToSingleSampledFeatures.multisampledRenderToSingleSampled  
== 1U  
*VkDevice supports the VK\_EXT\_multisampled\_render\_to\_single\_sampled extension*
- **supportsMultisampledRenderToSingleSampledGOOGLEX** (Vulkan features)  
[anglebug:4836](#): **Disabled**:  
!mFeatures.supportsMultisampledRenderToSingleSampled.enabled &&  
mFeatures.supportsRenderpass2.enabled &&  
mFeatures.supportsDepthStencilResolve.enabled &&  
mMultisampledRenderToSingleSampledFeaturesGOOGLEX.multisampledRenderToSingleS  
== 1U  
*VkDevice supports the VK\_GOOGLEX\_multisampled\_render\_to\_single\_sampled  
extension*
- **supportsMultiview** (Vulkan features) [anglebug:6048](#): **Enabled**:  
mMultiviewFeatures.multiview == 1U  
*VkDevice supports the VK\_KHR\_multiview extension*
- **supportsNegativeViewport** (Vulkan features): **Enabled**: supportsNegativeViewport  
*The driver supports inverting the viewport with a negative height.*
- **supportsPipelineCreationCacheControl** (Vulkan features) [anglebug:5881](#): **Enabled**:  
mPipelineCreationCacheControlFeatures.pipelineCreationCacheControl &&  
!isSwiftShader  
*VkDevice supports VK\_EXT\_pipeline\_creation\_cache\_control extension*
- **supportsPipelineCreationFeedback** (Vulkan features) [anglebug:5881](#): **Enabled**:  
ExtensionFound("VK\_EXT\_pipeline\_creation\_feedback", deviceExtensionNames) ||  
mPhysicalDeviceProperties.apiVersion >= (((uint32\_t)(0)) << 29) | (((uint32\_t)(1)) << 22) |  
(((uint32\_t)(3)) << 12) | ((uint32\_t)(0)))  
*VkDevice supports VK\_EXT\_pipeline\_creation\_feedback extension*
- **supportsPipelineProtectedAccess** (Vulkan features) [anglebug:7714](#): **Disabled**:  
mPipelineProtectedAccessFeatures.pipelineProtectedAccess == 1U &&  
mProtectedMemoryFeatures.protectedMemory == 1U  
*VkDevice supports the VK\_EXT\_pipeline\_protected\_access extension*
- **supportsPipelineRobustness** (Vulkan features) [anglebug:5845](#): **Disabled**:  
mPipelineRobustnessFeatures.pipelineRobustness == 1U &&  
mPhysicalDeviceFeatures.robustBufferAccess  
*VkDevice supports VK\_EXT\_pipeline\_robustness extension*
- **supportsPipelineStatisticsQuery** (Vulkan features) [anglebug:5430](#): **Enabled**:  
mPhysicalDeviceFeatures.pipelineStatisticsQuery == 1U  
*VkDevice supports the pipelineStatisticsQuery feature*
- **supportsPresentation** (Vulkan features): **Enabled**: !displayVk->isGBM()  
*VkDisplay supports presentation through a present family queue*
- **supportsPrimitiveTopologyListRestart** (Vulkan features) [anglebug:3832](#): **Enabled**:  
mPrimitiveTopologyListRestartFeatures.primitiveTopologyListRestart == 1U  
*VkDevice supports VK\_EXT\_primitive\_topology\_list\_restart extension.*



- **supportsPrimitivesGeneratedQuery** (Vulkan features) [anglebug:5430](#): **Enabled**:  
mPrimitivesGeneratedQueryFeatures.primitivesGeneratedQuery == 1U  
*VkDevice supports VK\_EXT\_primitives\_generated\_query extension*
- **supportsProtectedMemory** (Vulkan features) [anglebug:3965](#): **Disabled**:  
mProtectedMemoryFeatures.protectedMemory == 1U && (!isARM ||  
mPipelineProtectedAccessFeatures.pipelineProtectedAccess == 1U)  
*VkDevice supports protected memory*
- **supportsRasterizationOrderAttachmentAccess** (Vulkan features) [anglebug:7604](#):  
**Disabled**: !isQualcomm &&  
mRasterizationOrderAttachmentAccessFeatures.rasterizationOrderColorAttachmentAccess  
== 1U  
*VkDevice supports VK\_EXT\_rasterization\_order\_attachment\_access extension*
- **supportsRenderPassLoadStoreOpNone** (Vulkan features) [anglebug:5371](#): **Disabled**:  
ExtensionFound("VK\_EXT\_load\_store\_op\_none", deviceExtensionNames)  
*VkDevice supports VK\_EXT\_load\_store\_op\_none extension.*
- **supportsRenderPassStoreOpNone** (Vulkan features) [anglebug:5055](#): **Disabled**:  
!mFeatures.supportsRenderPassLoadStoreOpNone.enabled &&  
ExtensionFound("VK\_QCOM\_render\_pass\_store\_ops", deviceExtensionNames)  
*VkDevice supports VK\_QCOM\_render\_pass\_store\_ops extension.*
- **supportsRenderpass2** (Vulkan features): **Enabled**:  
ExtensionFound("VK\_KHR\_create\_renderpass2", deviceExtensionNames)  
*VkDevice supports the VK\_KHR\_create\_renderpass2 extension*
- **supportsShaderFloat16** (Vulkan features) [anglebug:4551](#): **Enabled**:  
mShaderFloat16Int8Features.shaderFloat16 == 1U  
*VkDevice supports the VK\_KHR\_shader\_float16\_int8 extension and has the  
shaderFloat16 feature*
- **supportsShaderFramebufferFetch** (Vulkan features): **Disabled**: (IsAndroid() && isARM)  
|| mFeatures.supportsRasterizationOrderAttachmentAccess.enabled  
*Whether the Vulkan backend supports coherent framebuffer fetch*
- **supportsShaderFramebufferFetchNonCoherent** (Vulkan features): **Disabled**:  
(IsAndroid() && !(isARM || isQualcomm)) || isSwiftShader  
*Whether the Vulkan backend supports non-coherent framebuffer fetch*
- **supportsShaderStencilExport** (Vulkan features): **Enabled**:  
ExtensionFound("VK\_EXT\_shader\_stencil\_export", deviceExtensionNames)  
*VkDevice supports the VK\_EXT\_shader\_stencil\_export extension*
- **supportsSharedPresentableImageExtension** (Vulkan features): **Disabled**:  
*VkSurface supports the VK\_KHR\_shared\_presentable\_images extension*
- **supportsSurfaceCapabilities2Extension** (Vulkan features): **Enabled**: true  
*VkInstance supports the VK\_KHR\_get\_surface\_capabilities2 extension*
- **supportsSurfaceProtectedCapabilitiesExtension** (Vulkan features): **Enabled**: true  
*VkInstance supports the VK\_KHR\_surface\_protected\_capabilities extension*
- **supportsSurfaceProtectedSwapchains** (Vulkan features): **Disabled**: IsAndroid()  
*VkSurface supportsProtected for protected swapchains*
- **supportsSurfacelessQueryExtension** (Vulkan features): **Disabled**:  
*VkInstance supports the VK\_GOOGLE\_surfaceless\_query extension*
- **supportsTimestampSurfaceAttribute** (Vulkan features) [anglebug:7489](#): **Disabled**:  
IsAndroid() && ExtensionFound("VK\_GOOGLE\_display\_timing", deviceExtensionNames)  
*Platform supports setting frame timestamp surface attribute*
- **supportsTransformFeedbackExtension** (Vulkan features) [anglebug:3206](#): **Enabled**:  
mTransformFeedbackFeatures.transformFeedback == 1U  
*Transform feedback uses the VK\_EXT\_transform\_feedback extension.*
- **supportsVertexInputDynamicState** (Vulkan features) [anglebug:7162](#): **Disabled**:  
*VkDevice supports VK\_EXT\_vertex\_input\_dynamic\_state extension*
- **supportsYUVSamplerConversion** (Vulkan features): **Enabled**:  
mSamplerYcbcrConversionFeatures.samplerYcbcrConversion != 0U  
*VkDevice supports the VK\_KHR\_sampler\_ycbcr\_conversion extension*
- **supportsYuvTarget** (Vulkan features): **Disabled**:  
*VkDevice supports VK\_ANDROID\_render\_to\_external\_format and*

VK\_EXT\_ycbcr\_attachment

- **swapbuffersOnFlushOrFinishWithSingleBuffer** (Vulkan features) [anglebug:6878](#):  
**Disabled**: IsAndroid()  
*Bypass deferredFlush with calling swapbuffers on flush or finish when in Shared Present mode*
- **syncMonolithicPipelinesToBlobCache** (Vulkan workarounds) [anglebug:7369](#): **Enabled**:  
mFeatures.hasEffectivePipelineCacheSerialization.enabled && (hasNoPipelineWarmUp || canSyncLargeMonolithicCache)  
*Whether it's beneficial to store monolithic pipelines in the blob cache when VK\_EXT\_graphics\_pipeline\_library is in use. Otherwise the libraries are stored only, and monolithic pipelines are recreated on every run*
- **useMultipleDescriptorsForExternalFormats** (Vulkan workarounds) [anglebug:6141](#):  
**Enabled**: true  
*Return a default descriptor count for external formats.*
- **useNonZeroStencilWriteMaskStaticState** (Vulkan workarounds) [anglebug:7556](#):  
**Disabled**: isARM && armDriverVersion < ARMDriverVersion(40, 0, 0)  
*Work around a driver bug where 0 in stencil write mask static state would make the corresponding dynamic state malfunction in the presence of discard or alpha to coverage*
- **varyingsRequireMatchingPrecisionInSpirv** (Vulkan workarounds) [anglebug:7488](#):  
**Disabled**: isPowerVR  
*Add additional SPIRV instructions to make sure precision between shader stages match with each other*
- **waitIdleBeforeSwapchainRecreation** (Vulkan workarounds) [anglebug:5061](#): **Disabled**:  
IsAndroid() && isARM  
*Before passing an oldSwapchain to VkSwapchainCreateInfoKHR, wait for queue to be idle. Works around a bug on platforms which destroy oldSwapchain in vkCreateSwapchainKHR.*
- **warmUpPipelineCacheAtLink** (Vulkan features) [anglebug:5881](#): **Disabled**:  
libraryBlobsAreReusedByMonolithicPipelines && !isQualcommProprietary && !(IsLinux() && isIntel) && !(IsChromeOS()) && isSwiftShader)  
*Warm up the Vulkan pipeline cache at link time*

## DAWN Info

### <Integrated GPU> Vulkan backend - Intel(R) Graphics (ADL GT2)

[Default Toggle Names]

- **lazy\_clear\_resource\_on\_first\_use**: <https://crbug.com/dawn/145>: Clears resource to zero on first usage. This initializes the resource so that no dirty bits from recycled memory is present in the new resource.
- **use\_temporary\_buffer\_in\_texture\_to\_texture\_copy**: <https://crbug.com/dawn/42>: Split texture-to-texture copy into two copies: copy from source texture into a temporary buffer, and copy from the temporary buffer into the destination texture when copying between compressed textures that don't have block-aligned sizes. This workaround is enabled by default on all Vulkan drivers to solve an issue in the Vulkan SPEC about the texture-to-texture copies with compressed formats. See #1005 (<https://github.com/KhronosGroup/Vulkan-Docs/issues/1005>) for more details.
- **vulkan\_use\_d32s8**: <https://crbug.com/dawn/286>: Vulkan mandates support of either D32\_FLOAT\_S8 or D24\_UNORM\_S8. When available the backend will use D32S8 (toggle to on) but setting the toggle to off will make it use the D24S8 format when possible.
- **vulkan\_use\_s8**: <https://crbug.com/dawn/666>: Vulkan has a pure stencil8 format but it is not universally available. When this toggle is on, the backend will use S8 for the stencil8 format, otherwise it will fallback to D32S8 or D24S8.
- **disallow\_unsafe\_apis**: <http://crbug.com/1138528>: Produces validation errors on API entry points or parameter combinations that aren't considered secure yet.
- **use\_vulkan\_zero\_initialize\_workgroup\_memory\_extension**:  
<https://crbug.com/dawn/1302>: Initialize workgroup memory with OpConstantNull on Vulkan when the Vulkan extension VK\_KHR\_zero\_initialize\_workgroup\_memory is supported.  
[WebGPU Forced Toggles - enabled]

- **disallow\_spirv:** <https://crbug.com/1214923>: Disallow usage of SPIR-V completely so that only WGSL is used for shader modules. This is useful to prevent a Chromium renderer process from successfully sending SPIR-V code to be compiled in the GPU process.  
[Supported Features]
- texture-compression-bc
- texture-compression-etc2
- texture-compression-astc
- pipeline-statistics-query
- timestamp-query
- timestamp-query-inside-passes
- depth-clip-control
- depth32float-stencil8
- chromium-experimental-dp4a
- indirect-first-instance
- rg11b10float-renderable
- dawn-internal-usages
- dawn-native

### <CPU> Vulkan backend - llvmpipe (LLVM 11.0.1, 256 bits)

#### [Default Toggle Names]

- **lazy\_clear\_resource\_on\_first\_use:** <https://crbug.com/dawn/145>: Clears resource to zero on first usage. This initializes the resource so that no dirty bits from recycled memory is present in the new resource.
- **use\_temporary\_buffer\_in\_texture\_to\_texture\_copy:** <https://crbug.com/dawn/42>: Split texture-to-texture copy into two copies: copy from source texture into a temporary buffer, and copy from the temporary buffer into the destination texture when copying between compressed textures that don't have block-aligned sizes. This workaround is enabled by default on all Vulkan drivers to solve an issue in the Vulkan SPEC about the texture-to-texture copies with compressed formats. See #1005 (<https://github.com/KhronosGroup/Vulkan-Docs/issues/1005>) for more details.
- **vulkan\_use\_d32s8:** <https://crbug.com/dawn/286>: Vulkan mandates support of either D32\_FLOAT\_S8 or D24\_UNORM\_S8. When available the backend will use D32S8 (toggle to on) but setting the toggle to off will make it use the D24S8 format when possible.
- **vulkan\_use\_s8:** <https://crbug.com/dawn/666>: Vulkan has a pure stencil8 format but it is not universally available. When this toggle is on, the backend will use S8 for the stencil8 format, otherwise it will fallback to D32S8 or D24S8.
- **disallow\_unsafe\_apis:** <http://crbug.com/1138528>: Produces validation errors on API entry points or parameter combinations that aren't considered secure yet.
- **use\_vulkan\_zero\_initialize\_workgroup\_memory\_extension:** <https://crbug.com/dawn/1302>: Initialize workgroup memory with OpConstantNull on Vulkan when the Vulkan extension VK\_KHR\_zero\_initialize\_workgroup\_memory is supported.  
[WebGPU Forced Toggles - enabled]
- **disallow\_spirv:** <https://crbug.com/1214923>: Disallow usage of SPIR-V completely so that only WGSL is used for shader modules. This is useful to prevent a Chromium renderer process from successfully sending SPIR-V code to be compiled in the GPU process.  
[Supported Features]
- texture-compression-bc
- pipeline-statistics-query
- timestamp-query
- timestamp-query-inside-passes
- depth-clip-control
- depth32float-stencil8
- indirect-first-instance
- rg11b10float-renderable
- dawn-internal-usages
- dawn-native

### <CPU> Vulkan backend - SwiftShader Device (Subzero)

**[Default Toggle Names]**

- **lazy\_clear\_resource\_on\_first\_use:** <https://crbug.com/dawn/145>: Clears resource to zero on first usage. This initializes the resource so that no dirty bits from recycled memory is present in the new resource.
- **use\_temporary\_buffer\_in\_texture\_to\_texture\_copy:** <https://crbug.com/dawn/42>: Split texture-to-texture copy into two copies: copy from source texture into a temporary buffer, and copy from the temporary buffer into the destination texture when copying between compressed textures that don't have block-aligned sizes. This workaround is enabled by default on all Vulkan drivers to solve an issue in the Vulkan SPEC about the texture-to-texture copies with compressed formats. See #1005 (<https://github.com/KhronosGroup/Vulkan-Docs/issues/1005>) for more details.
- **vulkan\_use\_d32s8:** <https://crbug.com/dawn/286>: Vulkan mandates support of either D32\_FLOAT\_S8 or D24\_UNORM\_S8. When available the backend will use D32S8 (toggle to on) but setting the toggle to off will make it use the D24S8 format when possible.
- **vulkan\_use\_s8:** <https://crbug.com/dawn/666>: Vulkan has a pure stencil8 format but it is not universally available. When this toggle is on, the backend will use S8 for the stencil8 format, otherwise it will fallback to D32S8 or D24S8.
- **disallow\_unsafe\_apis:** <http://crbug.com/1138528>: Produces validation errors on API entry points or parameter combinations that aren't considered secure yet.
- **use\_vulkan\_zero\_initialize\_workgroup\_memory\_extension:** <https://crbug.com/dawn/1302>: Initialize workgroup memory with OpConstantNull on Vulkan when the Vulkan extension VK\_KHR\_zero\_initialize\_workgroup\_memory is supported.

**[WebGPU Forced Toggles - enabled]**

- **disallow\_spirv:** <https://crbug.com/1214923>: Disallow usage of SPIR-V completely so that only WGSL is used for shader modules. This is useful to prevent a Chromium renderer process from successfully sending SPIR-V code to be compiled in the GPU process.

**[Supported Features]**

- texture-compression-bc
- texture-compression-etc2
- texture-compression-astc
- timestamp-query
- timestamp-query-inside-passes
- depth-clip-control
- depth32float-stencil8
- indirect-first-instance
- rg11b10float-renderable
- dawn-internal-usages
- dawn-native

**Version Information**

<b>Data exported</b>	2023-05-24T14:32:44.079Z
<b>Chrome version</b>	YaBrowser/23.3.1.946 (corp)
<b>Operating system</b>	Linux 6.1.29-un-def-alt1
<b>Software rendering list path</b>	Path=/src/gpu/config/software_rendering_list.json RevisionId=694ad6c9e86008a1a7f5b19dfd05c1971af3b905
<b>Driver bug list path</b>	Path=/src/gpu/config/gpu_driver_bug_list.json RevisionId=694ad6c9e86008a1a7f5b19dfd05c1971af3b905
<b>ANGLE commit id</b>	unknown hash
<b>2D graphics backend</b>	Skia/110 694ad6c9e86008a1a7f5b19dfd05c1971af3b905
<b>Command Line</b>	/usr/bin/yandex-browser-stable --enable-features=DefaultANGLEVulkan,Vulkan,VulkanFromANGLE --flag-switches-begin --flag-switches-end --external-app-null-path --external-app-data=null_data

**Driver Information**

<b>Initialization time</b>	49
<b>In-process GPU</b>	false



<b>Passthrough Command Decoder</b>	true
<b>Sandboxed</b>	false
<b>GPU0</b>	VENDOR= 0x8086 [Google Inc. (Intel)], DEVICE=0x46a6 [ANGLE (Intel, Vulkan 1.3.230 (Intel(R) Graphics (ADL GT2) (0x000046A6)), Intel open-source Mesa driver-22.3.6)], DRIVER_VENDOR=Mesa, DRIVER_VERSION=driver *ACTIVE*
<b>Optimus</b>	false
<b>AMD switchable</b>	false
<b>GPU CUDA compute capability major version</b>	0
<b>Pixel shader version</b>	1.00
<b>Vertex shader version</b>	1.00
<b>Max. MSAA samples</b>	16
<b>Machine model name</b>	
<b>Machine model version</b>	
<b>GL_VENDOR</b>	Google Inc. (Intel)
<b>GL_RENDERER</b>	ANGLE (Intel, Vulkan 1.3.230 (Intel(R) Graphics (ADL GT2) (0x000046A6)), Intel open-source Mesa driver-22.3.6)
<b>GL_VERSION</b>	OpenGL ES 2.0.0 (ANGLE 2.1.0 git hash: unknown hash)
<b>GL_EXTENSIONS</b>	GL_AMD_performance_monitor GL_ANGLE_base_vertex_base_instance GL_ANGLE_base_vertex_base_instance_shader_builtin GL_ANGLE_client_arrays GL_ANGLE_compressed_texture_etc GL_ANGLE_depth_texture GL_ANGLE_framebuffer_blit GL_ANGLE_framebuffer_multisample GL_ANGLE_get_image GL_ANGLE_get_serialized_context_string GL_ANGLE_get_tex_level_parameter GL_ANGLE_instanced_arrays GL_ANGLE_logic_op GL_ANGLE_memory_object_flags GL_ANGLE_memory_size GL_ANGLE_multi_draw GL_ANGLE_program_cache_control GL_ANGLE_read_only_depth_stencil_feedback_loops GL_ANGLE_relaxed_vertex_attribute_type GL_ANGLE_request_extension GL_ANGLE_rgbx_internal_format GL_ANGLE_robust_client_memory GL_ANGLE_robust_fragment_shader_output GL_ANGLE_texture_compression_dxt3 GL_ANGLE_texture_compression_dxt5 GL_ANGLE_texture_usage GL_ANGLE_vulkan_image GL_APPLE_clip_distance GL_CHROMIUM_bind_generates_resource GL_CHROMIUM_bind_uniform_location GL_CHROMIUM_color_buffer_float_rgb GL_CHROMIUM_color_buffer_float_rgba GL_CHROMIUM_copy_compressed_texture GL_CHROMIUM_copy_texture GL_CHROMIUM_lose_context GL_EXT_EGL_image_external_wrap_modes GL_EXT_base_instance GL_EXT_blend_func_extended GL_EXT_blend_minmax GL_EXT_buffer_storage GL_EXT_clip_control GL_EXT_color_buffer_half_float GL_EXT_compressed_ETC1_RGB8_sub_texture GL_EXT_copy_image GL_EXT_debug_label GL_EXT_debug_marker GL_EXT_discard_framebuffer GL_EXT_disjoint_timer_query GL_EXT_draw_buffers GL_EXT_draw_elements_base_vertex GL_EXT_float_blend GL_EXT_frag_depth GL_EXT_instanced_arrays GL_EXT_map_buffer_range GL_EXT_memory_object

	GL_EXT_memory_object_fd GL_EXT_multi_draw_indirect GL_EXT_multisample_compatibility GL_EXT_occlusion_query_boolean GL_EXT_read_format_bgra GL_EXT_robustness GL_EXT_sRGB GL_EXT_sRGB_write_control GL_EXT_semaphore GL_EXT_semaphore_fd GL_EXT_separate_shader_objects GL_EXT_shader_non_constant_global_initializers GL_EXT_shader_texture_lod GL_EXT_shadow_samplers GL_EXT_texture_border_clamp GL_EXT_texture_compression_bptc GL_EXT_texture_compression_dxt1 GL_EXT_texture_compression_rgtc GL_EXT_texture_compression_s3tc_srgb GL_EXT_texture_filter_anisotropic GL_EXT_texture_format_BGRA8888 GL_EXT_texture_norm16 GL_EXT_texture_rg GL_EXT_texture_sRGB_decode GL_EXT_texture_storage GL_EXT_texture_type_2_10_10_10_REV GL_EXT_unpack_subimage GL_KHR_debug GL_KHR_texture_compression_astc_ldr GL_NV_depth_buffer_float2 GL_NV_fence GL_NV_framebuffer_blit GL_NV_pack_subimage GL_NV_pixel_buffer_object GL_NV_read_depth GL_NV_read_depth_stencil GL_NV_read_stencil GL_OES_EGL_image GL_OES_EGL_image_external GL_OES_EGL_sync GL_OES_compressed_EAC_R11_signed_texture GL_OES_compressed_EAC_R11_unsigned_texture GL_OES_compressed_EAC_RG11_signed_texture GL_OES_compressed_EAC_RG11_unsigned_texture GL_OES_compressed_ETC1_RGB8_texture GL_OES_compressed_ETC2_RGB8_texture GL_OES_compressed_ETC2_RGBA8_texture GL_OES_compressed_ETC2_punchthroughA_RGBA8_texture GL_OES_compressed_ETC2_punchthroughA_sRGB8_alpha_texture GL_OES_compressed_ETC2_sRGB8_alpha8_texture GL_OES_compressed_ETC2_sRGB8_texture GL_OES_depth24 GL_OES_depth32 GL_OES_depth_texture GL_OES_depth_texture_cube_map GL_OES_draw_elements_base_vertex GL_OES_element_index_uint GL_OES_fbo_render_mipmap GL_OES_get_program_binary GL_OES_mapbuffer GL_OES_packed_depth_stencil GL_OES_primitive_bounding_box GL_OES_rgb8_rgba8 GL_OES_sample_shading GL_OES_standard_derivatives GL_OES_surfaceless_context GL_OES_texture_3D GL_OES_texture_border_clamp GL_OES_texture_float GL_OES_texture_float_linear GL_OES_texture_half_float GL_OES_texture_half_float_linear GL_OES_texture_npot GL_OES_texture_stencil8 GL_OES_vertex_array_object GL_OES_vertex_half_float GL_QCOM_shading_rate
<b>Disabled Extensions</b>	GL_KHR_blend_equation_advanced GL_KHR_blend_equation_advanced_coherent
<b>Disabled WebGL Extensions</b>	
<b>Window system binding vendor</b>	Google Inc. (Intel)
<b>Window system binding version</b>	1.5 (ANGLE 2.1.0 git hash: unknown hash)
<b>Window system binding extensions</b>	EGL_EXT_create_context_robustness EGL_ANGLE_surface_orientation EGL_KHR_create_context EGL_KHR_image EGL_KHR_image_base


	EGL_EXT_image_gl_colorspace EGL_KHR_gl_colorspace EGL_KHR_gl_texture_2D_image EGL_KHR_gl_texture_cubemap_image EGL_KHR_gl_texture_3D_image EGL_KHR_gl_renderbuffer_image EGL_KHR_get_all_proc_addresses EGL_KHR_fence_sync EGL_KHR_wait_sync EGL_ANGLE_create_context_webgl_compatibility EGL_CHROMIUM_create_context_bind_generates_resource EGL_KHR_swap_buffers_with_damage EGL_EXT_pixel_format_float EGL_KHR_surfaceless_context EGL_ANGLE_display_texture_share_group EGL_ANGLE_display_semaphore_share_group EGL_ANGLE_create_context_client_arrays EGL_ANGLE_program_cache_control EGL_ANGLE_robust_resource_initialization EGL_ANGLE_create_context_extensions_enabled EGL_ANDROID_blob_cache EGL_ANDROID_recordable EGL_ANGLE_create_context_backwards_compatible EGL_KHR_no_config_context EGL_IMG_context_priority EGL_KHR_create_context_no_error EGL_EXT_image_dma_buf_import EGL_EXT_image_dma_buf_import_modifiers EGL_KHR_reusable_sync EGL_EXT_buffer_age EGL_ANGLE_create_surface_swap_interval EGL_ANGLE_vulkan_image EGL_KHR_partial_update
XDG_CURRENT_DESKTOP	MATE
XDG_SESSION_TYPE	x11
GDMSESSION	mate
Ozone platform	x11
Direct rendering version	unknown
Reset notification strategy	0x8252
GPU process crash count	0
gfx::BufferFormats supported for allocation and texturing	R_8: not supported, R_16: not supported, RG_88: not supported, RG_1616: not supported, BGR_565: not supported, RGBA_4444: not supported, RGBX_8888: not supported, RGBA_8888: not supported, BGRX_8888: not supported, BGRA_1010102: not supported, RGBA_1010102: not supported, BGRA_8888: not supported, RGBA_F16: not supported, YVU_420: not supported, YUV_420_BIPLANAR: not supported, YUVA_420_TRIPLANAR: not supported, P010: not supported

Compositor Information

Tile Update Mode	One-copy
Partial Raster	Enabled

GpuMemoryBuffers Status

R_8	Software only
R_16	Software only
RG_88	Software only
RG_1616	Software only
BGR_565	Software only
RGBA_4444	Software only
RGBX_8888	Software only

<b>RGBA_8888</b>	Software only
<b>BGRX_8888</b>	Software only
<b>BGRA_1010102</b>	Software only
<b>RGBA_1010102</b>	Software only
<b>BGRA_8888</b>	Software only
<b>RGBA_F16</b>	Software only
<b>YVU_420</b>	Software only
<b>YUV_420_BIPLANAR</b>	Software only
<b>YUVA_420_TRIPLANAR</b>	Software only
	
<b>P010</b>	Software only

## Display(s) Information

<b>Info</b>	Display[0] bounds=[0,0 1920x1200], workarea=[0,0 1920x1169], scale=1, rotation=0, panel_rotation=0 external.
<b>Color space (all)</b>	{primaries:BT709, transfer:SRGB, matrix:RGB, range:FULL}
<b>Buffer format (all)</b>	BGRA_8888
<b>Color volume</b>	{name:'srgb', r:[0.6400, 0.3300], g:[0.3000, 0.6000], b:[0.1500, 0.3300], w:[0.3127, 0.3290]}
<b>SDR white level in nits</b>	203
<b>HDR relative maximum luminance</b>	1
<b>Bits per color component</b>	8
<b>Bits per pixel</b>	24

## Video Acceleration Information

<b>Decoding</b>	
<b>Encoding</b>	

## Vulkan Information

<b>info</b>	{ "apiVersion": "1.3.237", "usedApiVersion": "1.1.0", "instanceExtensions": { "VK_KHR_device_group_creation": 1, "VK_KHR_external_fence_capabilities": 1, "VK_KHR_external_memory_capabilities": 1, "VK_KHR_external_semaphore_capabilities": 1, "VK_KHR_get_physical_device_properties2": 2, "VK_KHR_get_surface_capabilities2": 1, "VK_KHR_surface": 25, "VK_KHR_surface_protected_capabilities": 1, "VK_KHR_wayland_surface": 6, "VK_KHR_xcb_surface": 6, "VK_EXT_debug_report": 10, "VK_EXT_debug_utils": 2, "VK_KHR_display": 23, "VK_KHR_get_display_properties2": 1, "VK_EXT_acquire_drm_display": 1, "VK_EXT_direct_mode_display": 1, "VK_EXT_display_surface_counter": 1, "VK_KHR_portability_enumeration": 1 }, "enabledInstanceExtensions": [ "VK_EXT_debug_report", "VK_KHR_external_fence_capabilities", "VK_KHR_external_semaphore_capabilities", "VK_KHR_get_physical_device_properties2", "VK_KHR_get_surface_capabilities2", "VK_KHR_surface", "VK_KHR_surface_protected_capabilities", "VK_KHR_xcb_surface" ], "instanceLayers": [ { "layerName": "VK_LAYER_VALVE_steam_fossilize_64", "specVersion": 4206799, "implementationVersion": "0.0.1", "description": "Steam Pipeline Caching Layer" }, { "layerName": "VK_LAYER_VALVE_steam_fossilize_32", "specVersion": 4206799, "implementationVersion": "0.0.1", "description": "Steam Pipeline
-------------	--



```
Caching Layer" }, { "layerName":  
"VK_LAYER_VALVE_steam_overlay_32", "specVersion": 4206799,  
"implementationVersion": "0.0.1", "description": "Steam Overlay Layer"  
}, { "layerName": "VK_LAYER_VALVE_steam_overlay_64",  
"specVersion": 4206799, "implementationVersion": "0.0.1",  
"description": "Steam Overlay Layer" }, { "layerName":  
"VK_LAYER_MESA_device_select", "specVersion": 4206803,  
"implementationVersion": "0.0.1", "description": "Linux device selection  
layer" }, { "layerName": "VK_LAYER_MESA_overlay", "specVersion":  
4206803, "implementationVersion": "0.0.1", "description": "Mesa  
Overlay layer" } ], "physicalDevices": [ { "properties": { "apiVersion":  
"1.3.230", "driverVersion": "22.3.6", "vendorID": 32902, "deviceID":  
18086, "deviceType": 1, "deviceName": "Intel(R) Graphics (ADL GT2)",  
"pipelineCacheUUID": "dddf85b3-eef3-b44c-b6ea-27e2ff6b5b01",  
"limits": { "maxImageDimension1D": 16384, "maxImageDimension2D":  
16384, "maxImageDimension3D": 2048, "maxImageDimensionCube":  
16384, "maxImageArrayLayers": 2048, "maxTexelBufferElements":  
134217728, "maxUniformBufferRange": 1073741824,  
"maxStorageBufferRange": 1073741824, "maxPushConstantsSize":  
128, "maxMemoryAllocationCount": 4294967295,  
"maxSamplerAllocationCount": 65536, "bufferImageGranularity": 1,  
"sparseAddressSpaceSize": 0, "maxBoundDescriptorSets": 32,  
"maxPerStageDescriptorSamplers": 65535,  
"maxPerStageDescriptorUniformBuffers": 64,  
"maxPerStageDescriptorStorageBuffers": 65535,  
"maxPerStageDescriptorSampledImages": 65535,  
"maxPerStageDescriptorStorageImages": 65535,  
"maxPerStageDescriptorInputAttachments": 64,  
"maxPerStageResources": 4294967295, "maxDescriptorSetSamplers":  
393210, "maxDescriptorSetUniformBuffers": 384,  
"maxDescriptorSetUniformBuffersDynamic": 8,  
"maxDescriptorSetStorageBuffers": 393210,  
"maxDescriptorSetStorageBuffersDynamic": 8,  
"maxDescriptorSetSampledImages": 393210,  
"maxDescriptorSetStorageImages": 393210,  
"maxDescriptorSetInputAttachments": 256, "maxVertexInputAttributes":  
29, "maxVertexInputBindings": 31, "maxVertexInputAttributeOffset":  
2047, "maxVertexInputBindingStride": 4095,  
"maxVertexOutputComponents": 128,  
"maxTessellationGenerationLevel": 64, "maxTessellationPatchSize": 32,  
"maxTessellationControlPerVertexInputComponents": 128,  
"maxTessellationControlPerVertexOutputComponents": 128,  
"maxTessellationControlPerPatchOutputComponents": 128,  
"maxTessellationControlTotalOutputComponents": 2048,  
"maxTessellationEvaluationInputComponents": 128,  
"maxTessellationEvaluationOutputComponents": 128,  
"maxGeometryShaderInvocations": 32,  
"maxGeometryInputComponents": 128,  
"maxGeometryOutputComponents": 128,  
"maxGeometryOutputVertices": 256,  
"maxGeometryTotalOutputComponents": 1024,  
"maxFragmentInputComponents": 116,  
"maxFragmentOutputAttachments": 8,  
"maxFragmentDualSrcAttachments": 1,  
"maxFragmentCombinedOutputResources": 131078,  
"maxComputeSharedMemorySize": 65536,  
"maxComputeWorkGroupCount": [ 65535, 65535, 65535 ],  
"maxComputeWorkGroupInvocations": 1024,
```

```
"maxComputeWorkGroupSize": [ 1024, 1024, 1024 ],
"subPixelPrecisionBits": 8, "subTexelPrecisionBits": 8,
"mipmapPrecisionBits": 8, "maxDrawIndexedIndexValue": 4294967295,
"maxDrawIndirectCount": 4294967295, "maxSamplerLodBias": 16,
"maxSamplerAnisotropy": 16, "maxViewports": 16,
"maxViewportDimensions": [ 16384, 16384 ], "viewportBoundsRange": [
-32768, 32767 ], "viewportSubPixelBits": 13,
"minMemoryMapAlignment": 4096, "minTexelBufferOffsetAlignment": 1,
"minUniformBufferOffsetAlignment": 1,
"minStorageBufferOffsetAlignment": 1, "minTexelOffset": -8,
"maxTexelOffset": 7, "minTexelGatherOffset": -32,
"maxTexelGatherOffset": 31, "minInterpolationOffset": -0.5,
"maxInterpolationOffset": 0.4375, "subPixelInterpolationOffsetBits": 4,
"maxFramebufferWidth": 16384, "maxFramebufferHeight": 16384,
"maxFramebufferLayers": 2048, "framebufferColorSampleCounts": 31,
"framebufferDepthSampleCounts": 31,
"framebufferStencilSampleCounts": 31,
"framebufferNoAttachmentsSampleCounts": 31,
"maxColorAttachments": 8, "sampledImageColorSampleCounts": 31,
"sampledImageIntegerSampleCounts": 31,
"sampledImageDepthSampleCounts": 31,
"sampledImageStencilSampleCounts": 31,
"storageImageSampleCounts": 1, "maxSampleMaskWords": 1,
"timestampComputeAndGraphics": true, "timestampPeriod":
52.08333206176758, "maxClipDistances": 8, "maxCullDistances": 8,
"maxCombinedClipAndCullDistances": 8, "discreteQueuePriorities": 2,
"pointSizeRange": [ 0.125, 255.875 ], "lineWidthRange": [ 0, 8 ],
"pointSizeGranularity": 0.125, "lineWidthGranularity": 0.0078125,
"strictLines": false, "standardSampleLocations": true,
"optimalBufferCopyOffsetAlignment": 1,
"optimalBufferCopyRowPitchAlignment": 1, "nonCoherentAtomSize": 1
}, "sparseProperties": { "residencyStandard2DBlockShape": false,
"residencyStandard2DMultisampleBlockShape": false,
"residencyStandard3DBlockShape": false, "residencyAlignedMipSize":
false, "residencyNonResidentStrict": false } }, "extensions": {
"VK_KHR_8bit_storage": 1, "VK_KHR_16bit_storage": 1,
"VK_KHR_bind_memory2": 1, "VK_KHR_buffer_device_address": 1,
"VK_KHR_copy_commands2": 1, "VK_KHR_create_renderpass2": 1,
"VK_KHR_dedicated_allocation": 3,
"VK_KHR_deferred_host_operations": 4,
"VK_KHR_depth_stencil_resolve": 1,
"VK_KHR_descriptor_update_template": 1, "VK_KHR_device_group":
4, "VK_KHR_draw_indirect_count": 1, "VK_KHR_driver_properties": 1,
"VK_KHR_dynamic_rendering": 1, "VK_KHR_external_fence": 1,
"VK_KHR_external_fence_fd": 1, "VK_KHR_external_memory": 1,
"VK_KHR_external_memory_fd": 1, "VK_KHR_external_semaphore":
1, "VK_KHR_external_semaphore_fd": 1,
"VK_KHR_format_feature_flags2": 2,
"VK_KHR_fragment_shading_rate": 2,
"VK_KHR_get_memory_requirements2": 1,
"VK_KHR_image_format_list": 1, "VK_KHR_imageless_framebuffer": 1,
"VK_KHR_incremental_present": 2, "VK_KHR_maintenance1": 2,
"VK_KHR_maintenance2": 1, "VK_KHR_maintenance3": 1,
"VK_KHR_maintenance4": 2, "VK_KHR_multiview": 1,
"VK_KHR_pipeline_executable_properties": 1,
"VK_KHR_pipeline_library": 1, "VK_KHR_push_descriptor": 2,
"VK_KHR_relaxed_block_layout": 1,
"VK_KHR_sampler_mirror_clamp_to_edge": 3,
```

```
"VK_KHR_sampler_ycbcr_conversion": 14,  
"VK_KHR_separate_depth_stencil_layouts": 1,  
"VK_KHR_shader_atomic_int64": 1, "VK_KHR_shader_clock": 1,  
"VK_KHR_shader_draw_parameters": 1,  
"VK_KHR_shader_float16_int8": 1, "VK_KHR_shader_float_controls":  
4, "VK_KHR_shader_integer_dot_product": 1,  
"VK_KHR_shader_non_semantic_info": 1,  
"VK_KHR_shader_subgroup_extended_types": 1,  
"VK_KHR_shader_subgroup_uniform_control_flow": 1,  
"VK_KHR_shader_terminate_invocation": 1, "VK_KHR_spirv_1_4": 1,  
"VK_KHR_storage_buffer_storage_class": 1, "VK_KHR_swapchain":  
70, "VK_KHR_swapchain_mutable_format": 1,  
"VK_KHR_synchronization2": 1, "VK_KHR_timeline_semaphore": 2,  
"VK_KHR_uniform_buffer_standard_layout": 1,  
"VK_KHR_variable_pointers": 1, "VK_KHR_vulkan_memory_model": 3,  
"VK_KHR_workgroup_memory_explicit_layout": 1,  
"VK_KHR_zero_initialize_workgroup_memory": 1,  
"VK_EXT_4444_formats": 1, "VK_EXT_border_color_swizzle": 1,  
"VK_EXT_buffer_device_address": 2,  
"VK_EXT_calibrated_timestamps": 2, "VK_EXT_color_write_enable":  
1, "VK_EXT_conditional_rendering": 2,  
"VK_EXT_conservative_rasterization": 1,  
"VK_EXT_custom_border_color": 12,  
"VK_EXT_depth_clamp_zero_one": 1, "VK_EXT_depth_clip_control":  
1, "VK_EXT_depth_clip_enable": 1, "VK_EXT_descriptor_indexing": 2,  
"VK_EXT_display_control": 1, "VK_EXT_extended_dynamic_state": 1,  
"VK_EXT_extended_dynamic_state2": 1,  
"VK_EXT_extended_dynamic_state3": 2,  
"VK_EXT_external_memory_dma_buf": 1,  
"VK_EXT_external_memory_host": 1,  
"VK_EXT_fragment_shader_interlock": 1, "VK_EXT_global_priority": 2,  
"VK_EXT_global_priority_query": 1, "VK_EXT_host_query_reset": 1,  
"VK_EXT_image_2d_view_of_3d": 1,  
"VK_EXT_image_drm_format_modifier": 2,  
"VK_EXT_image_robustness": 1, "VK_EXT_image_view_min_lod": 1,  
"VK_EXT_index_type_uint8": 1, "VK_EXT_inline_uniform_block": 1,  
"VK_EXT_line_rasterization": 1, "VK_EXT_memory_budget": 1,  
"VK_EXT_multi_draw": 1, "VK_EXT_mutable_descriptor_type": 1,  
"VK_EXT_non_seamless_cube_map": 1, "VK_EXT_pci_bus_info": 2,  
"VK_EXT_physical_device_drm": 1,  
"VK_EXT_pipeline_creation_cache_control": 3,  
"VK_EXT_pipeline_creation_feedback": 1,  
"VK_EXT_post_depth_coverage": 1,  
"VK_EXT_primitive_topology_list_restart": 1,  
"VK_EXT_primitives_generated_query": 1, "VK_EXT_private_data": 1,  
"VK_EXT_provoking_vertex": 1, "VK_EXT_queue_family_foreign": 1,  
"VK_EXT_robustness2": 1, "VK_EXT_sample_locations": 1,  
"VK_EXT_sampler_filter_minmax": 2, "VK_EXT_scalar_block_layout":  
1, "VK_EXT_separate_stencil_usage": 1,  
"VK_EXT_shader_atomic_float": 1, "VK_EXT_shader_atomic_float2":  
1, "VK_EXT_shader_demote_to_helper_invocation": 1,  
"VK_EXT_shader_module_identifier": 1,  
"VK_EXT_shader_stencil_export": 1,  
"VK_EXT_shader_subgroup_ballot": 1,  
"VK_EXT_shader_subgroup_vote": 1,  
"VK_EXT_shader_viewport_index_layer": 1,  
"VK_EXT_subgroup_size_control": 2,  
"VK_EXT_texel_buffer_alignment": 1, "VK_EXT_tooling_info": 1,
```

```

"VK_EXT_transform_feedback": 1, "VK_EXT_vertex_attribute_divisor":
3, "VK_EXT_ycbcr_image_arrays": 1, "VK_GOOGLE_decorate_string":
1, "VK_GOOGLE_hlsl_functionality1": 1, "VK_GOOGLE_user_type": 1,
"VK_INTEL_shader_integer_functions2": 1,
"VK_NV_compute_shader_derivatives": 1,
"VK_VALVE_mutable_descriptor_type": 1 }, "features": {
"robustBufferAccess": true, "fullDrawIndexUint32": true,
"imageCubeArray": true, "independentBlend": true, "geometryShader":
true, "tessellationShader": true, "sampleRateShading": true,
"dualSrcBlend": true, "logicOp": true, "multiDrawIndirect": true,
"drawIndirectFirstInstance": true, "depthClamp": true,
"depthBiasClamp": true, "fillModeNonSolid": true, "depthBounds": true,
"wideLines": true, "largePoints": true, "alphaToOne": true,
"multiViewport": true, "samplerAnisotropy": true,
"textureCompressionETC2": true, "textureCompressionASTC_LDR":
true, "textureCompressionBC": true, "occlusionQueryPrecise": true,
"pipelineStatisticsQuery": true, "vertexPipelineStoresAndAtomics": true,
"fragmentStoresAndAtomics": true,
"shaderTessellationAndGeometryPointSize": true,
"shaderImageGatherExtended": true,
"shaderStorageImageExtendedFormats": true,
"shaderStorageImageMultisample": false,
"shaderStorageImageReadWithoutFormat": false,
"shaderStorageImageWriteWithoutFormat": true,
"shaderUniformBufferArrayDynamicIndexing": true,
"shaderSampledImageArrayDynamicIndexing": true,
"shaderStorageBufferArrayDynamicIndexing": true,
"shaderStorageImageArrayDynamicIndexing": true,
"shaderClipDistance": true, "shaderCullDistance": true,
"shaderFloat64": false, "shaderInt64": true, "shaderInt16": true,
"shaderResourceResidency": false, "shaderResourceMinLod": true,
"sparseBinding": false, "sparseResidencyBuffer": false,
"sparseResidencyImage2D": false, "sparseResidencyImage3D": false,
"sparseResidency2Samples": false, "sparseResidency4Samples":
false, "sparseResidency8Samples": false,
"sparseResidency16Samples": false, "sparseResidencyAliased": false,
"variableMultisampleRate": true, "inheritedQueries": true },
"featureSamplerYcbcrConversion": true, "featureProtectedMemory":
false, "queueFamilies": [ { "queueFlags": 7, "queueCount": 1,
"timestampValidBits": 36, "minImageTransferGranularity": { "width": 1,
"height": 1, "depth": 1 } } ] } } } } }

```

## Device Performance Information

### Log Messages

- [7328:7328:0524/173229.004548:WARNING:sandbox\_linux.cc(393)] : InitializeSandbox() called with multiple threads in process gpu-process.
- [7328:7328:0524/173229.008523:ERROR:gpu\_memory\_buffer\_support\_x11.cc(49)] : dri3 extension not supported.
- [7328:7328:0524/173229.087646:ERROR:skia\_output\_device\_vulkan.cc(283)] : Failed to initialize vulkan surface.