

Metal feature set tables

This table lists each current Apple GPU family, its processors, and how each family relates to older feature sets.

Apple GPUs

Apple GPU family ¹	GPUs in family	Corresponding feature sets
Apple2	A8	iOS GPU Family 2 tvOS GPU Family 1
Apple3	A9, A10	iOS GPU Family 3 tvOS GPU Family 2
Apple4	A11	iOS GPU Family 4
Apple5	A12	iOS GPU Family 5
Apple6	A13	—
Apple7	A14 M1, M1 Pro, M1 Max, M1 Ultra	—
Apple8	A15 M2	—

1. See [MTLGPUPFamily](#) for each GPU family's enumeration constant.

This table lists each current Metal 3 GPU family and the processors in that family.

Metal 3 GPUs

Metal GPU family ¹	Platform	GPUs in family
Metal3	iOS	A13, A14, A15
	iPadOS	A13, A14, A15 M1
	macOS	M1, M1 Pro, M1 Max, M1 Ultra M2 AMD Vega AMD 5000-series, 6000-series Intel UHD Graphics 630 Intel Iris Pro Graphics

1. See [MTLGPUPFamily](#) for each GPU family's enumeration constant.

Metal feature availability by GPU family

GPU family ¹	Common1	Common2	Common3	Meta13	Apple2	Apple3	Apple4	Apple5	Apple6	Apple7	Apple8	Mac2
Feature	Available in family											
MetalKit	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Metal Performance Shaders		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Programmable blending					✓	✓	✓	✓	✓	✓	✓	
PVRTC pixel formats					✓	✓	✓	✓	✓	✓	✓	
EAC/ETC pixel formats					✓	✓	✓	✓	✓	✓	✓	
ASTC pixel formats					✓	✓	✓	✓	✓	✓	✓	
BC pixel formats												✓
Compressed volume texture formats		✓	✓	✓		✓	✓	✓	✓	✓	✓	✓
Extended range pixel formats						✓	✓	✓	✓	✓	✓	
Wide color pixel format	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Depth-16 pixel format	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Linear textures	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
MSAA depth resolve			✓	✓		✓	✓	✓	✓	✓	✓	✓
Array of textures (read)		✓	✓	✓		✓	✓	✓	✓	✓	✓	✓
Array of textures (write)		✓	✓	✓					✓	✓	✓	✓
Cube map texture arrays		✓	✓	✓			✓	✓	✓	✓	✓	✓
Stencil texture views	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Array of samplers		✓	✓	✓		✓	✓	✓	✓	✓	✓	✓
Sampler max anisotropy	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Sampler LOD clamp	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
MTLSamplerState support for comparison functions		✓	✓	✓		✓	✓	✓	✓	✓	✓	✓
16-bit unsigned integer coordinates	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Border color				✓						✓	✓	✓
Counting occlusion query		✓	✓	✓		✓	✓	✓	✓	✓	✓	✓
Base vertex/instance drawing		✓	✓	✓		✓	✓	✓	✓	✓	✓	✓
Layered rendering			✓	✓				✓	✓	✓	✓	✓
Layered rendering to multisample textures			✓	✓						✓	✓	✓
Memoryless render targets					✓	✓	✓	✓	✓	✓	✓	
Dual-source blending	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Combined MSAA store and resolve action		✓	✓	✓		✓	✓	✓	✓	✓	✓	✓
MSAA blits	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Programmable sample positions	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Deferred store action	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Texture barriers												✓
Memory barriers ²				✓		✓	✓	✓	✓	✓	✓	✓
Tessellation		✓	✓	✓		✓	✓	✓	✓	✓	✓	✓

GPU family ¹	Common1	Common2	Common3	Meta13	Apple2	Apple3	Apple4	Apple5	Apple6	Apple7	Apple8	Mac2
Indirect tessellation arguments				✓				✓	✓	✓	✓	✓
Tessellation in indirect command buffers				✓				✓	✓	✓	✓	✓
Resource heaps	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Function specialization	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Read/write buffers in functions		✓	✓	✓		✓	✓	✓	✓	✓	✓	✓
Read/write textures in functions			✓	✓			✓	✓	✓	✓	✓	✓
Extract, insert, and reverse bits	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
SIMD barrier	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Indirect draw & dispatch arguments		✓	✓	✓		✓	✓	✓	✓	✓	✓	✓
Argument buffers tier	Varies	Varies	Varies	Tier 2	Tier 1	Tier 1	Tier 1	Tier 1	Tier 2	Tier 2	Tier 2	Tier 2
Indirect command buffers (rendering)		✓	✓	✓		✓	✓	✓	✓	✓	✓	✓
Indirect command buffers (compute)		✓	✓	✓		✓	✓	✓	✓	✓	✓	
Uniform type	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Imageblocks							✓	✓	✓	✓	✓	
Tile shaders							✓	✓	✓	✓	✓	
Imageblock sample coverage control							✓	✓	✓	✓	✓	
Post-depth coverage							✓	✓	✓	✓	✓	
Quad-scoped permute operations			✓	✓			✓	✓	✓	✓	✓	✓
SIMD-scoped permute operations				✓					✓	✓	✓	✓
SIMD-scoped reduction operations				✓						✓	✓	✓
SIMD-scoped matrix multiply operations										✓	✓	
Raster order groups ³			✓	✓			✓	✓	✓	✓	✓	Varies
Non-uniform threadgroup size			✓	✓			✓	✓	✓	✓	✓	✓
Multiple viewports			✓	✓				✓	✓	✓	✓	✓
Device notifications												✓
Stencil feedback			✓	✓				✓	✓	✓	✓	✓
Stencil resolve			✓	✓				✓	✓	✓	✓	✓
Non-square tile dispatch								✓	✓	✓	✓	
Texture swizzle				✓	✓	✓	✓	✓	✓	✓	✓	✓
Placement heap				✓	✓	✓	✓	✓	✓	✓	✓	✓
Primitive ID				✓						✓	✓	✓
Barycentric coordinates ⁴				Varies						✓	✓	Varies
Read/write cube map textures in functions				✓			✓	✓	✓	✓	✓	✓
Sparse textures									✓	✓	✓	
Sparse depth and stencil textures											✓	
Variable rasterization rate ⁵									✓	✓	✓	Varies
Vertex amplification ⁶									✓	✓	✓	Varies
64-bit integer math				✓		✓	✓	✓	✓	✓	✓	
Lossy texture compression											✓	

GPU family ¹	Common1	Common2	Common3	Metal3	Apple2	Apple3	Apple4	Apple5	Apple6	Apple7	Apple8	Mac2
SIMD shift and fill											✓	
Render dynamic libraries									✓	✓	✓	
Compute dynamic libraries				✓					✓	✓	✓	✓
Mesh shading				✓						✓	✓	✓
MetalFX spatial upscaling				Varies						Varies	Varies	✓
MetalFX temporal upscaling				Varies						Varies	Varies	
Fast resource loading	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Ray tracing in compute pipelines				✓					✓	✓	✓	✓
Ray tracing in render pipelines									✓	✓	✓	
Floating point atomics										✓	✓	✓

1. See [MTLGPUPFamily](#) for each GPU family's enumeration constant.
2. The GPUs in Apple3 through Apple8 families only support memory barriers for compute command encoders, and for vertex-to-vertex and vertex-to-fragment stages of render command encoders.
3. Some GPU devices in Mac families support raster order groups. You can check an individual GPU's support for this feature by inspecting its [MTLDevice.rasterOrderGroupsSupported](#) property at runtime.
4. Some GPU devices in the Mac 2 and Metal 3 families support barycentric coordinates. You can check an individual GPU's support for this feature by inspecting its [MTLDevice.supportsShaderBarycentricCoordinates](#) property at runtime.
5. Some GPU devices in the Mac 2 family support variable rasterization rates. You can check an individual GPU's support for this feature by calling its [MTLDevice.supportsRasterizationRateMap\(layerCount:\)](#) method at runtime.
6. Some Mac 2 family GPU devices support vertex amplification. You can check an individual GPU's support for this feature by calling its [MTLDevice.supportsVertexAmplificationCount\(_:\)](#) method at runtime.

Implementation limits by GPU family

GPU family ¹	Apple2	Apple3	Apple4	Apple5	Apple6	Apple7	Apple8	Mac2
Function arguments	Function arguments							
Maximum number of vertex attributes, per vertex descriptor	31	31	31	31	31	31	31	31
Maximum number of entries in the buffer argument table, per graphics or kernel function	31	31	31	31	31	31	31	31
Maximum number of entries in the texture argument table, per graphics or kernel function	31	31	96	96	128	128	128	128
Maximum number of entries in the sampler state argument table, per graphics or kernel function ²	16	16	16	16	16	16	16	16
Maximum number of entries in the threadgroup memory argument table, per kernel function	31	31	31	31	31	31	31	31
Maximum number of constant buffer arguments in vertex, fragment, tile, or kernel function	31	31	31	31	31	31	31	14
Maximum length of constant buffer arguments in vertex, fragment, tile, or kernel function	4 KB	4 KB	4 KB	4 KB	4 KB	4 KB	4 KB	4 KB
Maximum threads per threadgroup ³	512	512	1024	1024	1024	1024	1024	1024
Maximum total threadgroup memory allocation	16352 B	16 KB	32 KB	32 KB	32 KB	32 KB	32 KB	32 KB
Maximum total tile memory allocation ⁴	Not accessible	Not accessible	32 KB	32 KB	32 KB	32 KB	32 KB	Not accessible
Threadgroup memory length alignment	16 B	16 B	16 B	16 B	16 B	16 B	16 B	16 B
Maximum function memory allocation for a buffer in the constant address space	No limit	No limit	No limit	No limit	No limit	No limit	No limit	No limit
Maximum scalars or vectors inputs to a fragment function. (Declare with the <code>[[stage_in]]</code> qualifier ⁴ .)	60	60	124	124	124	124	124	32
Maximum number of input components to a fragment function. (Declare with the <code>[[stage_in]]</code> qualifier ⁵ .)	60	60	124	124	124	124	124	124
Maximum number of function constants	65536	65536	65536	65536	65536	65536	65536	65536
Maximum tessellation factor	Not available	16	16	64	64	64	64	64

GPU family ¹	Apple2	Apple3	Apple4	Apple5	Apple6	Apple7	Apple8	Mac2
Maximum number of viewports and scissor rectangles, per vertex function	1	1	1	16	16	16	16	16
Maximum number of raster order groups, per fragment function	Not available	Not available	8	8	8	8	8	8
Argument buffers ⁶	Argument buffers ⁶							
Maximum number of buffers you can access, per stage, from an argument buffer	31	31	96	96	Unlimited	Unlimited	Unlimited	Unlimited
Maximum number of textures you can access, per stage, from an argument buffer	31	31	96	96	1 M	1 M	1 M	1 M
Maximum number of samplers you can access, per stage, from an argument buffer	16	16	16	16	1024	1024	1024	1024
Resources	Resources							
Minimum constant buffer offset alignment	4 B	4 B	4 B	4 B	4 B	4 B	4 B	32 B
Maximum 1D texture width	8192 px	16384 px	16384 px	16384 px	16384 px	16384 px	16384 px	16384 px
Maximum 2D texture width and height	8192 px	16384 px	16384 px	16384 px	16384 px	16384 px	16384 px	16384 px
Maximum cube map texture width and height	8192 px	16384 px	16384 px	16384 px	16384 px	16384 px	16384 px	16384 px
Maximum 3D texture width, height, and depth	2048 px	2048 px	2048 px	2048 px	2048 px	2048 px	2048 px	2048 px
Maximum number of layers per 1D texture array, 2D texture array, or 3D texture	2048	2048	2048	2048	2048	2048	2048	2048
Buffer alignment for copying an existing texture to a buffer	64 B	16 B	16 B	16 B	16 B	16 B	16 B	256 B
Render targets	Render targets							
Maximum number of color render targets per render pass descriptor	8	8	8	8	8	8	8	8
Maximum size of a point primitive	54	54	54	54	54	54	54	54
Maximum total render target size, per pixel, when using multiple color render targets	256 bits	256 bits	512 bits	512 bits	512 bits	512 bits	512 bits	No limit

GPU family ¹	Apple2	Apple3	Apple4	Apple5	Apple6	Apple7	Apple8	Mac2
Maximum visibility query offset	65528 B	65528 B	65528 B	65528 B	65528 B	256 KB	256 KB	256 KB
Synchronization	Synchronization							
Maximum number of fences	32768	32768	32768	32768	32768	32768	32768	32768
Maximum vertex count for vertex amplification	Not available	Not available	Not available	Not available	2	2	2	Not available

1. See [MTLGPUPFamily](#) for each GPU family's enumeration constant.
2. Inline `constexpr` samplers that you declare in [Metal Shading Language](#) (MSL) code count against the limit. For example, for a feature set limit of 16, you can have 12 API samplers and 4 language samplers (16 total), but you can't have 12 API samplers and 6 language samplers (18 total).
3. The values in this row are the theoretical maximum number of threads per threadgroup. Check the actual maximum by inspecting the `MTLComputePipelineState.maxTotalThreadsPerThreadgroup` property at runtime.
4. You can allocate memory between `imageblock` and `threadgroup` memory, but the sum of these allocations can't exceed the maximum total tile memory limit. Some feature sets can't access tile memory directly, but they can access threadgroup memory.
5. A vector counts as n scalars, where n is the number of components in the vector. The iOS and tvOS feature sets only reach the maximum number of inputs if you don't exceed the maximum number of input components. For example, you can have 60 float inputs (components), but you can't have 60 `float4` inputs, which total 240 components.
6. The limits apply to the items you place both in the argument buffers you bind directly and in the argument buffers you can access indirectly through your bound argument buffers.

- This table lists the GPU’s texture capabilities for each pixel format:
- **All:** The GPU has all of the texture capabilities below for the pixel format.
 - **Filter:** The GPU can filter a texture with the pixel format during sampling.
 - **Write:** The GPU can write to a texture on a per-pixel basis with the pixel format.¹
 - **Color:** The GPU can use a texture with the pixel format as a color render target.
 - **Blend:** The GPU can blend a texture with the pixel format.
 - **MSAA:** The GPU can use a texture with the pixel format as a destination for multisample antialias (MSAA) data.
 - **Sparse:** The GPU supports sparse-texture allocations for textures with the pixel format.
 - **Resolve:** The GPU can use a texture with the pixel format as a source for multisample antialias (MSAA) resolve operations.

Note
All graphics and compute kernels can read or sample a texture with any pixel format.

GPU texture capabilities by pixel format

GPU family ²	Apple2	Apple3	Apple4	Apple5	Apple6	Apple7	Apple8	Mac2
Ordinary 8-bit pixel formats	Texture capabilities for ordinary 8-bit pixel formats by GPU Family							
A8Unorm	Filter	Filter	Filter	Filter	Filter Sparse	Filter Sparse	Filter Sparse	Filter
R8Unorm ¹	All	All	All	All	All	All	All	All
R8Unorm_sRGB	All	All	All	All	All	All	All	Not available
R8Snorm	All	All	All	All	All	All	All	All
R8Uint ¹ R8Sint ¹	Write Color MSAA	Write Color MSAA	Write Color MSAA	Write Color MSAA	Write Color MSAA Sparse	Write Color MSAA Sparse	Write Color MSAA Sparse	Write Color MSAA
Ordinary 16-bit pixel formats	Texture capabilities for ordinary 16-bit pixel formats by GPU family							
R16Unorm R16Snorm	Filter Write Color MSAA Blend	Filter Write Color MSAA Blend	Filter Write Color MSAA Blend	Filter Write Color MSAA Blend	Filter Write Color MSAA Blend Sparse	Filter Write Color MSAA Blend Sparse	Filter Write Color MSAA Blend Sparse	All
R16Uint ¹ R16Sint ¹	Write Color MSAA	Write Color MSAA	Write Color MSAA	Write Color MSAA	Write Color MSAA Sparse	Write Color MSAA Sparse	Write Color MSAA Sparse	Write Color MSAA
R16Float ¹	All	All	All	All	All	All	All	All
RG8Unorm	All	All	All	All	All	All	All	All
RG8Unorm_sRGB	All	All	All	All	All	All	All	Not available
RG8Snorm	All	All	All	All	All	All	All	All
RG8Uint RG8Sint	Write Color MSAA	Write Color MSAA	Write Color MSAA	Write Color MSAA	Write Color MSAA Sparse	Write Color MSAA Sparse	Write Color MSAA Sparse	Write Color MSAA
Packed 16-bit pixel formats	Texture capabilities for packed 16-bit pixel formats by GPU family							
B5G6R5Unorm A1BGR5Unorm ABGR4Unorm BGR5A1Unorm	Filter Color MSAA Resolve Blend	Filter Color MSAA Resolve Blend	Filter Color MSAA Resolve Blend	Filter Color MSAA Resolve Blend	Filter Color MSAA Resolve Blend Sparse	Filter Color MSAA Resolve Blend Sparse	Filter Color MSAA Resolve Blend Sparse	Not available

GPU family ²	Apple2	Apple3	Apple4	Apple5	Apple6	Apple7	Apple8	Mac2
Ordinary 32-bit pixel formats	Texture capabilities for ordinary 32-bit pixel formats by GPU family							
R32UInt ¹ R32Sint ¹	Write Color	Write Color	Write Color	Write Color	Write Color Sparse	Write Color Sparse	Write Color Sparse	Write Color MSAА
R32Float ¹	Write Color MSAА Blend	Write Color MSAА Blend	Write Color MSAА Blend	Write Color MSAА Blend	Write Color MSAА Blend Sparse	Write Color MSAА Blend Sparse	Write Color MSAА Blend Sparse	All
RG16Unorm RG16Snorm	Filter Write Color MSAА Blend	Filter Write Color MSAА Blend	Filter Write Color MSAА Blend	Filter Write Color MSAА Blend	Filter Write Color MSAА Blend Sparse	Filter Write Color MSAА Blend Sparse	Filter Write Color MSAА Blend Sparse	All
RG16UInt RG16Sint	Write Color MSAА	Write Color MSAА	Write Color MSAА	Write Color MSAА	Write Color MSAА Sparse	Write Color MSAА Sparse	Write Color MSAА Sparse	Write Color MSAА
RG16Float	All	All	All	All	All	All	All	All
RGBA8Unorm ¹	All	All	All	All	All	All	All	All
RGBA8Unorm_sRGB	All	All	All	All	All	All	All	Filter Color MSAА Resolve Blend
RGBA8Snorm	All	All	All	All	All	All	All	All
RGBA8UInt ¹ RGBA8Sint ¹	Write Color MSAА	Write Color MSAА	Write Color MSAА	Write Color MSAА	Write Color MSAА Sparse	Write Color MSAА Sparse	Write Color MSAА Sparse	Write Color MSAА
BGRA8Unorm	All	All	All	All	All	All	All	All
BGRA8Unorm_sRGB	All	All	All	All	All	All	All	Filter Color MSAА Resolve Blend
Packed 32-bit pixel formats	Texture capabilities for packed 32-bit pixel formats by GPU family							
RGB10A2Unorm	Filter Color MSAА Resolve Blend	All	All	All	All	All	All	All

GPU family ²	Apple2	Apple3	Apple4	Apple5	Apple6	Apple7	Apple8	Mac2
RGB10A2Uint	Color MSAA	Write Color MSAA	Write Color MSAA	Write Color MSAA	Write Color MSAA Sparse	Write Color MSAA Sparse	Write Color MSAA Sparse	Write Color MSAA
RG4B10Float	Filter Color MSAA Resolve Blend	All	All	All	All	All	All	All
RGB9E5Float	Filter Color MSAA Resolve Blend	All	All	All	All	All	All	Filter
Ordinary 64-bit pixel formats	Texture capabilities for ordinary 64-bit pixel formats by GPU family							
RG32Uint RG32Sint	Write Color	Write Color	Write Color	Write Color	Write Color Sparse	Write Color MSAA Sparse	Write Color MSAA Sparse	Write Color MSAA
RG32Float	Write Color Blend	Write Color Blend	Write Color Blend	Write Color Blend	Write Color Blend Sparse	Write Color MSAA Blend Sparse	Write Color MSAA Blend Sparse	All
RGBA16Unorm RGBA16Snorm	Filter Write Color MSAA Blend	Filter Write Color MSAA Blend	Filter Write Color MSAA Blend	Filter Write Color MSAA Blend	Filter Write Color MSAA Blend Sparse	Filter Write Color MSAA Blend Sparse	Filter Write Color MSAA Blend Sparse	All
RGBA16Uint ¹ RGBA16Sint ¹	Write Color MSAA	Write Color MSAA	Write Color MSAA	Write Color MSAA	Write Color MSAA Sparse	Write Color MSAA Sparse	Write Color MSAA Sparse	Write Color MSAA
RGBA16Float ¹	All	All	All	All	All	All	All	All
Ordinary 128-bit pixel formats	Texture capabilities for ordinary 128-bit pixel formats by GPU family							
RGBA32Uint ¹ RGBA32Sint ¹	Write Color	Write Color	Write Color	Write Color	Write Color Sparse	Write Color Sparse	Write Color Sparse	Write Color MSAA
RGBA32Float ¹	Write Color	Write Color	Write Color	Write Color	Write Color Sparse	Write Color MSAA Sparse	Write Color MSAA Sparse	All
Compressed pixel formats	Texture capabilities for compressed pixel formats by GPU family							
PVRTC pixel formats ³	Filter	Filter	Filter	Filter	Filter Sparse	Filter Sparse	Filter Sparse	Not available
EAC/ETC pixel formats	Filter	Filter	Filter	Filter	Filter Sparse	Filter Sparse	Filter Sparse	Not available
ASTC pixel formats	Filter	Filter	Filter	Filter	Filter Sparse	Filter Sparse	Filter Sparse	Not available
HDR ASTC pixel formats	Not available	Not available	Not available	Not available	Filter Sparse	Filter Sparse	Filter Sparse	Not available
BC pixel formats	Not available	Not available	Not available	Not available	Not available	Not available	Not available	Filter
YUV pixel formats⁴	Texture capabilities for YUV pixel formats by GPU family							
GBGR422 BGRG422	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter

GPU family ²	Apple2	Apple3	Apple4	Apple5	Apple6	Apple7	Apple8	Mac2
Depth and stencil pixel formats	Texture capabilities for depth and stencil pixel formats by GPU family							
Depth16Unorm	Filter MSAA	Filter MSAA Resolve	Filter MSAA Resolve	Filter MSAA Resolve	Filter MSAA Resolve	Filter MSAA Resolve	Filter MSAA Resolve Sparse	Filter MSAA Resolve
Depth32Float	MSAA	MSAA Resolve	MSAA Resolve	MSAA Resolve	MSAA Resolve	MSAA Resolve	MSAA Resolve Sparse	Filter MSAA Resolve
Stencil8	MSAA	MSAA	MSAA	MSAA	MSAA	MSAA	MSAA Sparse	MSAA
Depth24Unorm_Stencil8 ⁵	Not available	Not available	Not available	Not available	Not available	Not available	Not available	Filter MSAA Resolve
Depth32Float_Stencil8	MSAA	MSAA Resolve	MSAA Resolve	MSAA Resolve	MSAA Resolve	MSAA Resolve	MSAA Resolve	Filter MSAA Resolve
X24_Stencil8	Not available	Not available	Not available	Not available	Not available	Not available	Not available	MSAA
X32_Stencil8	MSAA	MSAA	MSAA	MSAA	MSAA	MSAA	MSAA	MSAA
Extended range and wide color pixel formats	Texture capabilities for extended range and wide color formats by GPU family							
BGRA10_XR BGRA10_XR_sRGB BGR10_XR BGR10_XR_sRGB	Not available	All	All	All	All	All	All	Not available
BGR10A2Unorm	All	All	All	All	All	All	All	All

1. Some GPUs support read-write textures — where a kernel can both read from and write to a texture. You can check an individual GPU’s support for this feature by inspecting its `MTLDevice.readWriteTextureSupport` property at runtime.

2. See `MTLGPUFamily` for each GPU family’s enumeration constant.

3. Only the GPUs in Apple 3 and Apple 4 families support `MTLSamplerAddressMode.clampToZero` for the PVRTC pixel formats.

4. The GPUs in Apple 6 through Apple 8 families don’t support sparse textures with YUV pixel formats.

5. Some GPUs support `MTLPixelFormat.depth24Unorm_stencil8`. You can check an individual GPU’s support for this pixel format by inspecting its `MTLDevice.isDepth24Stencil8PixelFormatSupported` property at runtime.

Texture buffer pixel formats

These tables list the pixel formats that texture buffers support and the GPU's read/write access to textures with those formats:

- **All:** The GPU can use all the accesses below for a texture in the pixel format.
- **Read:** The GPU can use read access for a texture buffer with the pixel format.
- **Write:** The GPU can use write access for a texture buffer with the pixel format.
- **Read/Write¹:** The GPU can use read_write access for a texture buffer with the pixel format.

Note

The GPU capabilities are generally the same across all hardware families, but some GPUs have additional options.²

Ordinary 8-bit pixel formats		Ordinary 32-bit pixel formats		Packed 32-bit pixel formats	
Format	Access	Format	Access	Format	Access
A8Unorm	Read	R32Uint R32Sint	All	RGB10A2Unorm	Read Write
R8Unorm	All	R32Float	All	RGB10A2Uint	Read Write
R8Snorm	Read Write	RG16Unorm RG16Snorm	Read Write	RG4B10Float	Read Write
R8Uint R8Sint	All	RG16Uint RG16Sint	Read Write	Ordinary 64-bit pixel formats	
Ordinary 16-bit pixel formats		RG16Float	Read Write	Format	Access
Format	Access	RGBA8Unorm	All	RG32Uint RG32Sint	Read Write
R16Unorm R16Snorm	Read Write	RGBA8Snorm	Read Write	RG32Float	Read Write
R16Uint R16Sint	All	RGBA8Uint RGBA8Sint	All	RGBA16Unorm RGBA16Snorm	Read Write
R16Float	All	BGRA8Unorm	Read	RGBA16Uint RGBA16Sint	All
RG8Unorm	Read Write	Ordinary 128-bit pixel formats		RGBA16Float	All
RG8Snorm	Read Write			Format	Access
RG8Uint RG8Sint	Read Write			RGBA32Uint RGBA32Sint	All
				RGBA32Float	All

1. GPUs with the Tier 2 feature set support read_write access to textures. You can check an individual GPU's support for this pixel format by inspecting its `MTLDevice.readWriteTextureSupport` property at runtime.

2. Some devices support this pixel format. Check a device by inspecting its `MTLDevice.depth24Stencil18PixelFormatSupported` property at runtime.



Apple Inc.
Copyright © 2022 Apple Inc.
All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, without prior written permission of Apple Inc., with the following exceptions: Any person is hereby authorized to store documentation on a single computer or device for personal use only and to print copies of documentation for personal use provided that the documentation contains Apple's copyright notice.

No licenses, express or implied, are granted with respect to any of the technology described in this document. Apple retains all intellectual property rights associated with the technology described in this document. This document is intended to assist application developers to develop applications only for Apple-branded products.

Apple Inc.
One Apple Park Way
Cupertino, CA 95014

Apple is a trademark of Apple Inc., registered in the U.S. and other countries.

APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS DOCUMENT IS PROVIDED "AS IS," AND YOU, THE READER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.

IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT, ERROR OR INACCURACY IN THIS DOCUMENT, even if advised of the possibility of such damages.

Some jurisdictions do not allow the exclusion of implied warranties or liability, so the above exclusion may not apply to you.