GPU family [1]	Metal3	Metal4	Apple2	Apple3	Apple4	Apple5	Apple6	Apple7	Apple8	Apple9	Mac2
GBGR422 BGRG422	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
Depth and stencil pixel formats [7]	Texture capabilities for **depth and stencil pixel formats** by GPU family										
Depth16Unorm	Filter MSAA Resolve	Filter MSAA Resolve Sparse [11]	Filter MSAA	Filter MSAA Resolve	Filter MSAA Resolve	Filter MSAA Resolve	Filter MSAA Resolve	Filter MSAA Resolve Sparse [11]	Filter MSAA Resolve Sparse	Filter MSAA Resolve Sparse	Filter MSAA Resolve
Depth32Float [6]	MSAA Resolve	MSAA Resolve Sparse [11]	MSAA	MSAA Resolve	MSAA Resolve	MSAA Resolve	MSAA Resolve	MSAA Resolve Sparse [11]	MSAA Resolve Sparse	Filter MSAA Resolve Sparse	Filter MSAA Resolve
Stencil8	Not available	MSAA Resolve Sparse [11]	MSAA	MSAA Resolve	MSAA Resolve	MSAA Resolve	MSAA Resolve	MSAA Resolve Sparse [11]	MSAA Resolve Sparse	MSAA Resolve Sparse	Not available
Depth24Unorm_Stencil8 [5]	Not available	Not available	Not available	Not available	Not available	Not available	Not available	Not available	Not available	Not available	Filter MSAA Resolve
Depth32Float_Stencil8	MSAA Resolve	MSAA Resolve	MSAA	MSAA Resolve	MSAA Resolve	MSAA Resolve	MSAA Resolve	MSAA Resolve	MSAA Resolve	Filter MSAA Resolve	Filter MSAA Resolve
X24_Stencil8	Not available	Not available	Not available	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	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	Not available	All	All	All	All	All	All	All	Not available

1. See `MTLGPUFamily` for each GPU family's enumeration constant.
2. 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.
3. Only the GPUs in `Apple3` and `Apple4` families support `MTLSamplerAddressMode.clampToZero` for the PVRTC pixel formats.
4. The GPUs in `Apple6` through `Apple9` 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 feature by inspecting its `MTLDevice.isDepth24Stencil8PixelFormatSupported` property at runtime.
6. Some GPUs in the `Apple7`, and `Apple8` families additionally support the **Filter** and **Resolve** texture capabilities for 32-bit floating-point pixel formats in iPadOS. You can check an individual GPU's support for this feature by inspecting the `MTLDevice.supports32BitFloatFiltering` property at runtime.
7. Formats in this group aren't compatible with lossy texture compression through `MTLTextureDescriptor.compressionType`.
8. Some GPU devices in the `Apple7` and `Apple8` families support filtering and sparse BC compressed textures in iPadOS. You can check an individual GPU's support for this feature by inspecting its `MTLDevice.supportsBCTextureCompression` property at runtime.
9. The `A8Unorm` pixel format is incompatible with imageblocks with explicit layout. Use either an `R8Unorm` texture view, or imageblocks with implicit layout.
10. You can only apply the `RG32Uint` format to a `ulong` texture on a GPU that supports the 64-bit atomics feature.
11. GPU devices in the `Apple7` family support **Sparse** depth and stencil textures only for placement sparse textures. GPU devices in `Apple8` through `Apple9` support both placement and automatic heap backing for sparse depth and stencil textures.