Protocol

MTLResidencySet

A collection of resource allocations that can move in and out of resident memory.

iOS 18.0+ | iPadOS 18.0+ | Mac Catalyst 18.0+ | macOS 15.0+ | tvOS 18.0+ | visionOS 2.0+

```
protocol MTLResidencySet : NSObjectProtocol
```

Mentioned in

Simplifying GPU resource management with residency sets

Overview

Residency sets are a way you can tell Metal which resource allocations, such as buffers, textures, and heaps, to make *resident*, or GPU-accessible. Adding allocations to a residency set requires less overhead than the equivalent methods of a command encoder. Residency sets also give you more control when Metal makes their allocations resident, and for how long they remain resident. However, residency sets don't track hazards, so you need to account for hazards with fences and events.

You can change which `MTLAllocation` instances are in a residency set at any time by:

1. Staging additions and removals with the `addAllocation(_:)` and `removeAllocation(_:)` methods, respectively, or with their sibling methods
2. Applying staged changes by calling the residency set's `commit()` method

Metal doesn't synchronize the state of the residency set between the CPU and the GPU. This means you can add resource allocations to the set while the GPU is actively running a command buffer that's accessing them.

> **Important**
>
> If there's a resource in a residency set that the GPU no longer needs access to, you can remove that resource from the residency set, even while the GPU is actively accessing other resources from the same residency set.

Metal makes the union of all residency sets' allocations resident. This means each resource allocation, such as a buffer, can have an entry in multiple residency sets at the same time. Removing an allocation from one residency set doesn't affect its residency if it also has an entry in another residency set. So you can remove an entire residency set from a command queue and only remove the allocations from residency that are unique to that set. All other resource allocations remain in residency because at least one other residency set has an entry for each.

Alternatively, render and compute command encoders have the following methods that make resource allocations resident:

`MTLRenderCommandEncoder`	`MTLComputeCommandEncoder`
`useResource(_:usage:stages:)`	`useResource(_:usage:)`
`useResources(_:usage:stages:)`	`useResources(_:usage:)`
`useHeap(_:stages:)`	`useHeap(_:)`
`useHeaps(_:stages:)`	`useHeaps(_:)`

These command encoder methods:

- Support hazard tracking to applicable resources (see Resource fundamentals)
- Require CPU overhead for each resource or heap, which scale up with each one you add
- Apply to a single command encoder, which means you need to call the methods again for the same resources for each command encoder

Residency sets, by contrast:

- Don't support hazard tracking, which means you need to account for hazards with `MTLFence` and `MTLEvent` instances
- Require minimal CPU overhead by aggregating allocations at little to no cost for each resource or heap
- Can attach to a command buffer with a single call, which makes residency set's allocations available to all of that command buffer's encoders

- Can attach to a command queue with a single call

Metal attaches all of a command queue's residency sets to a command buffer from that queue when you call the command buffer's `commit()` method.

> **Important**
>
> Residency sets don't support sparse heaps or sparse textures, and their methods aren't thread-safe.

See Simplifying GPU resource management with residency sets for information about associating a residency set to command buffers and command queues.

Create a residency set

Make a residency set by configuring an `MTLResidencySetDescriptor` instance and passing it to the `makeResidencySet(descriptor:)` method of an `MTLDevice`.

Swift | Objective-C

```
let setDescriptor = MTLResidencySetDescriptor()
setDescriptor.label = "Primary residency set"
setDescriptor.initialCapacity = 42

let residencySet = try device.makeResidencySet(descriptor: setDescriptor)
```

Add allocations to a residency set

Add individual resource allocations to a residency set by calling `addAllocation(_:)`, or add multiple allocations with `addAllocations(_:)`.

Swift | Objective-C

```
let residencySet = try device.makeResidencySet(descriptor: setDescriptor)

residencySet.addAllocation(buffer0)
residencySet.addAllocation(buffer1)
residencySet.addAllocation(texture0)
residencySet.addAllocation(texture1)
```

```
residencySet.addAllocation(heap)

let allocations = [buffer2,
                   texture2,
                   argumentBufferHeap,
                   textureHeap]

residencySet.addAllocations(allocations)
```

The residency set can handle redundant entries for the same allocation because it ignores duplicates that already have an entry in the set.

> **Important**
>
> Adding a resource, such as a buffer or texture, that originates from a heap to a residency set makes its entire heap resident.

Remove allocations from a residency set

Remove individual resource allocations from a residency set by calling `removeAllocation(_:)`, or remove multiple allocations with `removeAllocations(_:)`.

Swift | Objective-C

```
residencySet.removeAllocation(buffer1)
residencySet.removeAllocations( [argumentBufferHeap, textureHeap] )
```

Like the methods that add resource allocations to the set, these methods aggregate removals with little CPU overhead. So you can call the methods multiple times without adversely affecting runtime performance.

Commit the changes to a residency set

Apply the updates to a residency set by calling its `commit()` method.

Swift | Objective-C

```
residencySet.commit()
```

A residency set's addition and removal methods don't take effect until you call this method.

Topics

Adding allocations

Add allocation instances, including buffers, textures, and heaps, to a residency set.

`func addAllocation(any MTLAllocation)`

Stages a single resource to join the residency set's list of allocations.

Required

`func addAllocations([any MTLAllocation])`

Stages multiple resources to join the residency set's list of allocations.

Removing allocations

Remove allocation instances, including buffers, textures, and heaps, from a residency set.

`func removeAllAllocations()`

Stages all the resources in the residency set to leave its list of allocations.

Required

`func removeAllocation(any MTLAllocation)`

Stages a single resource to leave the residency set's list of allocations.

Required

`func removeAllocations([any MTLAllocation])`

Stages multiple resources to leave the residency set's list of allocations.

Finalizing pending allocation changes

Complete the additions and removals since the last commit.

`func commit()`

Applies any pending additions to and removals from the residency set.

Required

Requesting residency for the allocations

Ask Metal to make the residency set's allocations resident.

`func requestResidency()`

Tells Metal to do as much preparatory work as it can, with the system's current conditions, to make the set's resource allocations resident.

Required

Releasing the allocations from residency

Notify Metal that you no longer need the residency set's allocations to be resident so it can reallocate the underlying memory.

`func endResidency()`

Informs Metal that the residency set's allocations no longer need to be resident, and that it can reuse the memory for other allocations.

Required

Inspecting a residency set

Identify a residency and check its current allocations and memory footprint.

`var label: String?`

An optional name that can help you identify the residency set.

Required

`var device: any MTLDevice`

The Metal device that owns the residency set.

Required

`func containsAllocation(any MTLAllocation) -> Bool`

Returns a Boolean value that indicates whether the residency set contains a specific resource allocation.

Required

`var allAllocations: [any MTLAllocation]`

The residency set's current list of resource allocations.

Required

`var allocationCount: Int`

The number of resource allocations in the residency set.

Required

`var allocatedSize: UInt64`

The amount of resident memory, in bytes, the residency set's resource allocations consume.

Required

Relationships

Inherits From

`NSObjectProtocol`

See Also

Residency sets

Simplifying GPU resource management with residency sets

Organize your resources into groups and influence when they become accessible to the GPU.

`class MTLResidencySetDescriptor`

A configuration that customizes the behavior for a residency set.