The Swift Runtime

This document describes the ABI interface to the Swift runtime, which provides the following core functionality for Swift programs:

- memory management, including allocation and reference counting;
- the runtime type system, including dynamic casting, generic instantiation, and protocol conformance registration;

It is intended to describe only the runtime interface that compiler-generated code should conform to, not details of how things are implemented.

The final runtime interface is currently a work-in-progress; it is a goal of Swift 3 to stabilize it. This document attempts to describe both the current state of the runtime and the intended endpoint of the stable interface. Changes that are intended to be made before stabilization are marked with **ABI TODO**. Entry points that only exist on Darwin platforms with ObjC interop, and information that only pertains to ObjC interop, are marked **ObjC-only**.

Deprecated entry points

Entry points in this section are intended to be removed or internalized before ABI stabilization.

Exported C++ symbols

ABI TODO: Any exported C++ symbols are implementation details that are not intended to be part of the stable runtime interface.

swift_ClassMirror_count

swift_ClassMirror_quickLookObject

swift_ClassMirror_subscript

swift_EnumMirror_caseName

swift_EnumMirror_count

swift_EnumMirror_subscript

swift_MagicMirrorData_objcValue

swift_MagicMirrorData_objcValueType

swift_MagicMirrorData_summary

swift_MagicMirrorData_value

swift_MagicMirrorData_valueType

swift_ObjCMirror_count

swift_ObjCMirror_subscript

swift_StructMirror_count

swift_StructMirror_subscript

swift_TupleMirror_count

swift_TupleMirror_subscript

swift_reflectAny

ABI TODO: These functions are implementation details of the standard library `reflect` interface. They will be superseded by a low-level runtime reflection API.

swift_stdlib_demangleName

```
@convention(thin) (string: UnsafePointer<UInt8>,
                   length: UInt,
                   @out String) -> ()
```

Given a pointer to a Swift mangled symbol name as a byte string of `length` characters, returns the demangled name as a `Swift.String`.

ABI TODO: Decouple from the standard library `Swift.String` implementation. Rename with a non-`stdlib` naming scheme.

Memory allocation

TODO

```
0000000000001cb30 T _swift_allocBox
0000000000001cb30 T _swift_allocEmptyBox
0000000000001c990 T _swift_allocObject
0000000000001ca60 T _swift_bufferAllocate
0000000000001ca90 T _swift_bufferHeaderSize
0000000000001cd30 T _swift_deallocBox
0000000000001d490 T _swift_deallocClassInstance
0000000000001cd60 T _swift_deallocObject
0000000000001cd60 T _swift_deallocUninitializedObject
0000000000001d4c0 T _swift_deallocPartialClassInstance
0000000000001d400 T _swift_rootObjCDealloc
0000000000001c960 T _swift_slowAlloc
0000000000001c980 T _swift_slowDealloc
0000000000001ce10 T _swift_projectBox
0000000000001ca00 T _swift_initStackObject
```

Reference counting

swift_retainCount

```
@convention(c) (@unowned NativeObject) -> UInt
```

Returns a random number. Only used by allocation profiling tools.

TODO

```
0000000000027ba0 T _swift_bridgeObjectRelease
0000000000027c50 T _swift_bridgeObjectRelease_n
0000000000027b50 T _swift_bridgeObjectRetain
0000000000027be0 T _swift_bridgeObjectRetain_n
000000000001ce70 T _swift_release
000000000001cee0 T _swift_release_n
000000000001ce30 T _swift_retain
000000000001ce50 T _swift_retain_n
000000000001d240 T _swift_tryRetain
0000000000027b10 T _swift_unknownObjectRelease
0000000000027a70 T _swift_unknownObjectRelease_n
0000000000027ad0 T _swift_unknownObjectRetain
0000000000027a10 T _swift_unknownObjectRetain_n
0000000000027d50 T _swift_unknownObjectUnownedAssign
00000000000280a0 T _swift_unknownObjectUnownedCopyAssign
0000000000027fd0 T _swift_unknownObjectUnownedCopyInit
0000000000027ed0 T _swift_unknownObjectUnownedDestroy
0000000000027cb0 T _swift_unknownObjectUnownedInit
0000000000027f20 T _swift_unknownObjectUnownedLoadStrong
00000000000281f0 T _swift_unknownObjectUnownedTakeAssign
0000000000028070 T _swift_unknownObjectUnownedTakeInit
0000000000027f70 T _swift_unknownObjectUnownedTakeStrong
00000000000282b0 T _swift_unknownObjectWeakAssign
0000000000028560 T _swift_unknownObjectWeakCopyAssign
00000000000284e0 T _swift_unknownObjectWeakCopyInit
00000000000283e0 T _swift_unknownObjectWeakDestroy
0000000000028270 T _swift_unknownObjectWeakInit
0000000000028420 T _swift_unknownObjectWeakLoadStrong
0000000000028610 T _swift_unknownObjectWeakTakeAssign
0000000000028520 T _swift_unknownObjectWeakTakeInit
0000000000028470 T _swift_unknownObjectWeakTakeStrong
000000000001d3c0 T _swift_unownedCheck
000000000001cfb0 T _swift_unownedRelease
000000000001d0a0 T _swift_unownedRelease_n
000000000001cf70 T _swift_unownedRetain
000000000001cf60 T _swift_unownedRetainCount
000000000001d2b0 T _swift_unownedRetainStrong
000000000001d310 T _swift_unownedRetainStrongAndRelease
000000000001d060 T _swift_unownedRetain_n
000000000001ca20 T _swift_verifyEndOfLifetime
000000000001d680 T _swift_weakAssign
000000000001d830 T _swift_weakCopyAssign
000000000001d790 T _swift_weakCopyInit
000000000001d770 T _swift_weakDestroy
000000000001d640 T _swift_weakInit
000000000001d6d0 T _swift_weakLoadStrong
000000000001d8b0 T _swift_weakTakeAssign
000000000001d800 T _swift_weakTakeInit
000000000001d710 T _swift_weakTakeStrong
000000000002afe0 T _swift_isUniquelyReferencedNonObjC
000000000002af50 T _swift_isUniquelyReferencedNonObjC_nonNull
```

```
000000000002b060 T _swift_isUniquelyReferencedNonObjC_nonNull_bridgeObject
000000000002af00 T _swift_isUniquelyReferenced_native
000000000002aea0 T _swift_isUniquelyReferenced_nonNull_native
000000000????? T _swift_setDeallocating
000000000001d280 T _swift_isDeallocating
```

ABI TODO: `_unsynchronized` r/r entry points

Error objects

The `ErrorType` existential type uses a special single-word, reference- counted representation.

ObjC-only: The representation is internal to the runtime in order to provide efficient bridging with the platform `NSError` and `CFError` implementations. On non-ObjC platforms this bridging is unnecessary, and the error object interface could be made more fragile.

To preserve the encapsulation of the ErrorType representation, and allow for future representation optimizations, the runtime provides special entry points for allocating, projecting, and reference counting error values.

```
00000000000268e0 T _swift_allocError
0000000000026d50 T _swift_bridgeErrorTypeToNSError
0000000000026900 T _swift_deallocError
0000000000027120 T _swift_errorRelease
0000000000027100 T _swift_errorRetain
0000000000026b80 T _swift_getErrorValue
```

ABI TODO: `_unsynchronized` r/r entry points

ABI TODO: `_n` r/r entry points

Initialization

swift_once

```
@convention(thin) (Builtin.RawPointer, @convention(thin) () -> ()) -> ()
```

Used to lazily initialize global variables. The first parameter must point to a word-sized memory location that was initialized to zero at process start. It is undefined behavior to reference memory that has been initialized to something other than zero or written to by anything other than `swift_once` in the current process's lifetime. The function referenced by the second parameter will have been run exactly once in the time between process start and the function returns.

Dynamic casting

```
0000000000001470 T _swift_dynamicCast
0000000000000a60 T _swift_dynamicCastClass
0000000000000ae0 T _swift_dynamicCastClassUnconditional
0000000000028750 T _swift_dynamicCastForeignClass
000000000002ae20 T _swift_dynamicCastForeignClassMetatype
000000000002ae30 T _swift_dynamicCastForeignClassMetatypeUnconditional
0000000000028760 T _swift_dynamicCastForeignClassUnconditional
```

```
00000000000011c0 T _swift_dynamicCastMetatype
0000000000000cf0 T _swift_dynamicCastMetatypeToObjectConditional
0000000000000d20 T _swift_dynamicCastMetatypeToObjectUnconditional
00000000000012e0 T _swift_dynamicCastMetatypeUnconditional
00000000000286c0 T _swift_dynamicCastObjCClass
0000000000028bd0 T _swift_dynamicCastObjCClassMetatype
0000000000028c00 T _swift_dynamicCastObjCClassMetatypeUnconditional
0000000000028700 T _swift_dynamicCastObjCClassUnconditional
0000000000028af0 T _swift_dynamicCastObjCProtocolConditional
0000000000028a50 T _swift_dynamicCastObjCProtocolUnconditional
0000000000028960 T _swift_dynamicCastTypeToObjCProtocolConditional
00000000000287d0 T _swift_dynamicCastTypeToObjCProtocolUnconditional
0000000000000de0 T _swift_dynamicCastUnknownClass
0000000000000fd0 T _swift_dynamicCastUnknownClassUnconditional
```

Debugging

```
0000000000027140 T _swift_willThrow
```

Objective-C Bridging

ObjC-only.

ABI TODO: Decouple from the runtime as much as possible. Much of this should be implementable in the standard library now.

```
0000000000003c80 T _swift_bridgeNonVerbatimFromObjectiveCConditional
00000000000037e0 T _swift_bridgeNonVerbatimToObjectiveC
00000000000039c0 T _swift_getBridgedNonVerbatimObjectiveCType
0000000000003d90 T _swift_isBridgedNonVerbatimToObjectiveC
```

Code generation

Certain common code paths are implemented in the runtime as a code size optimization.

```
0000000000023a40 T _swift_assignExistentialWithCopy
000000000001dbf0 T _swift_copyPOD
000000000001c560 T _swift_getEnumCaseMultiPayload
000000000001c400 T _swift_storeEnumTagMultiPayload
```

Type metadata lookup

These functions look up metadata for types that potentially require runtime instantiation or initialization, including structural types, generics, classes, and metadata for imported C and Objective-C types.

ABI TODO: Instantiation APIs under flux as part of resilience work. For nominal types, `getGenericMetadata` is likely to become an implementation detail used to implement resilient per-type metadata accessor functions.

```
0000000000023230 T _swift_getExistentialMetatypeMetadata
0000000000023630 T _swift_getExistentialTypeMetadata
```

```
0000000000023b90 T _swift_getForeignTypeMetadata
000000000001ef30 T _swift_getFunctionTypeMetadata
000000000001eed0 T _swift_getFunctionTypeMetadata1
000000000001f1f0 T _swift_getFunctionTypeMetadata2
000000000001f250 T _swift_getFunctionTypeMetadata3
000000000001e940 T _swift_getGenericMetadata
0000000000022fd0 T _swift_getMetatypeMetadata
000000000001ec50 T _swift_getObjCClassMetadata
000000000001e6b0 T _swift_getResilientMetadata
0000000000022260 T _swift_getTupleTypeMetadata
00000000000225a0 T _swift_getTupleTypeMetadata2
00000000000225d0 T _swift_getTupleTypeMetadata3
0000000000028bc0 T _swift_getInitializedObjCClass
```

ABI TODO: Fast entry points for `getExistential*TypeMetadata1-3`. Static metadata for `Any` and `AnyObject` is probably worth considering too.

Type metadata initialization

Calls to these entry points are emitted when instantiating type metadata at runtime.

ABI TODO: Initialization APIs under flux as part of resilience work.

```
000000000001e3e0 T _swift_allocateGenericClassMetadata
000000000001e620 T _swift_allocateGenericValueMetadata
0000000000022be0 T _swift_initClassMetadata_UniversalStrategy
000000000001c100 T _swift_initEnumMetadataMultiPayload
000000000001bd60 T _swift_initEnumMetadataSingleCase
000000000001bd60 T _swift_initEnumMetadataSinglePayload
0000000000022a20 T _swift_initStructMetadata
0000000000024230 T _swift_initializeSuperclass
0000000000028b60 T _swift_instantiateObjCClass
```

Metatypes

```
00000000000000b60 T _swift_getDynamicType
0000000000022fb0 T _swift_getObjectType
00000000000006f0 T _swift_getTypeName
00000000000040c0 T _swift_isClassType
0000000000003f50 T _swift_isClassOrObjCExistentialType
0000000000004130 T _swift_isOptionalType
00000000000279f0 T _swift_objc_class_usesNativeSwiftReferenceCounting
000000000002b340 T _swift_objc_class_unknownGetInstanceExtents
000000000002b350 T _swift_class_getInstanceExtents
0000000000004080 T _swift_class_getSuperclass
```

ABI TODO: getTypeByName entry point.

ABI TODO: Should have a `getTypeKind` entry point with well-defined enum constants to supersede `swift_is*Type`.

ABI TODO: Rename class metadata queries with a consistent naming scheme.

Protocol conformance lookup

```
0000000000002ef0 T _swift_registerProtocolConformances
0000000000003060 T _swift_conformsToProtocol
```

Error reporting

```
000000000001c7d0 T _swift_reportError
000000000001c940 T _swift_deletedMethodError
```

Standard metadata

The Swift runtime exports standard metadata objects for `Builtin` types as well as standard value witness tables that can be freely adopted by types with common layout attributes. Note that, unlike public-facing types, the runtime does not guarantee a 1:1 mapping of Builtin types to metadata objects, and will reuse metadata objects to represent builtins with the same layout characteristics.

```
000000000004faa8 S __TMBB
000000000004fab8 S __TMBO
000000000004f9f8 S __TMBb
000000000004f9c8 S __TMBi128_
000000000004f998 S __TMBi16_
000000000004f9d8 S __TMBi256_
000000000004f9a8 S __TMBi32_
000000000004f9b8 S __TMBi64_
000000000004f988 S __TMBi8_
000000000004f9e8 S __TMBo
000000000004fac8 S __TMT_
000000000004f568 S __TWVBO
000000000004f4b0 S __TWVBb
000000000004f0a8 S __TWVBi128_
000000000004eec8 S __TWVBi16_
000000000004f148 S __TWVBi256_
000000000004ef68 S __TWVBi32_
000000000004f008 S __TWVBi64_
000000000004ee28 S __TWVBi8_
000000000004f1e8 S __TWVBo
000000000004f778 S __TWVFT_T_
000000000004f3f8 S __TWVMBo
000000000004f8e8 S __TWVT_
000000000004f830 S __TWVXfT_T_
000000000004f620 S __TWVXoBO
000000000004f2a0 S __TWVXoBo
000000000004f6d8 S __TWVXwGSqBO_
000000000004f358 S __TWVXwGSqBo_
```

Tasks

- Moving to per-type instantiation functions instead of using `getGenericMetadata` directly
- `swift_objc_` naming convention for ObjC
- Alternative ABIs for retain/release
- Unsynchronized retain/release
- Nonnull retain/release
- Decouple dynamic casting, bridging, and reflection from the standard library