

Swift Server APIs: HTTP stream meeting 1

Initial Agenda:

- Current discussion status roundup
- Value Types vs. Reference Types
- Use of NSURLRequest/Response
- Experiences of adding HTTP/2 support to Node.js

Attendees:

Chris Bailey - C vs Swift

Alex Blewitt -

Alfredo Delli Bovi -

Ben Cohen -

Michael Chiu -

Minutes:

C vs Swift Implementations:

Current Implementations

Framework	HTTP 1.x Parser	HTTP/2 Parser	Request Type	Response Type	Notes
Kitura	c-http parser	N/A	Class	Class	
Perfect	c-http parser(1)	Swift parser	Class	Class	(1) HTTP 1.x parser used to be pure Swift
Swifter	Swift parser	N/A	Class	Enum	
Vapor	Swift parser	N/A	Class (2)	Class (2)	(2) Switched to classes from structs
Zewo	c-http parser	N/A	Struct (3)	Struct (3)	(3) Class used for stream backed Body fields
spartanX	Swift parser	N/a	struct	struct	

Notes:

* (1) Perfect originally implemented their own HTTP 1.x parser in Swift, but switched to using the c-http parse from Node.js

* (2) Vapor originally implemented HTTP request and response as structs, but switch to using classes

* (3) Zewo uses structs for HTTP request and response, however the Body field is an enum. Where the enum is a stream, it is backed by a class.

Assume we re-use C:

- Danger of re-writing existing code is the danger of introducing additional bugs, simply due to the lack of available time in the field (c.f. Harmony)
- We have to make sure that the C API does not influence the Swift API.
 - Memory management driving design etc
- Performance: the only scope to really exceed C is likely around inlining and round-tripping types (ie, conversion of types crossing the C <-> Swift API boundary).

- We should be more wary of non-battle tested C APIs - does HTTP/2 fall into this category?
- We could test out by looking at whether we could have more than one back end? Could we test out the API by converting onto the Swift-based Vapor or Swifter library?

Licensing issues - around MIT in Apache 2 code. Should be possible - need to follow up with the Core team

Value types vs Reference Types:

We should probably use the terms “semantics” rather than types, because behaviours and types are separated.

Foundation was originally reference only, and we’ve started to add value semantics

- We probably should reuse the Foundation types, so we should focus in the short term in making sure that they are current right.

>> The only question is potentially around the mixing of reference and value types (ie, a struct wrapping a stream as a reference type).

Built a map/flow diagram of how a request is handled and use that to model what we need to do.

Next steps: