Mojo Message Format

OVERVIEW

The Mojo Message Format is an application of the Mojo Archive Format. A message is the data transmitted via the Read and Write methods of a MessagePipe. As such, the size of a message is known to the caller of the Read and Write methods.

A message begins with the MessageHeader struct. The rest is message-specific payload.

HEADER

The MessageHeader described in Mojom is as follows:

```
struct MessageHeader {
     uint32 interface_id;
     uint32 type;
     uint32 flags;
     [MinVersion=1] uint64 request_id;
     [MinVersion=2] GenericStruct? payload;
     [MinVersion=2] array<uint32>? payload_interface_ids;
};
```

The *interface_id* field is used to identify different interfaces running on the MessagePipe.
The *type* field names the message. A message's name is scoped to the interface for which the message is sent. A given service declares an enumeration of message types that it supports.

The same value for the *type* field is used both for a request message and its response.

The *flags* field is used to convey additional options for the message. The following bit values are defined:

```
enum {
     kMessageExpectsResponse = 1 << 0,
     kMessageIsResponse      = 1 << 1,
     kMessageIsSync          = 1 << 2
};
```

The *kMessageExpectsResponse* field is true when the receiver is expected to generate a corresponding response message.

The *kMessageIsResponse* field is mutually exclusive with *kMessageExpectsResponse* and is true when the receiver sends a response to a request.

The *kMessageIsSync* field indicates whether the message is for a sync method call. If *kMessageIsSync* is set, either *kMessageExpectsResponse* or *kMessageIsResponse* must also be set.

The *request_id* field is optional and only included in the MessageHeader struct if *kMessageExpectsResponse* or *kMessageIsResponse* is set. The *request_id* specified when *kMessageExpectsResponse* is set should be echoed in the corresponding message where *kMessageIsResponse* is set.

Note: *request_id* is 64-bit to provide for the possibility of in-process implementations stashing a pointer at this location in the MessageHeader as an optimization.

If the MessageHeader struct is of version 0 or version 1, then a message-specific payload struct immediately follows the MessageHeader with 8-byte alignment.

If the MessageHeader struct is of version 2 or above, the *payload* field points to the payload struct. The *payload_interface_ids* field points to a mojo array of uint32 containing all interface IDs transferred. In the message payload, an associated interface pointer is encoded as a uint32 index into the payload_interface_ids array + a uint32 version field; an associated interface request is encoded as a uint32 index into the payload_interface_ids array.