Message

Notation : `msg` or `msg_call`

Description : Messages allow communication between accounts (whether contract or external,) and are a carryover from established concepts in Computer Science, most notably the *MPI: Message-Passing Framework*. Messages can come in the form of `msg_calls` which give output data. If an account has EVM code in it (a contract account,) this code gets executed when the account receives a message call. Message calls and contract creations are both *transactions*, but contract creations are never considered the same as message calls. Message calls always transfer some amount of value to an account. If the message call is an account creation transaction then the value given is takes on the role of an endowment toward the new account. Every time an account receives a message call it returns the `body`, something which is triggered by the `init` function. A message call can come through a transaction, or through the internal execution of code. Message call `transactions` only contain data. They are separate from regular, standard *transactions*.

Message calls always have a universally agreed-upon cost in gas. There is a strong distinction between contract creation transactions and message call transactions. Computation performed, whether it is a contract creation or a message call, represents the currently legal valid state. There can be no invalid transactions from this point.[1] There is also a message call/contract creation *stack*. This stack has a depth, depending on how many transactions are in it. Contract creations and message calls have entirely different ways of executing, and are entirely different in their roles in Ethereum. The concepts can be conflated. Message calls can result in computation that occurs in the next state rather than the current one. If an account that is currently executing receives a message call, no code will execute, because the account might exist but has no code in it yet. To execute a message call transactions are required:

- `Sender`
- `Transaction_Originator`
- `Recipient`
- `Account` (usually the same as the recipient)

- `Available_Gas`
- `Value`
- `Gas_Price`
- An arbitrary length byte-array. `arb_array`
- `Present_Depth` of the message call/contract creation stack.

References

[1] D. G. Wood, *Ethereum: A secure decentralised generalised transaction ledger*, `https://github.com/ethereum/yellowpaper`, 2017.