Execution

Notation : `execution`

Description : The execution of a transaction defines the state transition function: `stf`. However, before any transaction can be executed it needs to go through the initial tests of intrinsic validity.

1 Intrinsic Validity

The criteria for intrinsic validity are as follows:

- The transaction follows the rules for *well-formed RLPs* (recursive length prefixes.)
- The *signature* on the transaction is valid.
- The *nonce* on the transaction is valid, i.e. it is equivalent to the sender account's current nonce.
- The `gas_limit` is greater than or equal to the `intrinsic_gas` used by the transaction.
- The sender's account balance contains the cost required in up-front payment.

Accordingly, the post-transactional state of Ethereum is expressed thus:

```
transaction(post.state) = stf(present.state, transaction)
```

While the amount of gas used in the execution is expressed: `stf(gas_used)` and the accrued log items belonging to the transaction are expressed: `stf(logsbloom, content)(logsbloom, set)` Information concering the result of a transaction's execution is stored in the transaction receipt `tx_receipt`. The set of log events which are created through the execution of the transaction, `logs_set` in addition to the bloom filter which contains the actual information from those log events `logs_bloom` are located in the transaction receipt. In addition, the post-transaction state `post_transaction(state)` and the amount of gas used in the block containing the transaction receipt post(gas_used)

are stored in the transaction receipt. Thusly the transaction receipt is a record of any given `execution`.

A valid transaction execution begins with a permanent change to the state: the nonce of the sender account is increased by one and the balance is decreased by the `collateral_gas`[1] which is the amount of gas a transaction is required to pay prior to its execution. The original transactor will differ from the sender if the message call or contract creation comes from a contract account executing code.

After a transaction is executed, there comes a PROVISIONAL STATE:

```
post_execution(provisional.state)
```

Gas used for the execution of individual EVM opcodes prior to their potential addition to the `world_state` creates the provisional state. `productive_gas`, and an associated substate `substate_a`.

Code execution always depletes `gas`. If gas runs out, an out-of-gas error is signaled (`oog`) and the resulting state defines itself as an empty set; it has no effeffect on the world state. This describes the transactional nature of Ethereum. In order to affect the WORLD STATE, a transaction must go through completely or not at all.

1. Designated "`intrinsic_gas`" in the Yellowpaper