Faeth – Fae as an Ethereum subchain

This document details the protocol that allows a limited form of transaction *between* Fae and Ethereum. Such transactions may, in essence, use Ether to pay for a Fae computation, but not to transfer structured transaction results in either direction. The integration also embeds Fae within Ethereum as a sub-blockchain, one whose transactions are lazily evaluated as usual for Fae. The protocol creates a cryptographically trustworthy association between the Ethereum half of the transaction and the Fae half, so that it defines a valid Fae transaction if, and only if, payment conditions specified by the Fae "seller" are met by the Ethereum transaction. The Ethereum transaction is valid regardless, however. Currently it is not possible for the Ethereum transaction to do anything other than transfer value; the protocol precludes a meaningful contract call.

Embedding

The embedding is based on the exploitation of one field of the Fae transaction message and several of the Ethereum message:

- The `salt` field of the Fae transaction, which can contain any data whatsoever and whose interpretation is unspecified, in Faeth is required to have the three fields:
  - `faeSalt`: a further free-format field (having the same purpose as `salt` in general: to distinguish otherwise identical transaction messages)
  - `ethFee`: an amount of Ether in units of *wei* (where *1 ether = 1e18 wei*) that must be transferred in the `value` field of the corresponding Ethereum transaction
  - `ethRecipient`: the Ethereum address that must appear in the Ethereum transaction's `to` field.
- The `value` and `to` fields of the Ethereum transaction, as described above
- The `input` field of the Ethereum transaction, which normally contains a string of bytes to be passed as input to the contract in the `to` field, but which in Faeth contains the binary serialization of the Fae transaction.

We will call an Ethereum transaction having an embedded Fae transaction of this form a "Faeth transaction". Note that since the `input` is entirely occupied by data that is nonsensical to Ethereum, the only changes to Ethereum state that arise from running the transaction are those relating to the transfer of Ether: either from sender to recipient via `value`, or from sender to coinbase via gas costs. If the `to` address actually contains contract code that consumes the input, that code will likely be invalid, so the recipient should simply be a private-key account.

Validity and economics

A Faeth transaction is valid if the embedded Fae transaction is valid (notably, including the correct required signatures) and if its `salt` matches the Ethereum transaction as described above. In combination with the possibility of multiple required Fae signatures, the following economics is therefore enabled:

- A "seller" in Fae creates the complete Fae transaction, including two required signatures: the seller's own, and another from the Ethereum "buyer". Only the former signature is provided at this stage.
- The partially signed Fae transaction is then sent to the "buyer" (via some means; the present implementation uses Faeth itself to do so, simply ignoring the fact that a Faeth transaction whose embedded Fae transaction is partially signed, hence invalid, will not execute, since in fact it should not).
- The "buyer" scrutinizes the Fae transaction, advisedly running it in a sandbox as usual for multi-party Fae transactions. When satisfied with its effects, the buyer signs the Fae transaction and places it in the appropriate Ethereum transaction, which is then included in an Ethereum block (eventually).

The "seller" thus offers a Fae-based service and accepts Ethereum currency as payment. An alternative perspective is that the "seller" is *invested* in the Fae subchain and wishes to *buy out* into Ethereum.

The “buyer”, correspondingly, hopes to obtain a Fae service in exchange for some of their own Ether balance. Alternatively, the buyer wishes to *buy in* to Fae.

No matter the interpretation, each party has one foot in each smart contract system, but there is no direct exchange of information between the two. From the perspective of other Ethereum participants, the Faeth transaction was merely an Ethereum payment; the “seller” and, possibly, the “buyer” then continue to interact normally with Ethereum with the updated balances. Since the EVM does not execute the Fae code (indeed, it does not even execute any EVM code) the Ethereum blockchain as a whole is spared the computational burden of verifying the operations of the Fae subchain; only the two interested parties need do so. The Ethereum blockchain must still include the corresponding transaction *messages*, however.

COMPLEX INTERACTIONS

The scheme described above allows a payment in Ether to enable a Fae computation, but it is possible to require more general contract activity. This relies on one observation and two extensions:

- **Observation:** The `ethRecipient` does not need to be a public-key account and can, in fact, be an account with code, so that the transaction will operate as a message call running that code.
- **Extension:** If the account's code follows the Ethereum contract ABI, then it expects a well-typed input byte array, which can be prepended to the encoded Fae transaction. The code will then read only the intended input and ignore the latter bytes, because they are not referred to by any offset given in the input.
- **Extension:** Fae can retrieve the buried Fae transaction message in a bounded number of steps if the index at which it starts is appended to the input as a single word (say, 64 bits, which allows quite a bit of space for the Ethereum portion).

With these three points satisfied, a Faeth transaction can demand that a particular contract be called with a particular input if the `salt` gets an additional field `ethInput` containing the Ethereum portion of the input, appearing at the beginning of the transaction message structure. The decoding process then has the steps:

- Read the last word-length of bytes of the input as a positive integer *n*.
- Prepend these bytes to the full input, thus length-prefixing the Ethereum portion.
- If this is indeed a Faeth transaction, the result should decode correctly to a Fae transaction message.

This allows Fae to quickly scan Ethereum transactions to find valid Faeth transactions.

The standard proposed here therefore allows a Faeth transaction to exchange an arbitrary contract call in Ethereum for a Fae transaction, enabling a very general form of economic interplay between Fae and Ethereum.

Blockchain

This protocol not only embeds Fae in Ethereum, it does so in a way that synchronizes their transaction chains. Specifically, it uses a subset of the Ethereum chain *as* the Fae chain. Fae has no native concept of “blocks”, only transactions executed in order, so it can operate on the basis of the order implied by Ethereum's chain, oblivious to the source of that ordering. The only connection between the two, the `salt` field of the Fae transaction, has no meaning within Fae, being purely metadata that, in this case, is used to participate in the Faeth protocol.

Fae continues to execute transactions lazily, per its normal functioning, so only interested parties will execute the Faeth transaction. *The same is true of Ethereum.* Although everyone must process the Faeth transaction as an Ethereum transaction, this entails little work because the EVM is not run. The actual work, the Fae transaction, is only executed by Ethereum participants who happen to be “invested” in Fae and who care about the results of the transaction. This brings a measure of scalability to Ethereum; in the extreme, it may function as a “Bitcoin with reasons”, where the only

Ethereum transactions are value transfers, with all the contractual work backing those transfers relegated to the lazy-evaluated Fae subchain. Note that Bitcoin itself does not allow a Faeth analogue because its transaction messages do not contain any uninterpreted fields in which to embed a Fae transaction.