Asterizm protocol on Cardano

Actors

- Client
- Relayer
- minting script (say, with `p0` = its `policyId`)

Elements

- `msg` = useful payload (message)
- `msgHash` = hash of `msg`
- `pkh_R` = pub-key-hash of relayer
- `pkh_C` = pub-key-hash of client

Workflow

1. Client generates `msg` (keeps private)
2. Client sends `msgHash` to relayer
3. Relayer mints token with token name: `hash(pkh_R <> msgHash)`. Minted token is kept in relayer's wallet.
4. Client listens for tokens onchain with policy id = p0.
5. If client detects a token that has policy id = p0 and token name is compatible with `msg`, then client posts a Tx with `msg` as *datum*, burning the relayer's token. (Burning serves the purpose of "certifying" that relayer received `msgHash` from client.) The minting script enforces that the following is required for burning:
    - passing `pkh_R` as redeemer (needed to recompute token name and verify consistency with `msg`),
    - paying a fee to relayer's wallet (payment pub-key-hash = `pkh_R`),
    - minting a new token with token-name: `hash(pkh_C <> msgHash)` (purpose: identify the UTxOs with datums containing `msg`'s),
    - newly minted token is sent to a predefined "always fails" validator address (purpose: prevent UTxOs with datums from being spent/modified).

Notes:

- The minting script (`asterizmMessage`) is parameterized by the “relayer’s fee” and the “always fails” validator.
- Optional: offchain code posts the `msgHash` as metadata in order to improve traceability.
- In a future version we could add a “client nonce” to the bytestring being hashed to produce the token name, in order to distinguish the (rare) cases when two clients send the same message.
- In a future version we could make the fee to be modifiable/dynamic by involving a reference UTxO that keeps a registry of relayers’ fees in its datum.