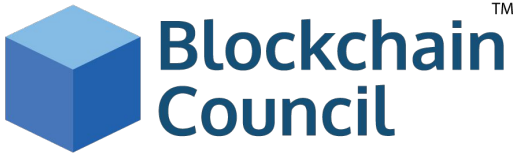

Certified Hyperledger Expert

Hyperledger Fabric Endorsement Flow

Hyperledger Fabric Endorsement Flow

- Clients creates a transaction and send it to endorsement peers of its choice.
- Endorsing Peers simulates the transaction and produces an endorsement signature.
- Client collects the endorsement signatures for a transaction and broadcasts it through an ordering service.
- Ordering service delivers the transactions to peers.

Client Creates Transaction

Client initiates the process by submitting a propose message to the endorsing peers of it's choice.

The set of endorsing peers are made available to the client which are available under endorsement policy.

- The format of a PROPOSE message is *<PROPOSE,tx,[anchor]>*, where tx is a mandatory and defined as *<clientID,chaincodeID,txPayload,timestamp,clientSig>* and anchor optional argument contains read version dependencies, which is used to bind the transaction to specific key value entry within the ledger.
- txPayload is also defined as *<operation, metadata>*, where operation is the specific function from the chaincode and metadata is additional information for the invocation.

Client decides the sequence of transactions with endorsers. For example they can send only the transaction without anchor to one endorser. Later endorser can compute the anchor and client can use that anchor to communicate with other endorsers.

Endorsement Signature

Once the endorsing peers receive the `<PROPOSE,tx,[anchor]>`, it verifies the client signature and simulates the transaction.

Simulating a transaction involves endorsing peer executing a transaction by invoking the chaincode over the copy of state that endorsing peer holds locally. This gives the peer the change of state in readset and writeset.

Important: Endorsing peer do not update its state, it only interprets the transaction records accessed by chaincode for reading and writing.

If a client specifies anchor in the PROPOSE message then client specified anchor must equal readset produced by endorsing peer when simulating the transaction.

Endorsement Signature

After simulating the transaction the endorsement peer forwards the transaction PROPOSAL internally to the endorsing logic.

Endorsing logic accepts the transaction PROPOSAL and signs it. Following messages are sent back through endorsing logic:

- If the transaction is accepted: *<TRANSACTION-ENDORSED, tid, tran-proposal,epSig>*
- If the transaction is rejected: *<TRANSACTION-INVALID, tid, REJECTED>*

Where:

- tran-proposal := *(epID,tid,chaincodeID,txContentBlob,readset,writeset)*, epID is the endorsing peer ID, tid is the transaction ID, chaincodeID is the chaincode called, txContentBlob is transaction specific information.
- epSig is the endorsing peer signature over the tran-proposal.

Broadcasting through Ordering Service

The Client waits to receives enough message signatures of transaction endorsement which is defined under endorsement policies.

If the Client do not receive enough transaction proposals then the client abandons the transaction with the option of retry.

If the endorsement policy is satisfied then the client invokes the ordering service by passing the endorsement blob to the ordering service.

Following event occurs:

`broadcast(blob)`, where blob=endorsement

Delivering Transactions to the peers

Once the ordering service receives the endorsement blob and validates against its state. It initiates a deliver event which is defined as: `deliver(seqno, prevhash, blob)`.

All the peers in the networks receives the deliver event and does the following:

- It checks that the `blob.endorsement` is valid according to the policy of the chaincode.
- It also verifies that the dependencies `(blob.endorsement.tran-proposal.readset)` have not been violated meanwhile.

Peers validates the transactions and change the state if required/validated. Invalid transactions do not change state but are maintained in ledger too. All the peers will have the same state after the deliver event has been executed.

THANK YOU!

Any questions?

You can mail us at

hello@blockchain-council.org