**Protocol Specification Hyperledger fabric**

**Transaction** is a request to the blockchain to execute a function on the ledger. The function is implemented by a chaincode.

Transactor is an entity that issues transactions such as a client application.

Ledger is a sequence of cryptographically linked blocks, containing transactions and current world state.

World State is the collection of variables containing the results of executed transactions.

Chaincode is an application-level code stored on the ledger as a part of a transaction. Chaincode runs transactions that may modify the world state.

Validating Peer is a computer node on the network responsible for running consensus, validating transactions, and maintaining the ledger.

Non-validating Peer is a computer node on the network which functions as a proxy connecting transactors to the neighboring validating peers. A non-validating peer doesn’t execute transactions but does verify them. It also hosts the event stream server and the REST service.

Permissioned Ledger is a blockchain network where each entity or node is required to be a member of the network. Anonymous nodes are not allowed to connect.

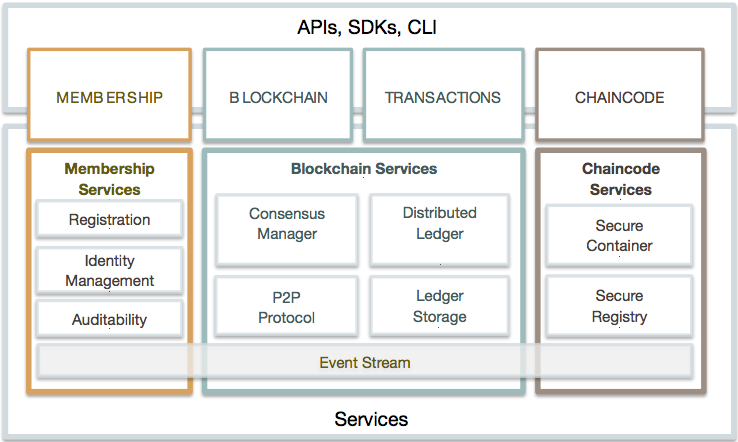
Privacy is required by the chain transactors to conceal their identities on the network. While members of the network may examine the transactions, the transactions can’t be linked to the transactor without special privilege.

Confidentiality is the ability to render the transaction content inaccessible to anyone other than the stakeholders of the transaction.

Auditability of the blockchain is required, as business usage of blockchain needs to comply with regulations to make it easy for regulators to investigate transaction records.

**Fabric architecture:**

The architecture is aligned in 3 categories logically: Membership, Blockchain, and Chaincode services.



**Membership Services:**

It provides services for

1. managing identity
2. privacy
3. confidentiality
4. auditability on the network

Membership services combine elements of Public Key Infrastructure (PKI) and decentralization/consensus to transform a non-permissioned blockchain into a permissioned blockchain.

In permissioned blockchain entities register in order to acquire long-term identity credentials (enrollment certificates,**eCert**).

In the case of users, such credentials enable the Transaction Certificate Authority (**TCA**) to issue pseudonymous credentials.

Such credentials, i.e., transaction certificates(**tCert**), are used to authorize submitted transactions.

Transaction certificates persist on the blockchain, and enable authorized auditors to cluster otherwise unlinkable transactions.

**Blockchain Services:**

Blockchain services manage the distributed ledger through a peer-to-peer protocol.

Different consensus (PBFT, Raft, PoW, PoS) may be plugged in and configured per deployment.

**Chaincode Services:**

Chaincode services provides a secured way to sandbox the chaincode execution on the validating nodes.

The environment is a “locked down” and secured container along with a set of signed base images containing secure OS and chaincode language, runtime and SDK layers for Go, Java, and Node.js

**Events:**

Validating peers and chaincodes can emit events on the network that applications may listen for and take actions on.

chaincodes can generate custom events.

Events are consumed by 1 or more event adapters.

Adapters may further deliver events .

**Application Programming Interface (API):**

Interface to the fabric is a REST API and its variations over Swagger 2.0. The API allows applications to register users, query the blockchain, and to issue transactions.

**Command Line Interface (CLI):**

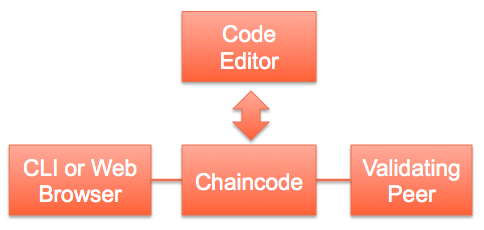
CLI includes a subset of the REST API to enable developers to quickly test chaincodes or query for status of transactions.

**Topology:**

Deployment of the fabric can consist of a

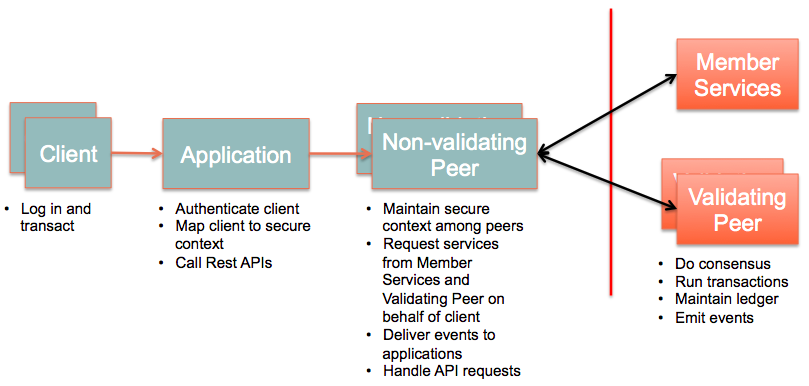
* membership service
* many validating peers
* non-validating peers
* one or more applications

**Single Validating Peer:**



* A non-validating peer is a subset of a validating peer .
* Appropriate for a development en.vironment(may be started up during the edit-compile-debug cycle).
* Doesn’t require consensus.
* By default uses the noops plugin, which executes transactions as they arrive.

**Multiple Validating Peers:**

****

* Production or test networks should be made up of multiple validating and non-validating peers .
* Non-validating peers can take workload off the validating peers, such as handling API requests and processing events.
* validating peer connects to every other validating peer) to disseminate information.
* non-validating peer connects to a neighboring validating peer.
* Non-validating peers are optional .
* Applications may communicate directly with validating peers.

**Multichain:**

Network of validating and non-validating peers may be created to address different needs.

## **Protocol**

Fabric’s peer-to-peer communication is built on gRPC, which allows bi-directional stream-based messaging.

Uses protocol buffers to serialize data structures for data transfer between peers.

These are language-neutral, platform-neutral and extensible mechanism for serializing structured data.

**Message:**

* Messages passed between nodes are encapsulated by **Message** proto structure
* Consists of 4 types: **Discovery, Transaction, Synchronization, and Consensus**
* Each type may define more subtypes embedded in the **payload**
* **payload** is an opaque byte array (byte array of unspecified format)containing other objects such as **Transaction** or **Response.**

**Discovery Messages:**

Upon start up, a peer runs discovery protocol if **CORE\_PEER\_DISCOVERY\_ROOTNODE** (starting point for discovering all the peers on the network)is specified.

* PeerID :name of the peer at start up or defined in the config file
* PeerEndpoint : endpoint and whether it’s a validating or a non-validating peer
* pkiID is the cryptographic ID of the peer
* address :host or IP address and port of the peer in the format ip:port
* blockNumber :height of the blockchain (of current peer)

**Transaction Messages:**

3 types of transactions:

**Deploy**(installs the specified chaincode )

**Invoke and Query**(call a function of a deployed chaincode.)

##### Transaction Data Structure(refer [fabric documentation](https://hyperledger-fabric.readthedocs.io/en/v0.6/protocol-spec.html" \l "preface) for detail format)

Blockchain(Block):

* version - used to track any protocol changes.
* timestamp - be filled in by the block proposer.
* transactionsHash - merkle root hash of the block’s transactions.
* stateHash - merkle root hash of the world state.
* previousBlockHash - hash of the previous block.
* consensusMetadata - Optional metadata that the consensus may include in a block.
* nonHashData - A NonHashData message that is set to nil before computing the hash of the block, but stored as part of the block in the database.
* BlockTransactions.transactions - An array of Transaction messages. **Transactions are not included in the block directly due to their size.**

**World state** of a peer refers to the collection of the states of all the deployed chaincodes

##### CHAINCODE DEPLOY:

**Deploy**

* Shim layer sends a one time REGISTER message to the validating peer with the payload containing the ChaincodeID.
* Validating peer responds with REGISTERED or ERROR on success or failure respectively.
* Shim closes the connection and exits if it receives an ERROR.

**After registration**

* validating peer sends INIT with the payload containing a ChaincodeInput object.
* The shim calls the Init function with the parameters from the ChaincodeInput, enabling the chaincode to perform any initialization.

##### **CHAINCODE INVOKE:**

* Validating peer sends a TRANSACTION message to the chaincode container shim, which in turn calls the chaincode Invoke function.
* Shim responds to the validating peer with RESPONSE or ERROR message
* If ERROR is received, the payload contains the error message generated by the chaincode.

##### **CHAINCODE QUERY:**

* Validating peer sends a QUERY message to the chaincode container shim, which in turn calls the chaincode Query function.
* Query function may return a state value or an error, which the shim forwards to the validating peer using RESPONSE or ERROR messages respectively.

##### **Chaincode State:**

**PUT\_STATE**(persist a key-value pair, with the payload)

**GET\_STATE**(retrieve the value whose key is specified in the payload.)

**DEL\_STATE**(delete the value whose key is specified in the payload)

**RANGE\_QUERY\_STATE**(get a range of values)

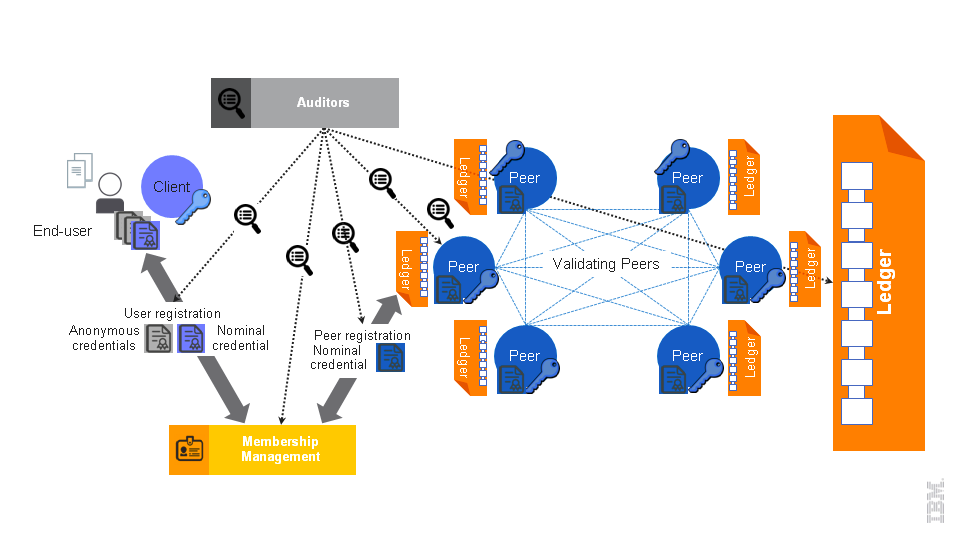
**INVOKE\_CHAINCODE**()

**QUERY\_CHAINCODE**

## **Security**

membership management infrastructure

* identifying an individual user
* open an account for that user to be able to register
* issue the necessary credentials to successfully create transactions
* deploy or invoke chaincode successfully through the fabric

Peers: validating peers and non-validating peers

**Validating peers**

* order and process (check validity, execute, and add to the blockchain) user-messages (transactions) submitted to the network.
* execute transactions

**Non validating peers (also known as peers)**

* receive user transactions on behalf of users, and after some fundamental validity checks, they forward the transactions to their neighboring validating peers.
* maintain an up-to-date copy of the blockchain**.**

**End users:**

registered membership

**Online wallets:**

maintain that user’s credentials

submit transactions solely upon user request

***Business security requirements***

Required for Incorporation of identity and role management.

**Accountability**:means that users of the system, individuals, or corporations, who misbehave can be traced back and be set accountable for their actions.

Transactional privacy:

Transaction anonymity:hidden among anonymity set.

Transaction unlinkability:two or more transactions of the same user should not be linked

Reconciling transactional privacy with identity management:

1. add certificates to transactions to implement a “permissioned” blockchain
2. utilize a two-level system:
3. (relatively) static enrollment certificates (ECerts), acquired via registration with an enrollment certificate authority (CA).
4. transaction certificates (TCerts) that faithfully but pseudonymously represent enrolled users, acquired via a transaction CA.
5. offer mechanisms to conceal the content of transactions to unauthorized members of the system.

Audit support:

check a certain transaction

activity of a particular user of the system

operation of the system itself.

**User Privacy through Membership Services:**

* manage the identity and privacy of users on the network
* validate user’s identity
* register the user in the system
* provide all the credentials

to be an active and compliant participant

to create and/or invoke transactions

**Public Key Infrastructure (PKI)**

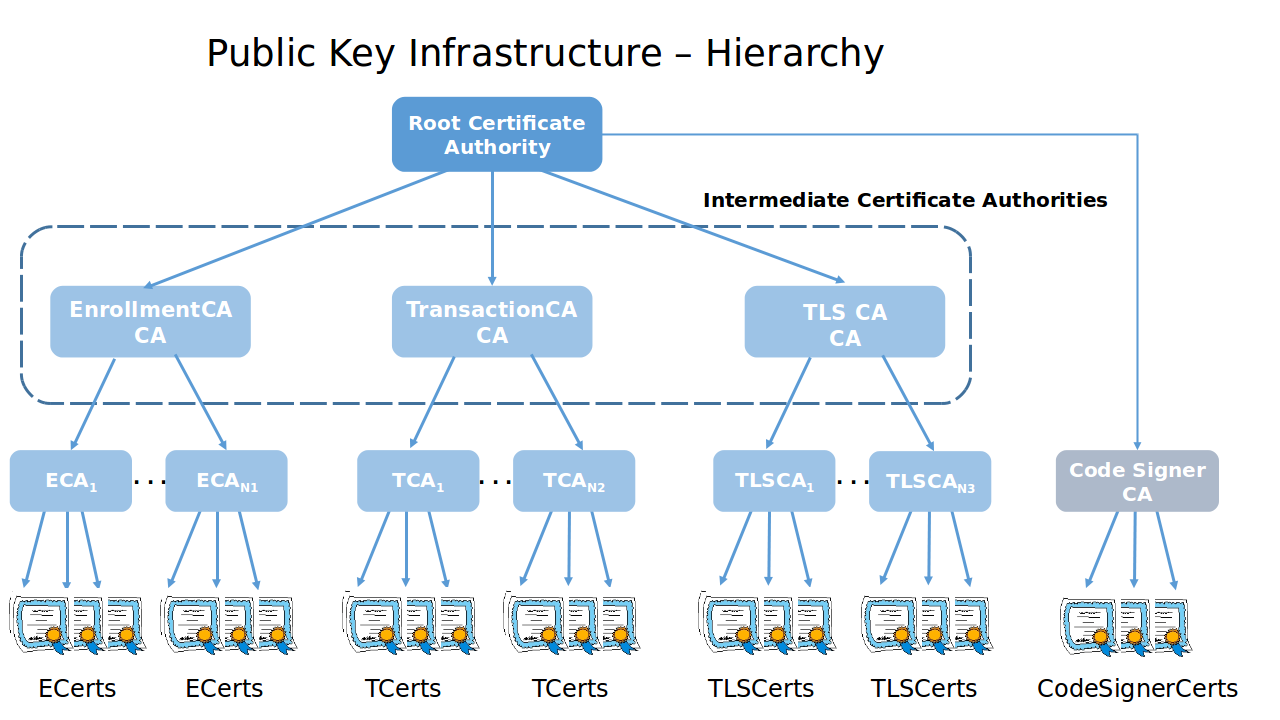
ensures secure exchange of data over public networks

affirms the identity of the other party

manages the generation, distribution and revocation of keys and digital certificates

It has typically:

* Registration Authority (RA)
  + authenticates users and vets the legitimacy of data)
* Certificate Authority (CA)
  + upon advice from an RA, issues digital certificates for specific uses
  + certified directly or hierarchically by a root CA
* certificate database
* certificate storage.



Root Certificate Authority (Root CA):

represents the trust anchor for the PKI scheme.

top-most CA in the PKI hierarchy

Enrollment Certificate Authority (ECA):

validates the registration credentials provided by the user.

issues Enrollment Certificates (ECerts)

Transaction Certificate Authority (TCA):

validates the enrollment credentials provided by the user.

issues Transaction Certificates (Tcerts)

TLS Certificate Authority (TLS-CA):

issues TLS certificates and credentials

Enrollment Certificates (Ecerts):

long-term certificates

issued for all roles, i.e. users, non-validating peers, and validating peers

ECerts Types:

Model A: accessible to everyone.

Model B:preferably accessible to only TCA and auditors, as relying parties.

Transaction Certificates (Tcerts):

short-term certificates for each transaction.

issued by the TCA upon authenticated user-request.

may be configured to not reveal the identities

issued only to users.

May be configured so that this association is known only by the TCA (and to authorized)

auditors)

enable the user to anonymously participate in the system

prevent linkability of transactions.

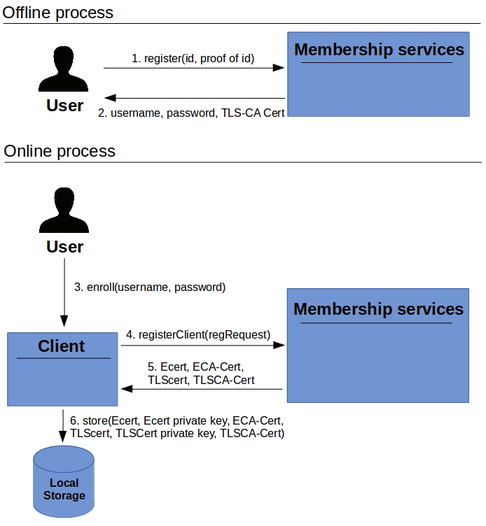
*(Auditability and accountability requirements assume that the TCA is able to retrieve TCerts of a given identity, or retrieve the owner of a specific Tcert.)*

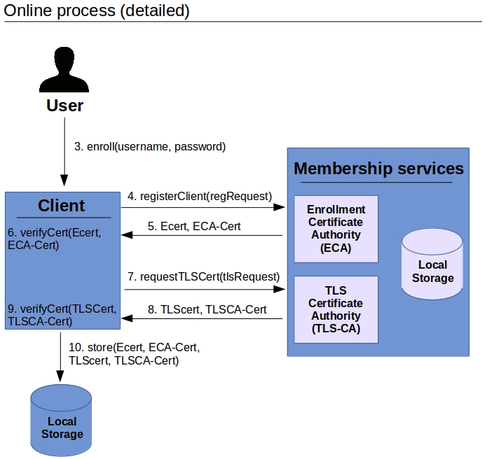
Transaction Certificate structure([refer link here](https://hyperledger-fabric.readthedocs.io/en/v0.6/protocol-spec.html" \l "protocol))

TLS-Certificates (TLS-Certs):

carry the identity of their owner and are used for network level security.

**User/Client Enrollment Process**





Useful Links

[**Certificate Authority API**](https://hyperledger-fabric.readthedocs.io/en/v0.6/API/MemberServicesAPI.html)