1. How many peers in the network need to endorse a transaction?

Answer: The number of peers required to endorse a transaction is driven by the endorsement policy that is specified in the chaincode definition.

2. Does an application client need to connect to all peers?

Answer: Clients only need to connect to as many peers as are required by the endorsement policy for the chaincode.

3. How do I ensure data privacy?

Answer: There are various aspects to data privacy. First, you can segregate your network into channels, where each channel represents a subset of participants that are authorized to see the data for the chaincodes that are deployed to that channel.

Second, you can use private-data to keep ledger data private from other organizations on the channel. A private data collection allows a defined subset of organizations on a channel the ability to endorse, commit, or query private data without having to create a separate channel. Other participants on the channel receive only a hash of the data. For more information refer to the Using Private Data in Fabric tutorial. Note that the key concepts topic also explains when to use private data instead of a channel.

Third, as an alternative to Fabric hashing the data using private data, the client application can hash or encrypt the data before calling chaincode. If you hash the data then you will need to provide a means to share the source data. If you encrypt the data then you will need to provide a means to share the decryption keys.

Fourth, you can restrict data access to certain roles in your organization, by building access control into the chaincode logic.

4. Do the orderers see the transaction data?

Answer: No, the orderers only order transactions, they do not open the transactions. If you do not want the data to go through the orderers at all,

then utilize the private data feature of Fabric. Alternatively, you can hash or encrypt the data in the client application before calling chaincode. If you encrypt the data then you will need to provide a means to share the decryption keys.

5. How do application clients know the outcome of a transaction?

Answer: The transaction simulation results are returned to the client by the endorser in the proposal response. If there are multiple endorsers, the client can check that the responses are all the same, and submit the results and endorsements for ordering and commitment. Ultimately the committing peers will validate or invalidate the transaction, and the client becomes aware of the outcome via an event, that the SDK makes available to the application client.

6. How do I query the ledger data?

Answer: Within chaincode you can query based on keys. Keys can be queried by range, and composite keys can be modeled to enable equivalence queries against multiple parameters. For example a composite key of (owner,asset_id) can be used to query all assets owned by a certain entity. These key-based queries can be used for read-only queries against the ledger, as well as in transactions that update the ledger.
If you model asset data as JSON in chaincode and use CouchDB as the state database, you can also perform complex rich queries against the chaincode data values, using the CouchDB JSON query language within chaincode. The application client can perform read-only queries, but these responses are not typically submitted as part of transactions to the ordering service.

7. How do I query the historical data to understand data provenance?

Answer: The chaincode API `GetHistoryForKey()` will return history of values for a key.

8. How to guarantee the query result is correct, especially when the peer being queried may be recovering and catching up on block processing?

Answer: The client can query multiple peers, compare their block heights, compare their query results, and favor the peers at the higher block heights.

9. Does Hyperledger Fabric support smart contract logic?

Yes. We call this feature Chaincode. It is our interpretation of the smart contract method/algorithm, with additional features.

Answer:

There are generally two ways to develop business contracts: the first way is to code individual contracts into standalone instances of chaincode; the second way, and probably the more efficient way, is to use chaincode to create decentralized applications that manage the life cycle of one or multiple types of business contracts, and let end users instantiate instances of contracts within these applications.

10. Which languages are supported for writing chaincode?

Answer:

Chaincode can be written in any programming language and executed in containers. Currently, Go, Node.js and Java chaincode are supported

11. Does the Hyperledger Fabric have native currency?

No. However, if you really need a native currency for your chain network, you can develop your own native currency with chaincode. One common attribute of native currency is that some amount will get transacted (the chaincode defining that currency will get called) every time a transaction is processed on its chain.