DUMPSBOSS.

Hyperledger Fabric Certified Practitioner (HFCP) Exam

Linux Foundation HFCP

Version Demo

Total Demo Questions: 7

Total Premium Questions: 72

Buy Premium PDF

https://dumpsboss.co

support@dumpsboss.co

support@dumpsboss.co
dumpsboss.co

Topic Break Down

Topic	No. of Questions
Topic 1, Architecture	20
Topic 2, Identity Management	11
Topic 3, Network Configuration	3
Topic 4, Chaincode Development	30
Topic 5, Network Operations	8
Total	72

QUESTION NO: 1

Which of the following statements describes Chaincode correctly?

A. Chaincode is software for creating new blocks on the ledger.

B. Chaincode is software defining assets and transaction instructions for modifying the assets on the ledger.

C. Chaincode is a legally enforceable business contract between organizations.

D. Chaincode is a code of conduct to manage a blockchain consortium.

ANSWER: B

Explanation:

Chaincode is software defining assets and transaction instructions for modifying the assets on the ledger. In Hyperledger Fabric, chaincode implements the business logic, often called a smart contract, that applications invoke to create, read, update, or otherwise manage ledger data. A chaincode transaction function can validate inputs, apply the organization's business rules, and produce the proposed updates to the ledger's world state. Endorsing peers execute the relevant chaincode function to simulate a transaction proposal and return an endorsed response; once the transaction is ordered and validated, its valid state changes are committed to the peer ledger. This model lets Fabric networks express consistent domain-specific rules for assets, ownership, transfers, and other business processes in program code, rather than relying on an external party to apply those rules. Fabric documentation distinguishes chaincode—the deployed program—from the smart-contract functions within it, while using both concepts to describe how transaction logic governs ledger updates. See the Hyperledger Fabric smart contracts and chaincode documentation and the chaincode lifecycle documentation.

QUESTION NO: 2

What is the purpose of the ordering service in Hyperledger Fabric model?

A. To endorse transactions and update the world state.

B. To manage the identities of the participants in the network

C. To assemble transactions into blocks for the blockchain ledger.

D. To validate transactions and maintain the blockchain ledger

ANSWER: C

Explanation:

The purpose of the ordering service is **to assemble transactions into blocks for the blockchain ledger**. In Hyperledger Fabric's execute-order-validate architecture, client applications first collect the required endorsement responses and submit the resulting transaction to an ordering service. The ordering service establishes a deterministic order for submitted transactions across the channel, batches the ordered transactions into blocks, and distributes those blocks to the channel's peers. This common sequence is essential: every peer receives blocks in the same order, allowing peers to maintain a consistent blockchain history when they subsequently validate and commit those blocks. The ordering service is therefore the component responsible for transaction ordering and block creation, rather than application transaction execution. Fabric documentation describes the ordering service as the mechanism that packages transactions into blocks and delivers them to peers, while peer nodes perform validation and ledger commitment. For additional detail, see the Hyperledger Fabric ordering service documentation and the transaction flow documentation.

QUESTION NO: 3

Which of the following information do signature policies provide when creating a network in Hyperledger Fabric?

A. Identity specific users who must sign in order to satisfy a policy.

B. Private keys of all network participants.

C. Type of consensus algorithm used

ANSWER: A

Explanation:

Identity specific users who must sign in order to satisfy a policy is correct because Hyperledger Fabric signature policies express the authorization requirements for an action in terms of signing identities, also called principals. A policy can require signatures from particular members, organizations, or roles recognized by a Membership Service Provider, and it can combine those requirements with logical rules such as requiring signatures from multiple specified principals. Fabric evaluates the signatures attached to a proposal response, transaction, or configuration update against the applicable policy to determine whether the required authorization threshold has been met. For chaincode transactions, endorsement policies define which organizations or identities must endorse a transaction before it can be committed. Network configuration policies similarly govern administrative actions such as configuration updates. Therefore, the useful information supplied by a signature policy is the set of authorized identities and the signature conditions needed to satisfy that policy. The Fabric documentation describes policies as defining the set of organizations permitted to perform an action and explains how signature policies are represented using principals and policy rules. See the Hyperledger Fabric policies documentation and the endorsement policies documentation.

QUESTION NO: 4

A client submits a state-changing transaction through Fabric Gateway. The client receives a transport timeout after submission and cannot determine whether the Gateway peer received the request. Why is immediately submitting the transaction again without further checks unsuitable?

A. The original transaction may already have been submitted, so the client should use its transaction ID to obtain commit status before deciding how to recover.

B. The ordering service automatically rejects every transaction submitted after a client timeout.

C. A timeout proves that endorsement policy validation has failed at every committing peer.

D. Gateway permanently stores the client connection and will retry the transaction after the client reconnects.

ANSWER: A

Explanation:

The original transaction may already have been submitted, so the client should use its transaction ID to obtain commit status before deciding how to recover is correct. A timeout or lost connection does not prove that Gateway failed to submit the transaction. The transaction may already be ordered and committed, or may still be in progress. Commit-status and event mechanisms allow the application to determine the final validation result for the known transaction ID.

Blind resubmission can produce a duplicate business operation or a conflicting transaction. Restarting the orderer or treating the timeout as a validation failure does not establish whether the original transaction reached the network.

QUESTION NO: 5

What is a range query with a start and end key?

A. A query that searches for all keys that are in the range defined by the start and end keys.

B. A query that searches for a specific value in a composite key and all values after it.

C. A query that searches for ledger entries that have values matching the range.

D. A query that searches for all the key field in the value.

ANSWER: A

Explanation:

A query that searches for all keys that are in the range defined by the start and end keys is correct because Hyperledger Fabric chaincode provides range-query APIs specifically to retrieve world-state records by their ledger keys. For example, the ChaincodeStubInterface `GetStateByRange(startKey, endKey)` operation returns a state-query iterator over key-value pairs whose keys sort within the requested range. The start key is inclusive, while the end key is exclusive; an empty start or end key can be used to express an unbounded side of the range. This query operates on keys, rather than comparing the contents of stored values. Applications commonly use it when asset identifiers are designed with sortable prefixes or other predictable key formats, allowing chaincode to iterate through a selected subset of records efficiently. The returned iterator supplies each matching key together with its current world-state value, enabling the chaincode to process the matching records. Fabric also tracks range-query information during transaction simulation so that relevant changes can be detected during validation, which is important for maintaining transaction consistency. See the Fabric chaincode ledger interaction documentation and the Fabric Contract API reference.

QUESTION NO: 6

What allows users to update channels, or invoke new Smart Contracts?

A. CouchDB

B. Private data collections

C. The peer CLI

D. Endorsement policies

ANSWER: C

Explanation:

The peer CLI is the Hyperledger Fabric command-line utility used to submit operational commands to peers. It provides commands for channel administration, including submitting a channel configuration update through `peer channel update`. This command sends a properly signed configuration-update transaction to the ordering service so that the channel's configuration can be changed according to the channel's governance and policy requirements.

The same CLI also supports invoking deployed chaincode, which is Fabric's implementation of smart contracts. The `peer chaincode invoke` command creates and submits an invocation proposal and transaction for a chaincode function on a specified channel. In current Fabric application designs, client applications commonly use Fabric Gateway APIs for transaction submission; however, the peer CLI remains an important administrative, testing, and troubleshooting tool and directly fits the question's channel-update and smart-contract-invocation functions. The Fabric documentation describes channel update operations in the peer command reference and documents chaincode invocation through the peer CLI.

References: Hyperledger Fabric peer channel command reference; Hyperledger Fabric peer chaincode command reference.

QUESTION NO: 7

Two organizations approve a chaincode definition for the same channel. Organization A installed package ID `inventory_1:hashA`, while Organization B installed package ID `inventory_1:hashB`. Both approvals specify the same chaincode name, version, sequence, endorsement policy, and collections configuration. Can the definition become ready to commit if the channel lifecycle endorsement policy is otherwise satisfied?

A. No, because every approving organization must use the same package ID.

B. Yes, because package IDs are organization-specific installation details.

C. No, because the ordering service validates package IDs before ordering a definition.

D. Yes, but only if neither organization has installed its package on a peer.

ANSWER: B

Explanation:

Yes, because organizations can approve the same chaincode definition while using different package IDs. A package ID identifies the chaincode package installed by a particular organization. The lifecycle approval records the package ID locally for an organization that wants its peers to run the chaincode, but the package ID is not a channel-wide field that must be identical across all organization approvals.

The definition parameters that govern the channel-level chaincode definition, such as name, version, sequence, endorsement policy, and collections configuration, must agree for the approvals to satisfy lifecycle requirements. Different package IDs can be appropriate when organizations build or package chaincode independently, provided that their installed packages implement compatible chaincode behavior.