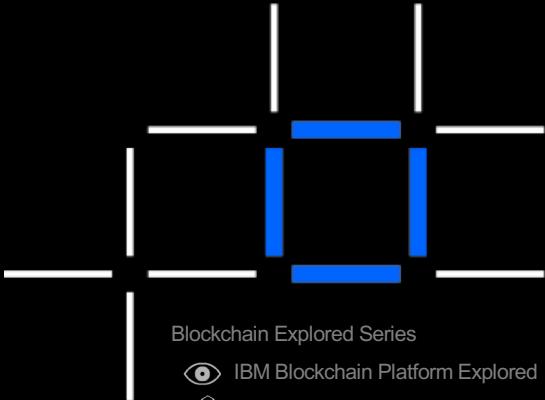


Fabric Explored

A Technical Deep-Dive on Hyperledger Fabric



Blockchain Explored Series

IBM Blockchain Platform Explored

Modeling Applications

Architectures Explored

Fabric Explored

What's New in Tech

IBM

V4.07, 1 March 2019

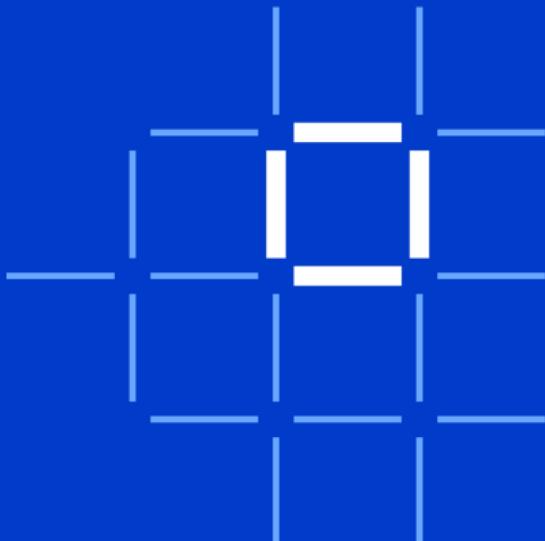
IBM Blockchain



Project Status and
Roadmap



Technical Deep Dive

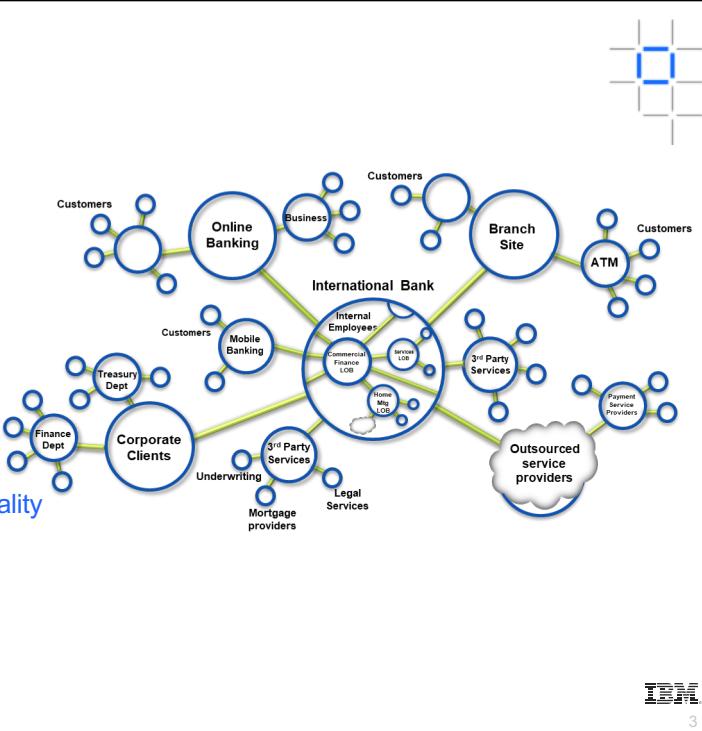


IBM Blockchain

IBM

Blockchain Recap

- Blockchain builds on basic business concepts
 - Business Networks connect businesses
 - Participants with Identity
 - Assets flow over business networks
 - Transactions describe asset exchange
 - Contracts underpin transactions
 - The ledger is a log of transactions
- Blockchain is a shared, replicated ledger
 - Consensus, provenance, immutability, finality



IBM Blockchain

IBM

3

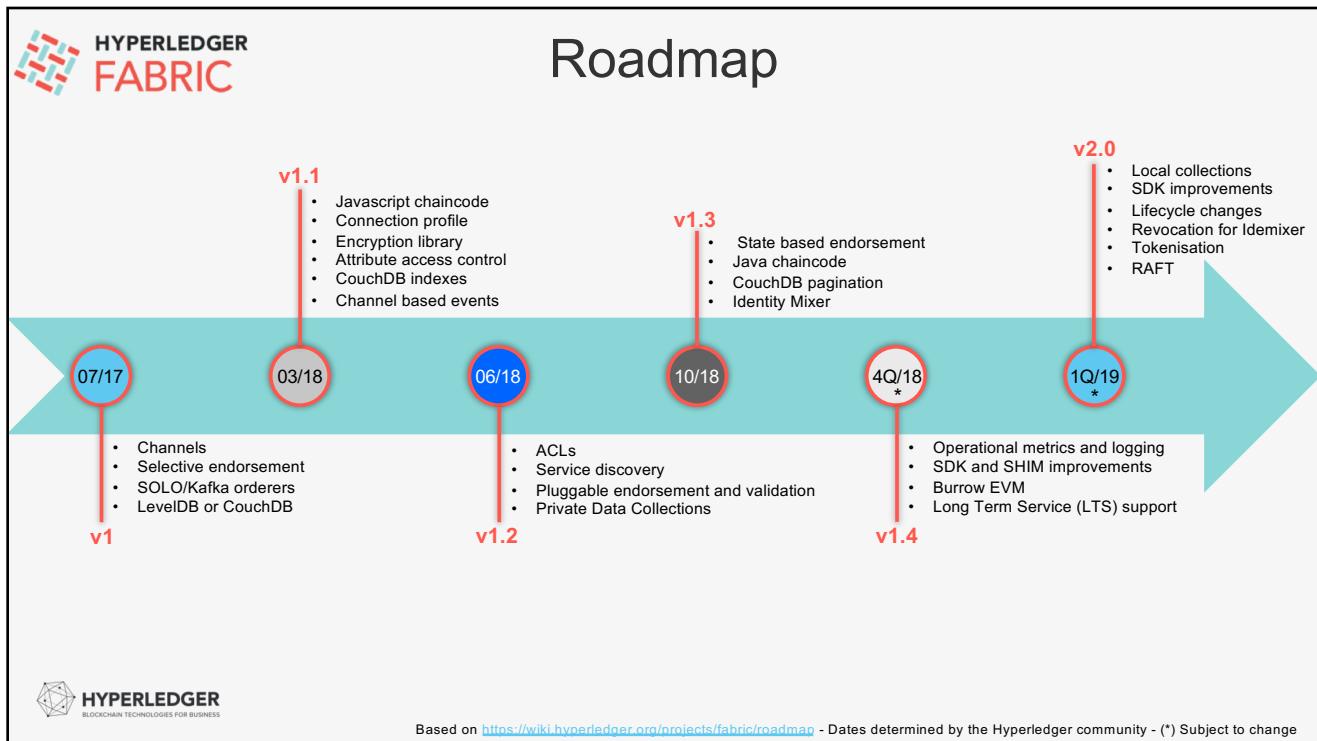
What is Hyperledger Fabric

- Linux Foundation Hyperledger
 - A collaborative effort created to advance cross-industry blockchain technologies for business
- Hyperledger Fabric
 - An implementation of blockchain technology that is intended as a foundation for developing blockchain applications
 - Key technical features:
 - A shared ledger and smart contracts implemented as “chaincode”
 - Privacy and permissioning through membership services
 - Modular architecture and flexible hosting options
- First major version released July 2017: contributions by 159 engineers from 27 organizations
 - IBM is one of the contributors to Hyperledger Fabric

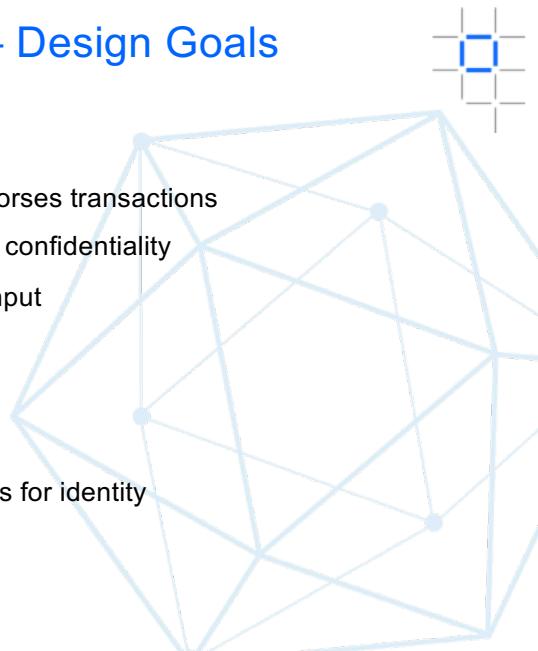
IBM Blockchain

IBM

4



Overview of Hyperledger Fabric v1.x – Design Goals



- Better reflect business processes by specifying who endorses transactions
- Support broader regulatory requirements for privacy and confidentiality
- Scale the number of participants and transaction throughput
- Eliminate non deterministic transactions
- Support rich data queries of the ledger
- Dynamically upgrade the network and chaincode
- Support for multiple credential and cryptographic services for identity
- Support for "bring your own identity"

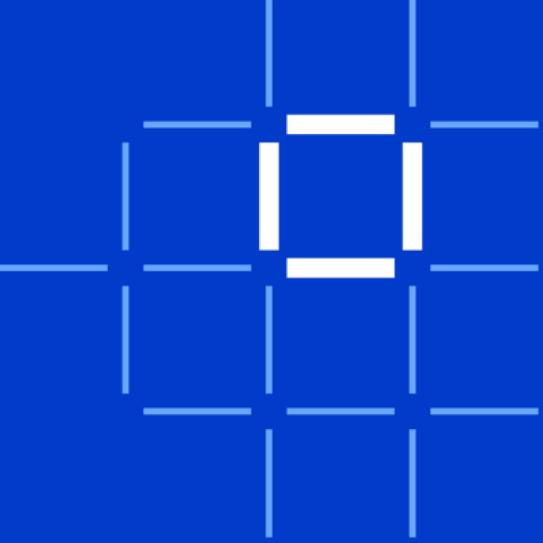
IBM Blockchain

IBM

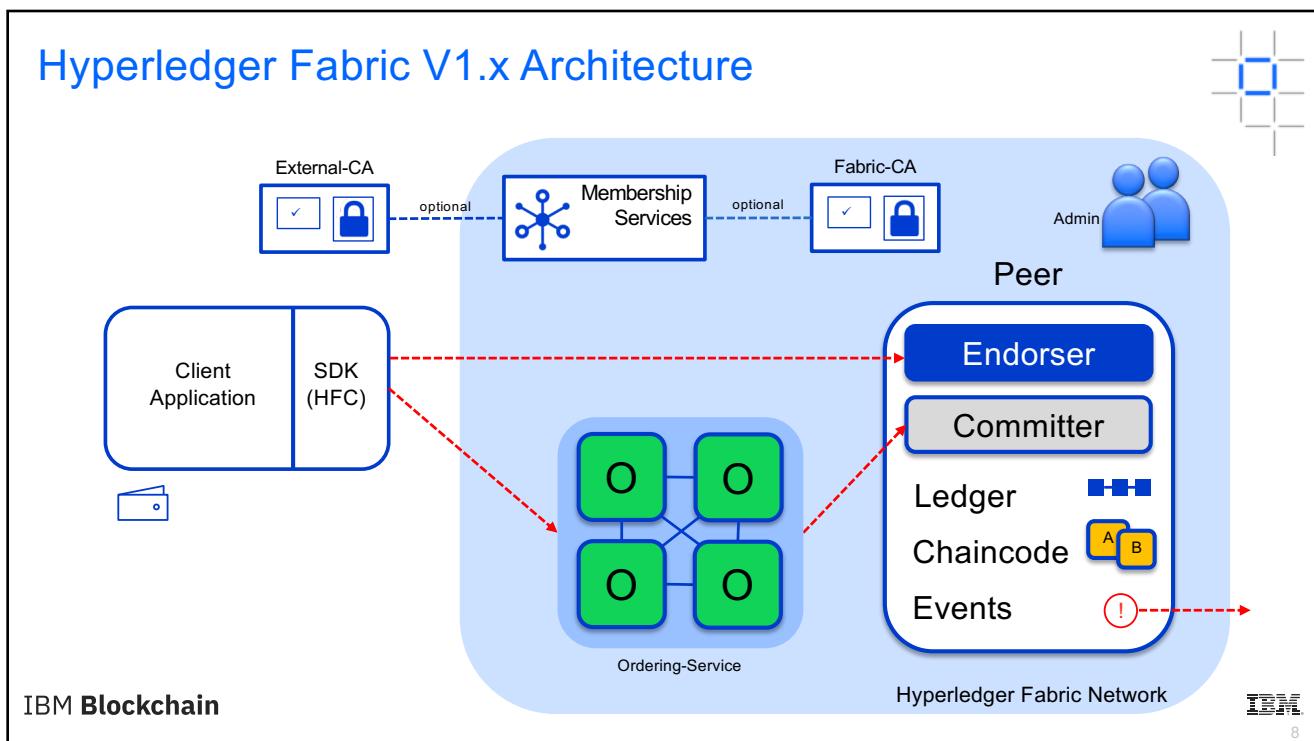
?

Technical Deep Dive

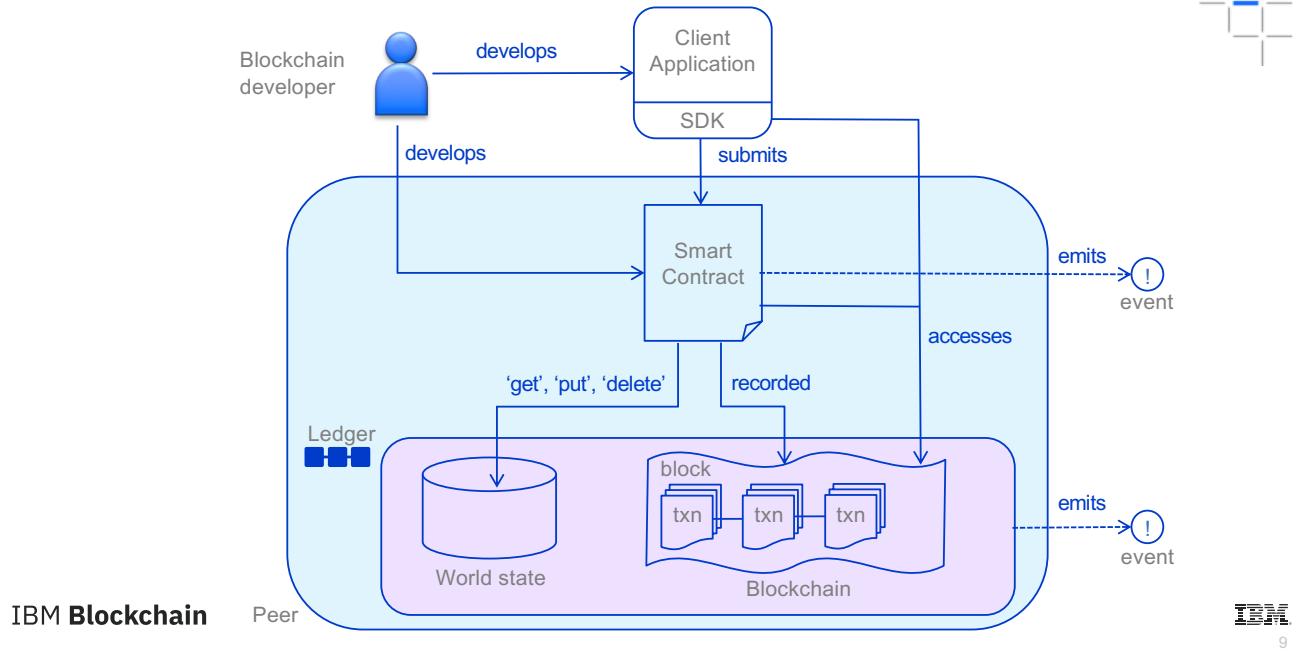
- [Architectural Overview]
- Network Consensus
- Channels and Ordering Service
- Components
- Network setup
- Endorsement Policies
- Membership Services



IBM Blockchain
IBM

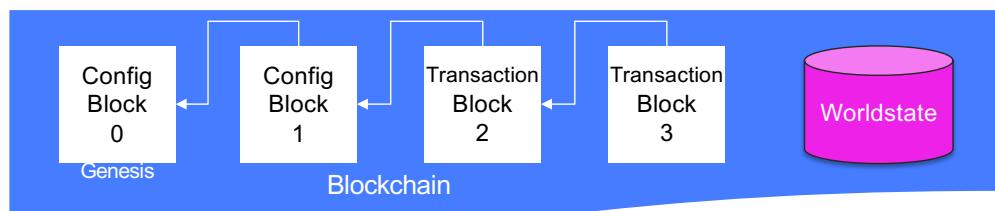


How applications interact with the ledger



Fabric Ledger

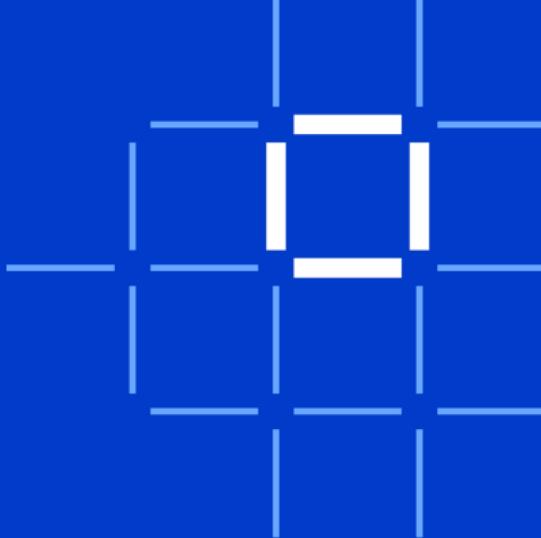
- The **Fabric ledger** is maintained by each peer and includes the **blockchain** and **worldstate**
- A separate ledger is maintained for each channel the peer joins
- Transaction **read/write sets** are written to the blockchain
- Channel configurations** are also written to the blockchain
- The worldstate can be either LevelDB (default) or CouchDB
 - LevelDB** is a simple key/value store
 - CouchDB** is a document store that allows complex queries
- The smart contact Contract decides what is written to the worldstate





Technical Deep Dive

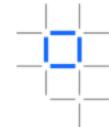
- Architectural Overview
- [Network Consensus]
- Channels and Ordering Service
- Components
- Network setup
- Endorsement Policies
- Membership Services



IBM Blockchain

IBM

Nodes and roles



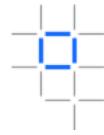
	Peer: Maintains ledger and state. Commits transactions. May hold smart contract (chaincode).
	Endorsing Peer: Specialized peer also endorses transactions by receiving a transaction proposal and responds by granting or denying endorsement. Must hold smart contract.
	Ordering Node: Approves the inclusion of transaction blocks into the ledger and communicates with committing and endorsing peer nodes. Does not hold smart contract. Does not hold ledger.

IBM Blockchain

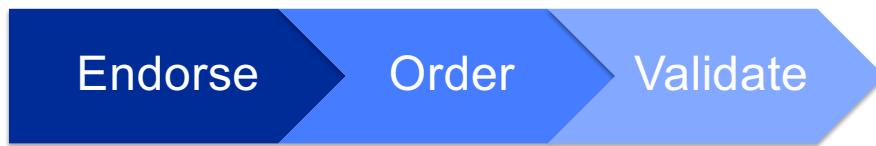
IBM

12

Hyperledger Fabric Consensus



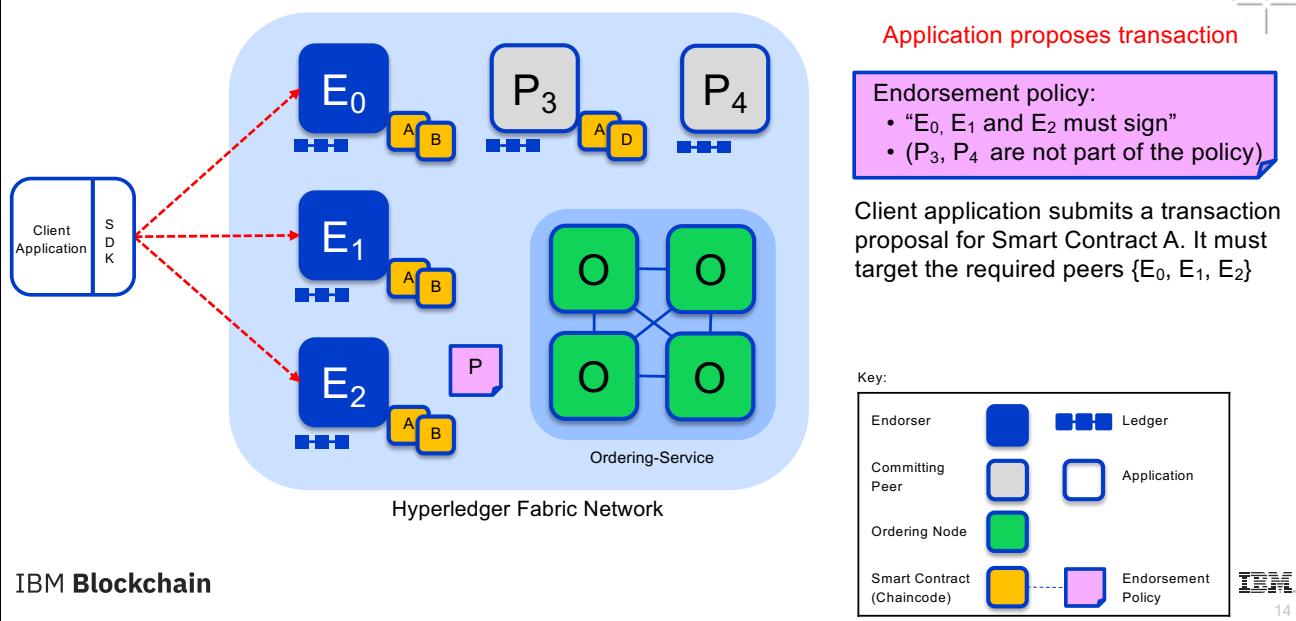
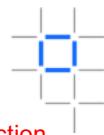
Consensus is achieved using the following transaction flow:



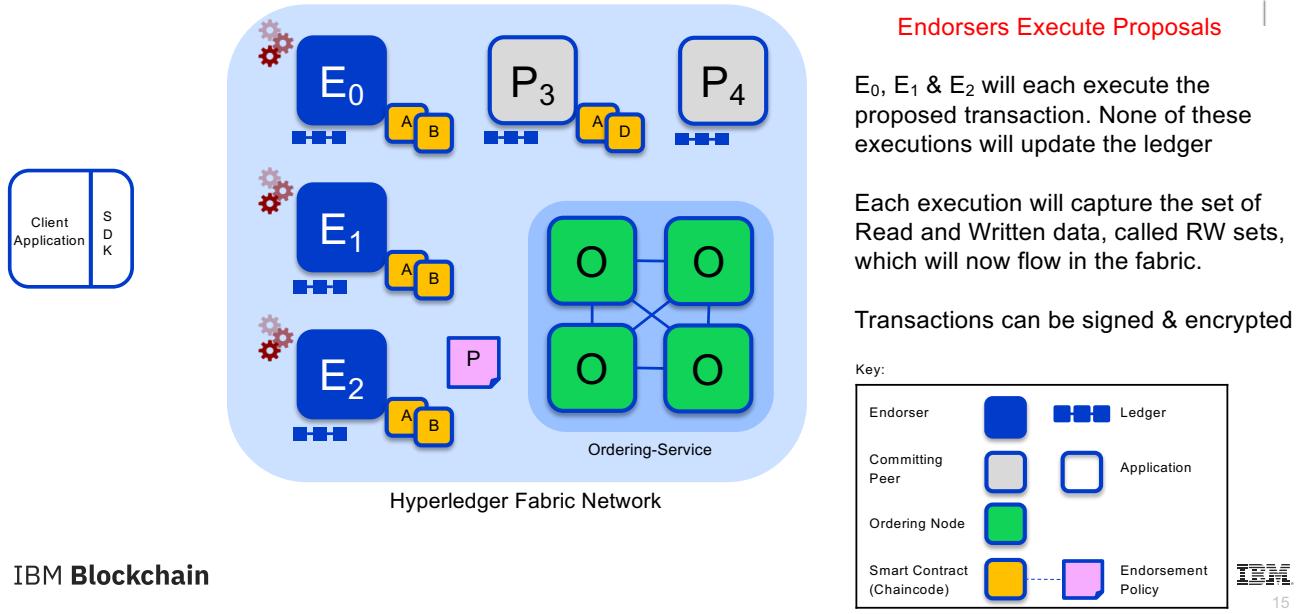
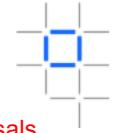
IBM Blockchain

IBM
13

Sample transaction: Step 1/7 – Propose transaction

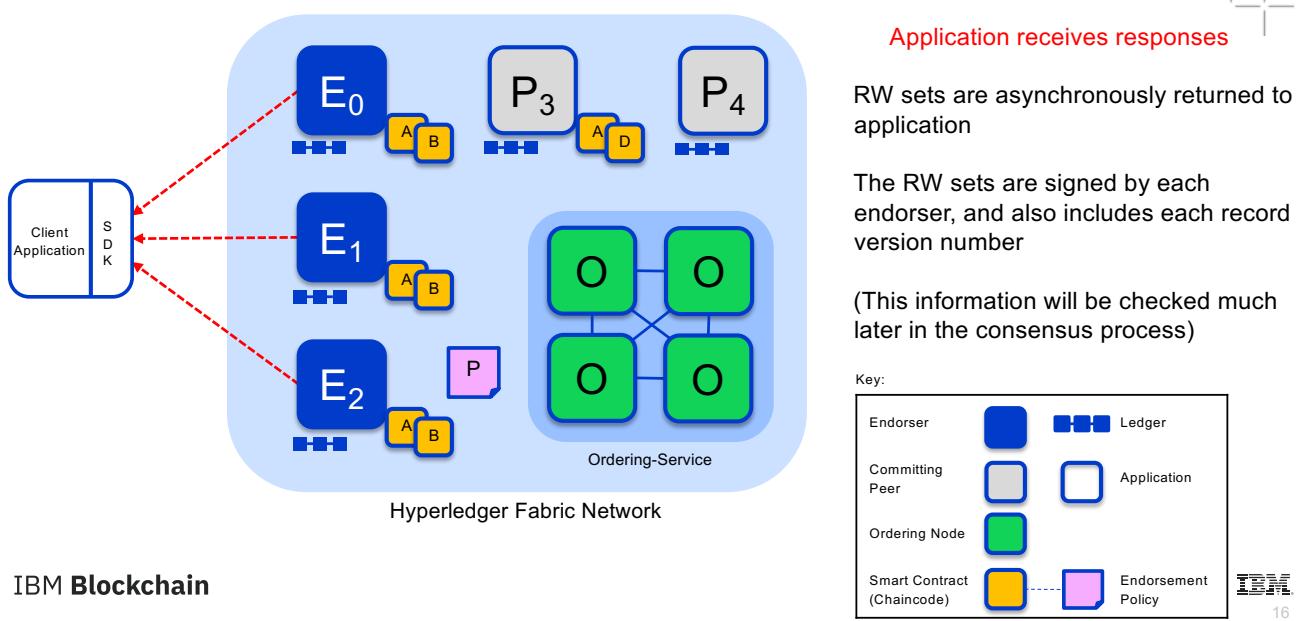
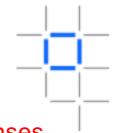


Sample transaction: Step 2/7 – Execute proposal



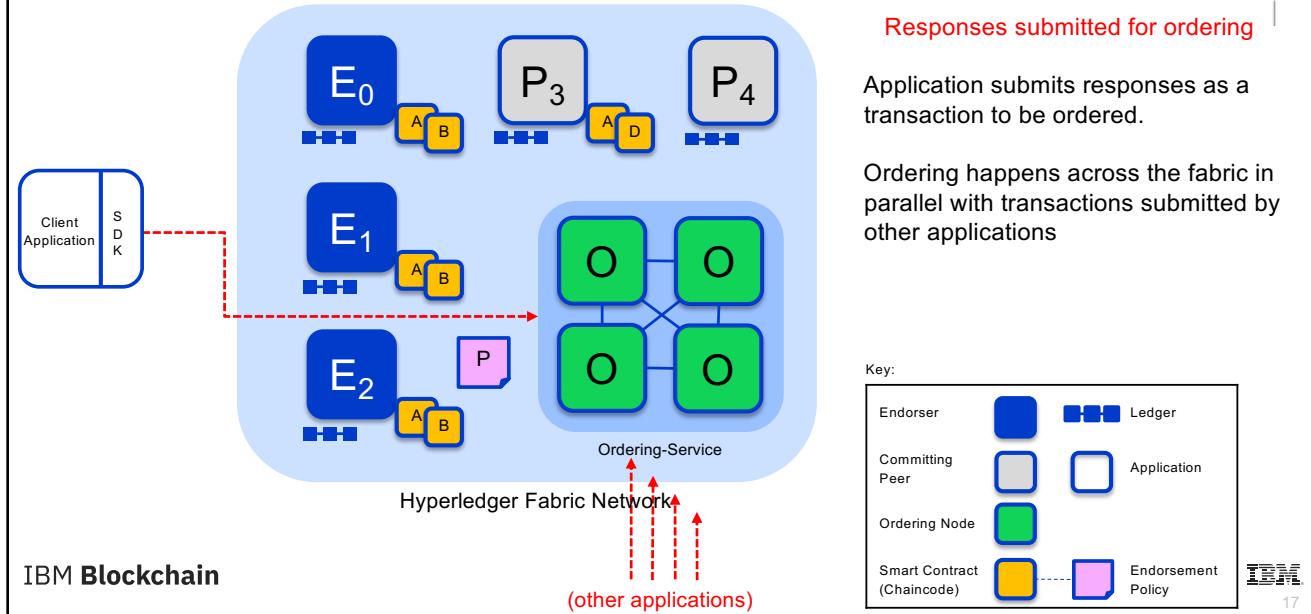
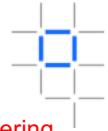
15

Sample transaction: Step 3/7 – Proposal Response

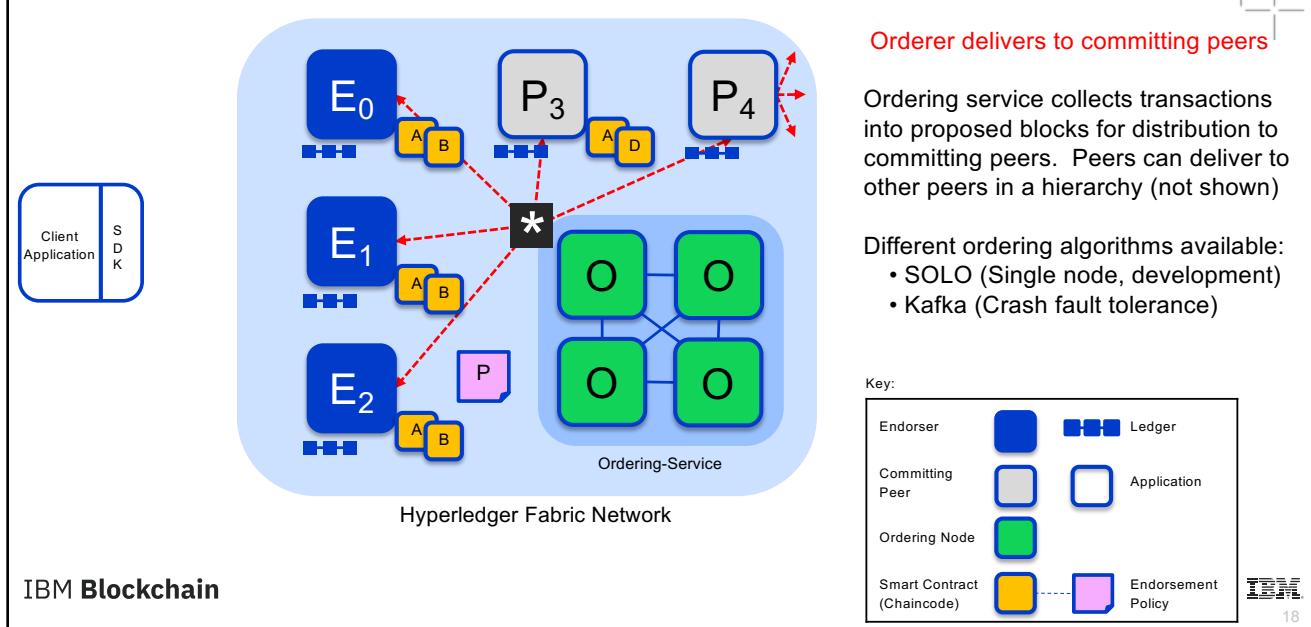
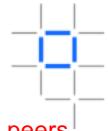


16

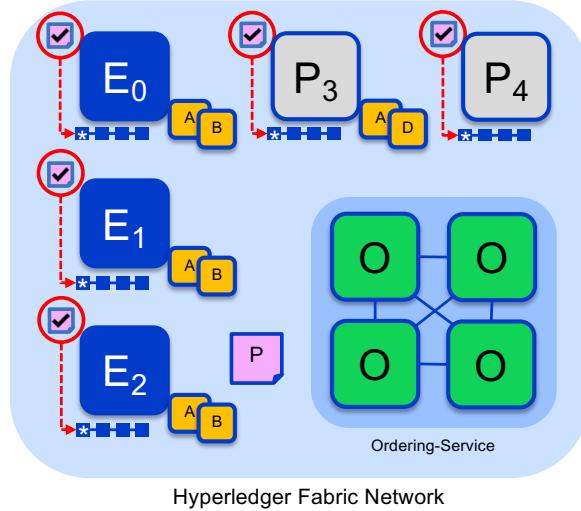
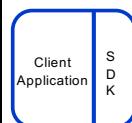
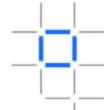
Sample transaction: Step 4/7 – Order Transaction



Sample transaction: Step 5/7 – Deliver Transaction



Sample transaction: Step 6/7 – Validate Transaction



Committing peers validate transactions

Every committing peer validates against the endorsement policy. Also check RW sets are still valid for current world state

Validated transactions are applied to the world state and retained on the ledger

Invalid transactions are also retained on the ledger but do not update world state

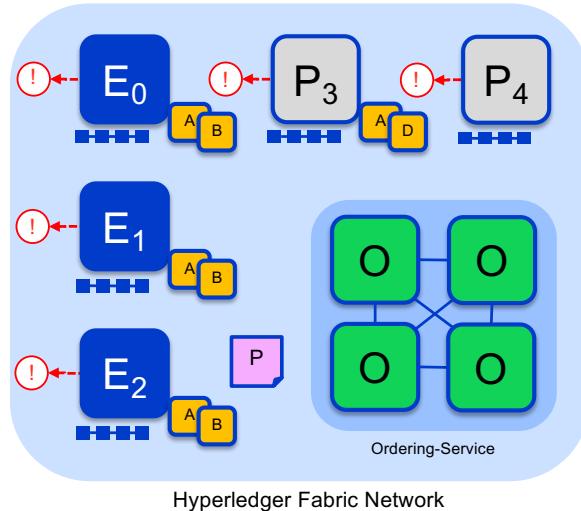
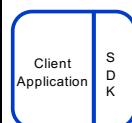
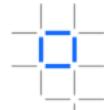
Key:

Endorser		Ledger
Committing Peer		Application
Ordering Node		
Smart Contract (Chaincode)		Endorsement Policy

IBM
19

IBM Blockchain

Sample transaction: Step 7/7 – Notify Transaction



Committing peers notify applications

Applications can register to be notified when transactions succeed or fail, and when blocks are added to the ledger

Applications will be notified by each peer to which they are connected

Key:

Endorser		Ledger
Committing Peer		Application
Ordering Node		
Smart Contract (Chaincode)		Endorsement Policy

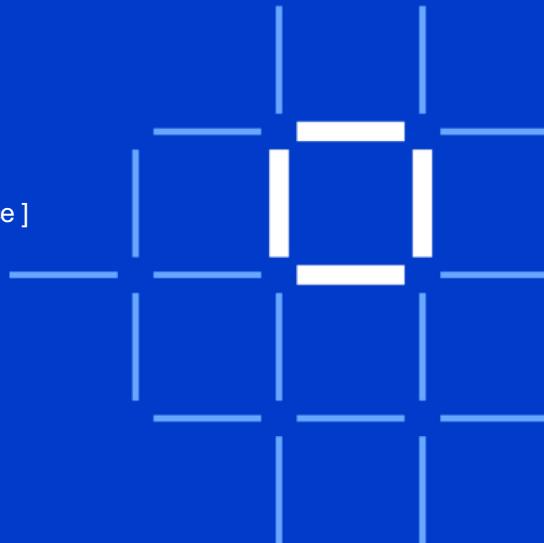
IBM
20

IBM Blockchain



Technical Deep Dive

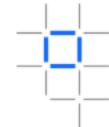
- Architectural Overview
- Network Consensus
- [Channels and Ordering Service]
- Components
- Network setup
- Endorsement Policies
- Membership Services



IBM Blockchain

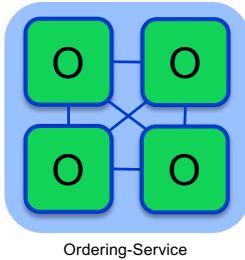
IBM

Ordering Service



The ordering service packages transactions into blocks to be delivered to peers. Communication with the service is via channels.

Different configuration options for the ordering service include:



- SOLO

- Single node for development

- Kafka : Crash fault tolerant consensus

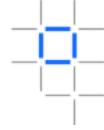
- 3 nodes minimum
- Odd number of nodes recommended

IBM Blockchain

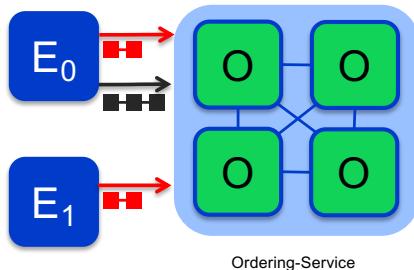
IBM

22

Channels



Channels provide privacy between different ledgers

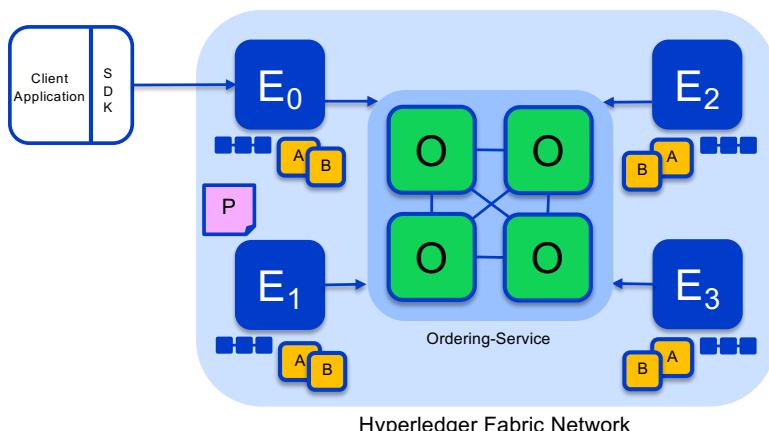
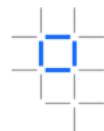


- Ledgers exist in the scope of a channel
 - Channels can be shared across an entire network of peers
 - Channels can be permissioned for a specific set of participants
- Chaincode is **installed** on peers to access the worldstate
- Chaincode is **instantiated** on specific channels
- Peers can participate in multiple channels
- Concurrent execution for performance and scalability

IBM Blockchain

IBM
23

Single Channel Network



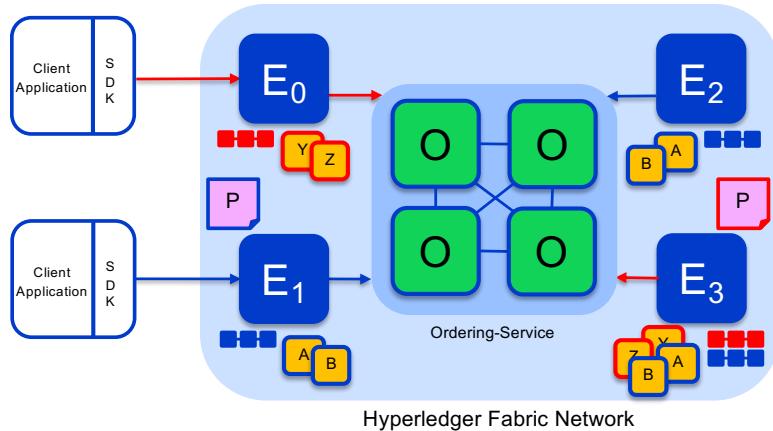
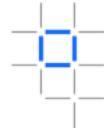
- Similar to v0.6 PBFT model
- All peers connect to the same system channel (blue).
- All peers have the same chaincode and maintain the same ledger
- Endorsement by peers E₀, E₁, E₂ and E₃

Key:	
Endorser	■
Committing Peer	□
Ordering Node	■
Smart Contract (Chaincode)	■
Endorsement Policy	---
Ledger	■
Application	□

IBM
24

IBM Blockchain

Multi Channel Network



- Peers E₀ and E₃ connect to the red channel for chaincodes Y and Z
- E₁, E₂ and E₃ connect to the blue channel for chaincodes A and B

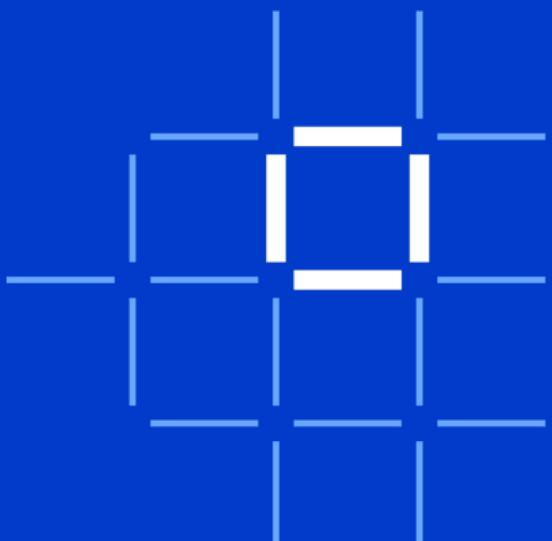
Key:	
Endorser	Ledger
Committing Peer	Application
Ordering Node	
Smart Contract (Chaincode)	Endorsement Policy

IBM
25



Technical Deep Dive

- Architectural Overview
- Network Consensus
- Channels and Ordering Service
- [Components]
- Network setup
- Endorsement Policies
- Membership Services

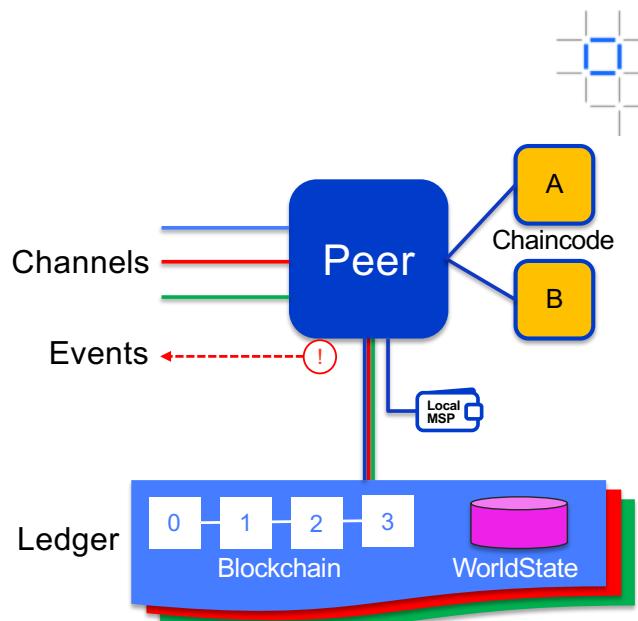


IBM Blockchain

IBM

Fabric Peer

- Each peer:
 - Connects to one or more **channels**
 - Maintains one or more **ledgers** per channel
 - Maintains **installed chaincode**
 - Manages **runtime docker containers** for **instantiated chaincode**
 - Chaincode is instantiated on a channel
 - Runtime docker container shared by channels with same chaincode instantiated (no state stored in container)
 - Has a local MSP (Membership Services Provider) that provides **crypto material**
 - **Emits events** to the client application

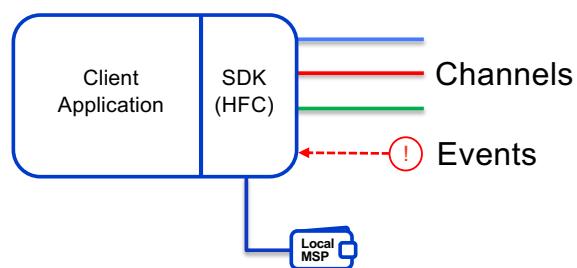


IBM Blockchain

IBM
27

Client Application

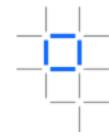
- Each client application uses Fabric SDK to:
 - Connects over channels to one or more peers
 - Connects over channels to one or more orderer nodes
 - Receives events from peers
 - Local MSP provides client **crypto material**
 - Client can be written in different languages (Node.js, Go, Java, Python?)



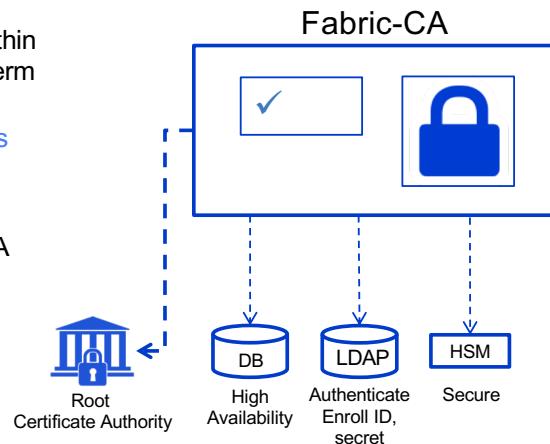
IBM Blockchain

IBM
28

Fabric-CA



- Default (optional) Certificate Authority within Fabric network for issuing [Ecerts](#) (long-term identity)
- Supports clustering for [HA characteristics](#)
- Supports LDAP for [user authentication](#)
- Supports HSM for [security](#)
- Can be configured as an intermediate CA

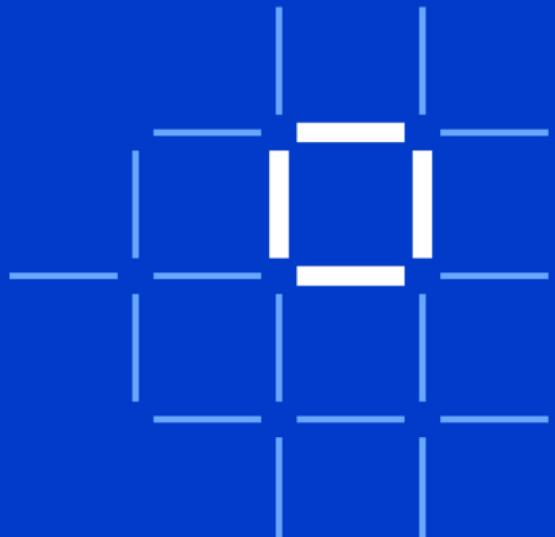


IBM Blockchain

IBM
29

Technical Deep Dive

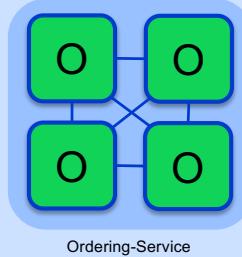
- Architectural Overview
- Network Consensus
- Channels and Ordering Service
- Components
- [Network setup]
- Endorsement Policies
- Membership Services



IBM Blockchain

IBM

Bootstrap Network (1/6) - Configure & Start Ordering Service



Hyperledger Fabric Network

An Ordering Service is configured and started for the network:
\$ docker-compose [-f orderer.yml] ...

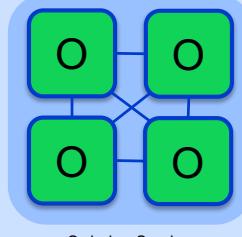
IBM Blockchain

IBM
31

Bootstrap Network (2/6) - Configure and Start Peer Nodes

E₀

E₁



E₂

P₃

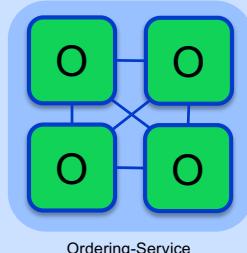
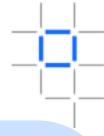
Hyperledger Fabric Network

A peer is configured and started for each Endorser or Committer in the network:
\$ peer node start ...

IBM Blockchain

IBM
32

Bootstrap Network (3/6) - Install Chaincode



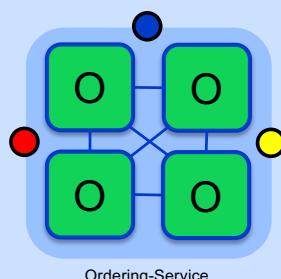
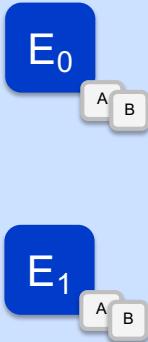
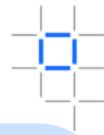
Hyperledger Fabric Network

Chaincode is installed onto each Endorsing Peer that needs to execute it:
\$ peer chaincode install ...

IBM Blockchain

IBM
33

Bootstrap Network (4/6) – Create Channels



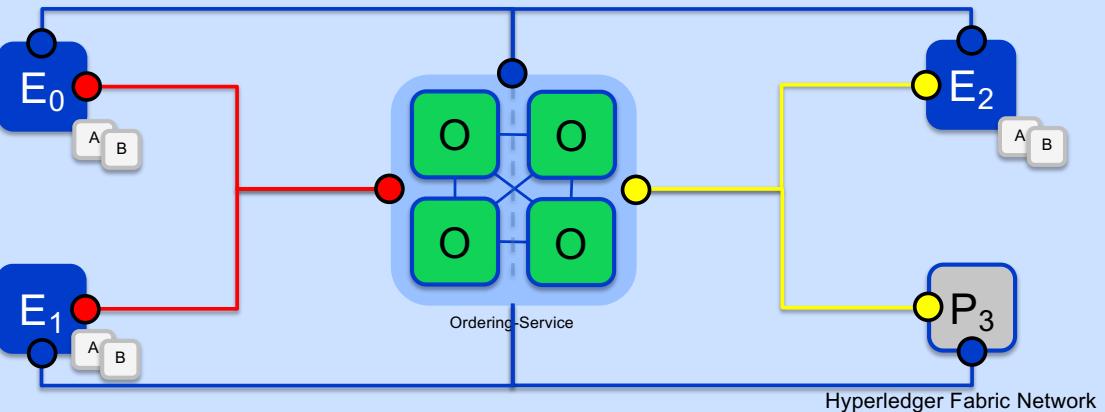
Hyperledger Fabric Network

Channels are created on the ordering service:
\$ peer channel create -o [orderer] ...

IBM Blockchain

IBM
34

Bootstrap Network (5/6) – Join Channels

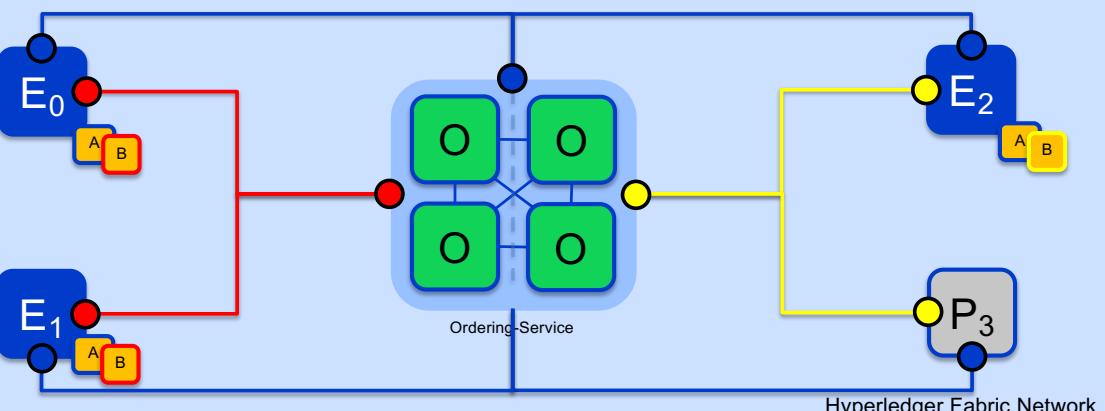


Peers that are permissioned can then join the channels they want to transact on:
\$ peer channel join ...

IBM Blockchain

35

Bootstrap Network (6/6) – Instantiate Chaincode



Peers finally instantiate the Chaincode on the channels they want to transact on:
\$ peer chaincode instantiate ... -P 'policy'

IBM Blockchain

An Endorsement Policy is specified and once instantiated chaincode can process transactions.

36

?

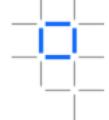
Technical Deep Dive

- Architectural Overview
- Network Consensus
- Channels and Ordering Service
- Components
- Network setup
- [Endorsement Policies]
- Membership Services

IBM Blockchain

IBM

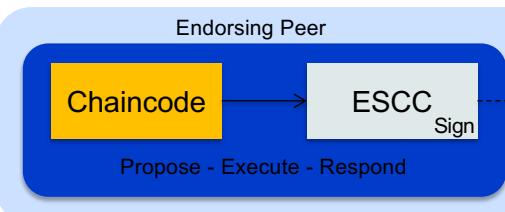
Endorsement Policies



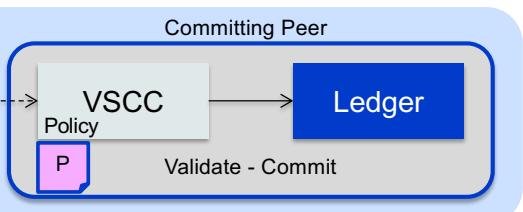
An endorsement policy describes the conditions by which a transaction can be endorsed. A transaction can only be considered valid if it has been endorsed according to its policy.

- Each chaincode is deployed with an Endorsement Policy
- **ESCC** (Endorsement System ChainCode) signs the proposal response on the endorsing peer
- **VSCC** (Validation System ChainCode) validates the endorsements

Endorsing Peer



Committing Peer

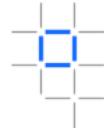


IBM Blockchain

IBM
38

Endorsement Policy Syntax

```
$ peer chaincode instantiate
-C mychannel
-n mycc
-v 1.0
-p chaincode_example02
-c '{"Args":["init","a", "100", "b", "200"]}'
-P "AND('Org1MSP.member')"
```



Instantiate the chaincode `mycc` on channel `mychannel` with the policy `AND('Org1MSP.member')`

Policy Syntax: `EXPR(E[, E...])`

Where `EXPR` is either AND or OR and `E` is either a principal or nested EXPR

Principal Syntax: `MSP.ROLE`

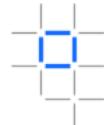
Supported roles are: member and admin

Where `MSP` is the MSP ID, and `ROLE` is either “member” or “admin”

IBM Blockchain

IBM
39

Endorsement Policy Examples



Examples of policies:

- Request 1 signature from all three principals
 - `AND('Org1.member', 'Org2.member', 'Org3.member')`
- Request 1 signature from either one of the two principals
 - `OR('Org1.member', 'Org2.member')`
- Request either one signature from a member of the Org1 MSP or (1 signature from a member of the Org2 MSP and 1 signature from a member of the Org3 MSP)
 - `OR('Org1.member', AND('Org2.member', 'Org3.member'))`

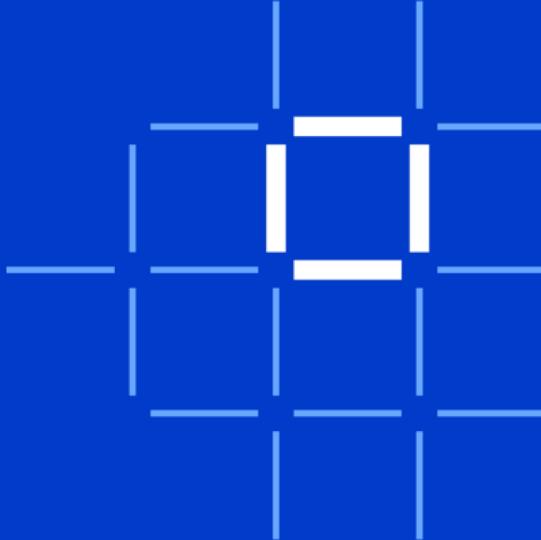
IBM Blockchain

IBM
40



Technical Deep Dive

- Architectural Overview
- Network Consensus
- Channels and Ordering Service
- Components
- Network setup
- Endorsement Policies
- [Membership Services]

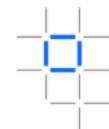


IBM Blockchain

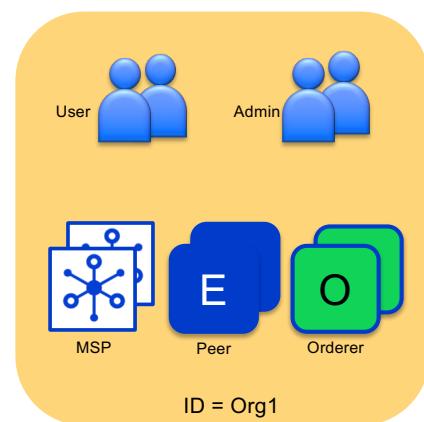
IBM

Organisations

Organisations define boundaries within a Fabric Blockchain Network



- Each organisation defines:
 - Membership Services Provider (MSP) for identities
 - Administrator(s)
 - Users
 - Peers
 - Orderers (optional)
- A network can include many organisations representing a consortium
- Each organisation has an ID



IBM Blockchain

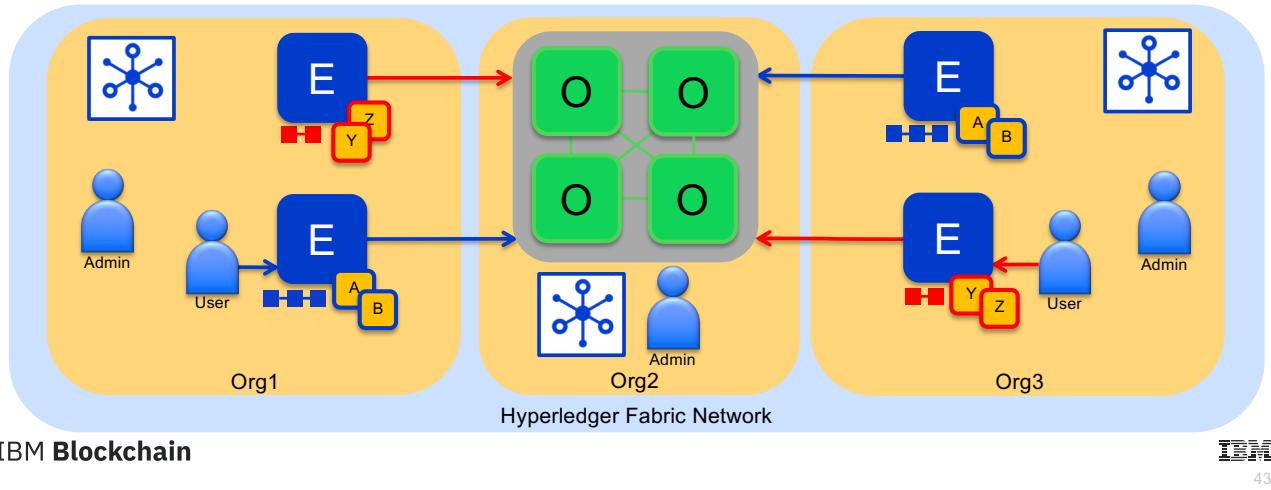
IBM

42

Consortium Network

An example consortium network of 3 organisations

- Orgs 1 and 3 run peers
- Org 2 provides the ordering service only



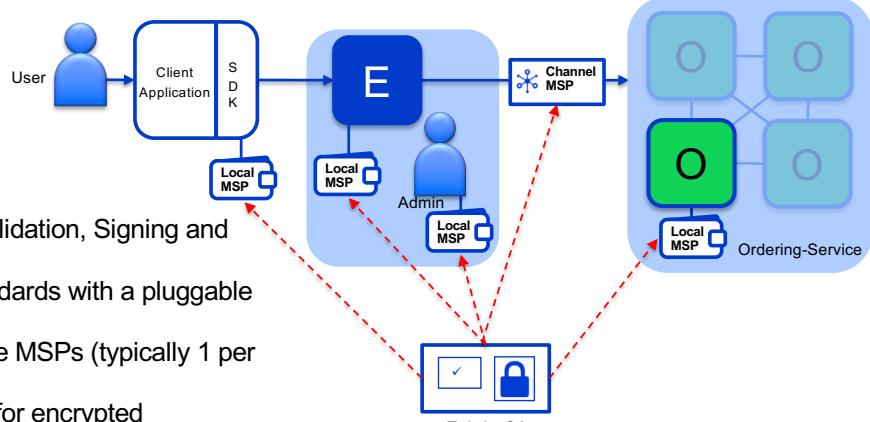
IBM Blockchain

IBM
43

Membership Services Provider - Overview

A MSP manages a set of identities within a distributed Fabric network

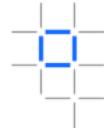
- Provides identity for:
 - Peers and Orderers
 - Client Applications
 - Administrators
- Identities can be issued by:
 - Fabric-CA
 - An external CA
- Provides: Authentication, Validation, Signing and Issuance
- Supports different crypto standards with a pluggable interface
- A network can include multiple MSPs (typically 1 per org)
- Includes TLS crypto material for encrypted communications



IBM Blockchain

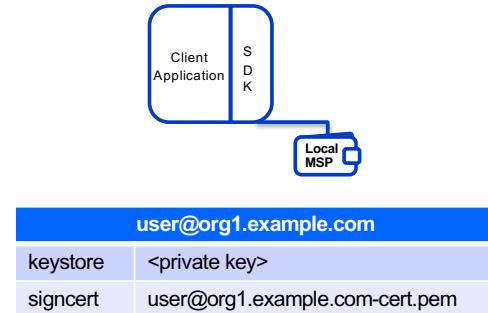
IBM
44

User Identities



Each client application has a local MSP to store user identities

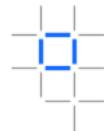
- Each local MSP includes:
 - **Keystore**
 - **Private key** for signing transactions
 - **Signcert**
 - **Public x.509 certificate**
- May also include TLS credentials
- Can be backed by a Hardware Security Module (HSM)



IBM Blockchain

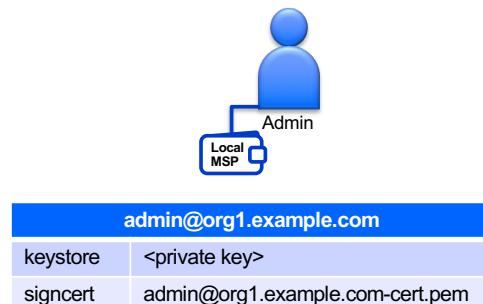
IBM
45

Admin Identities



Each Administrator has a local MSP to store their identity

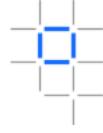
- Each local MSP includes:
 - **Keystore**
 - **Private key** for signing transactions
 - **Signcert**
 - **Public x.509 certificate**
- May also include TLS credentials
- Can be backed by a Hardware Security Module (HSM)



IBM Blockchain

IBM
46

Peer and Orderer Identities



Each peer and orderer has a local MSP

- Each local MSP includes:
 - **keystore**
 - **Private key** for signing transactions
 - **signcert**
 - **Public x.509 certificate**
- In addition Peer/Orderer MSPs identify authorized administrators:
 - **admincerts**
 - List of **administrator certificates**
 - **cacerts**
 - The **CA public cert** for verification
 - **crls**
 - List of **revoked certificates**
- Peers and Orderers also receive channel MSP info
- Can be backed by a Hardware Security Module (HSM)

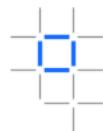


peer@org1.example.com	
admincerts	admin@org1.example.com-cert.pem
cacerts	ca.org1.example.com-cert.pem
keystore	<private key>
signcert	peer@org1.example.com-cert.pem
crls	<list of revoked admin certificates>

IBM Blockchain

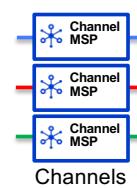


Channel MSP information



Channels include additional organisational MSP information

- Determines which orderers or peers can join the channel
- Determines client applications read or write access to the channel
- Stored in configuration blocks in the ledger
- Each channel MSP includes:
 - **admincerts**
 - Any public certificates for administrators
 - **cacerts**
 - The CA public certificate for this MSP
 - **crls**
 - List of revoked certificates
- **Does not include any private keys for identity**

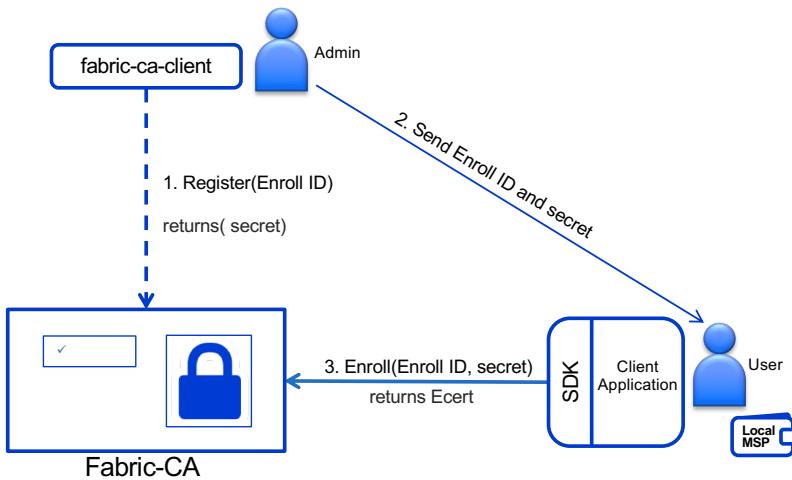
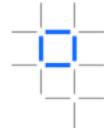


ID = MSP1	
admincerts	admin.org1.example.com-cert.pem
cacerts	ca.org1.example.com-cert.pem
crls	<list of revoked admin certificates>

IBM Blockchain



New User Registration and Enrollment



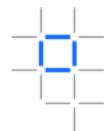
Registration and Enrollment

- Admin registers new user with Enroll ID
- User enrolls and receives credentials
- Additional offline registration and enrollment options available

IBM Blockchain

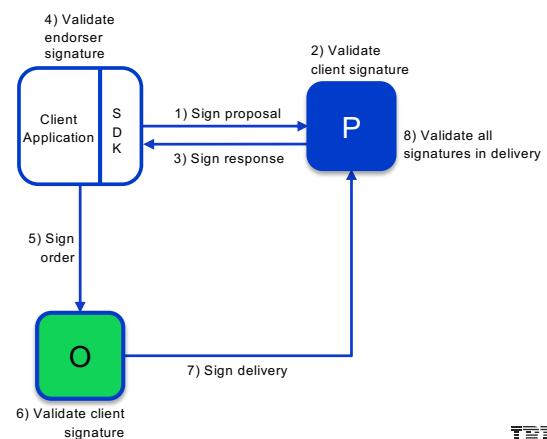
49

Transaction Signing



All transactions within a Hyperledger Fabric network are signed by permissioned actors, and those signatures validated

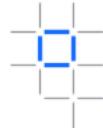
- Actors sign transactions with their enrolment private key
 - Stored in their local MSP
- Components validate transactions and certificates
 - Root CA certificates and CRLs stored in local MSP
 - Root CA certificates and CRLs stored in Org MSP in channel



IBM Blockchain

50

Summary and Next Steps

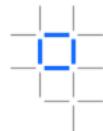


- Apply shared ledgers and smart contracts to your Business Network
- Think about your participants, assets and business processes
- Spend time thinking about realistic business use cases
- Get some hands-on experience with the technology
- Start with a First Project
- IBM can help with your journey

IBM Blockchain

IBM
51

Further Hyperledger Fabric Information



- Project Home: <https://www.hyperledger.org/projects/fabric>
- GitHub Repo: <https://github.com/hyperledger/fabric>
- Latest Docs: <https://hyperledger-fabric.readthedocs.io/en/latest/>
- Community Chat: <https://chat.hyperledger.org/channel/fabric>
- Project Wiki: <https://wiki.hyperledger.org/projects/fabric>
- Design Docs: <https://wiki.hyperledger.org/community/fabric-design-docs>

IBM Blockchain

IBM
52

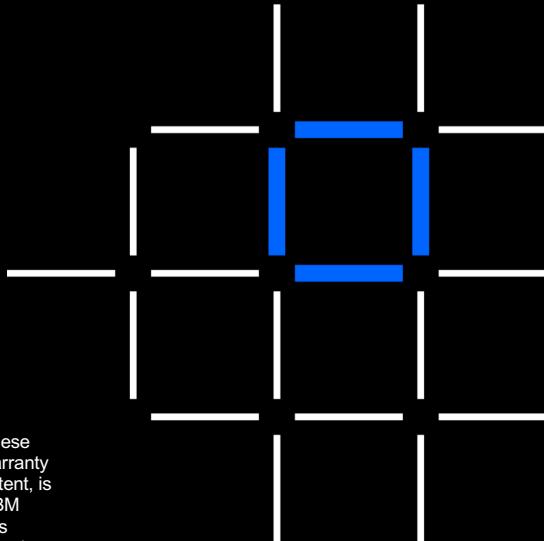
Thank you

IBM Blockchain

www.ibm.com/blockchain

developer.ibm.com/blockchain

www.hyperledger.org



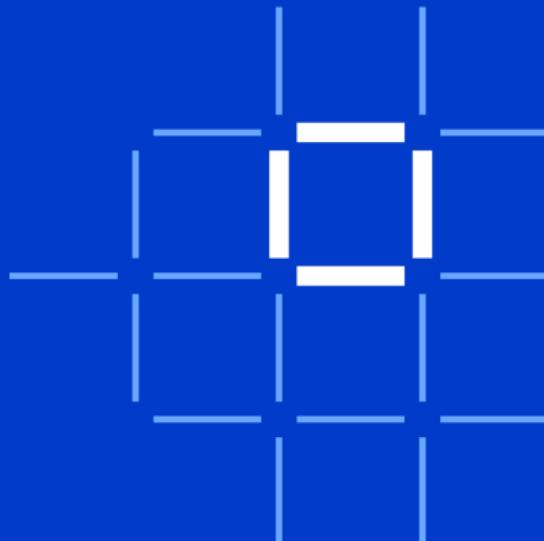
IBM

© Copyright IBM Corporation 2017. All rights reserved. The information contained in these materials is provided for informational purposes only, and is provided AS IS without warranty of any kind, express or implied. Any statement of direction represents IBM's current intent, is subject to change or withdrawal, and represents only goals and objectives. IBM, the IBM logo, and other IBM products and services are trademarks of the International Business Machines Corporation, in the United States, other countries or both. Other company, product, or service names may be trademarks or service marks of others.



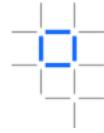
Appendix

IBM Blockchain



IBM

Fabric Commands

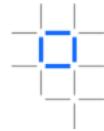


- Fabric has the following commands:
 - **peer** ... (For operating and configuring a peer)
 - **peer chaincode** ... (Manages chaincode on the peer)
 - **peer channel** ... (Manages channels on the peer)
 - **peer node** ... (Manages the peer)
 - **peer version** (Returns the peer version)
 - **cryptogen** ... (Utility for generating crypto material)
 - **configtxgen** ... (Creates configuration data such as the genesis block)
 - **configtxlator** ... (Utility for generating channel configurations)
 - **fabric-ca-client** ... (Manages identities)
 - **fabric-ca-server** ... (Manages the Fabric-CA server)

IBM Blockchain

55

Configuration Detail

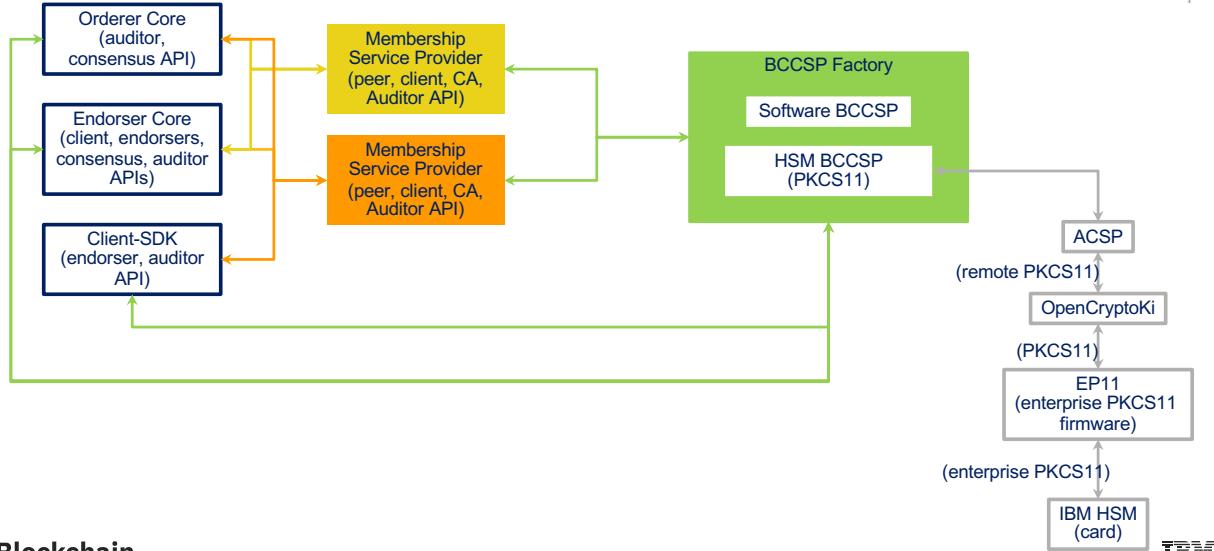


Path	MSP ID	Attributes	Attributes
config -> channel_group -> groups -> application -> groups	Org1MSP	mod_policy	Admins
	policies -> Admins	mod_policy	Admins
		policy -> value -> identities	Org1MSP, Admin
		Policy -> value -> rule	n_out_of
	Policies -> Readers	Mod_policy	Admins
		Policy -> value -> identities	Org1MSP
		Policy -> value -> rule	N_out_of
	Policies -> Writers	Mod_policy	Admins
		Policy -> value -> identities	Org1MSP
		Policy -> value -> rule	N_out_of

IBM Blockchain

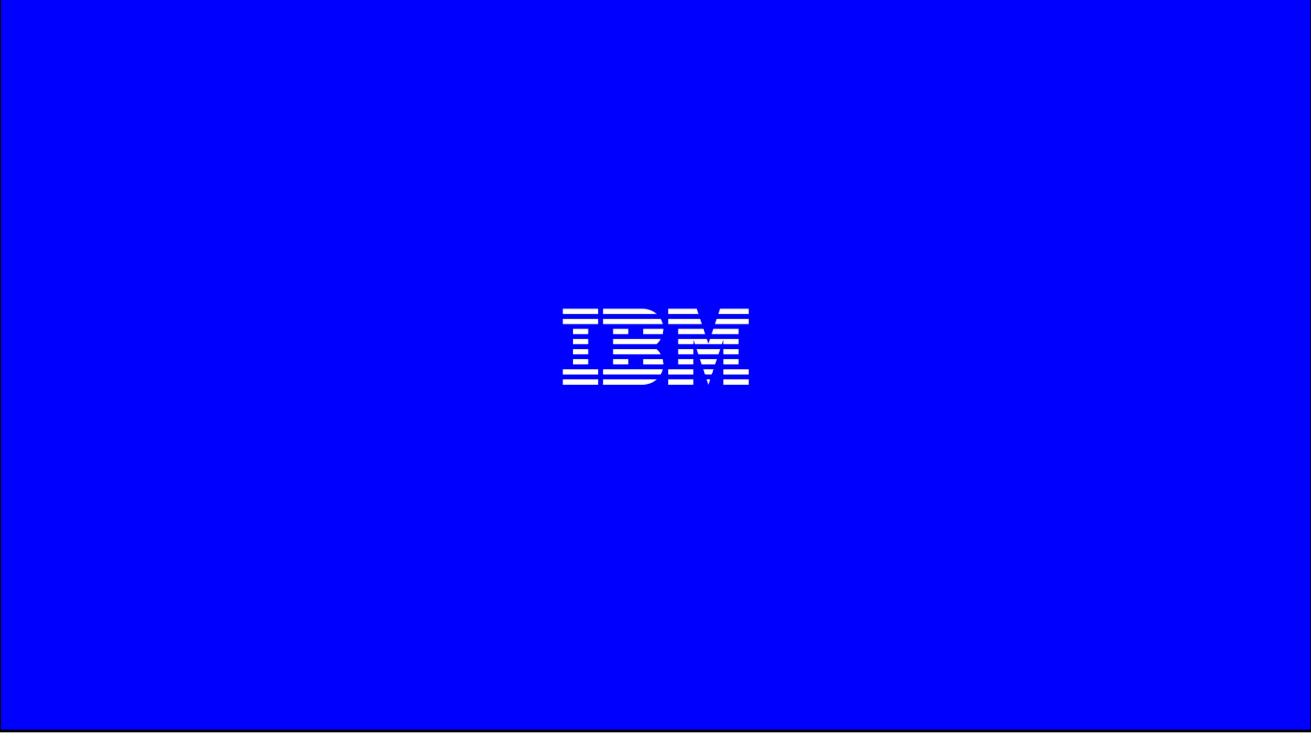
56

MSP and BCCSP (Modularity and Decentralisation)



Blockchain Crypto Service Provider (BCCSP)

- Pluggable implementation of cryptographic standards and algorithms.
- **Pluggability**
 - alternate implementations of crypto interface can be used within the Hyperledger Fabric code, without modifying the core
- **Support for Multiple CSPs**
 - Easy addition of more types of CSPs, e.g., of different HSM types
 - Enable the use of different CSP on different system components transparently
- **International Standards Support**
 - E.g., via a new/separate CSP
 - Interoperability among standards is not necessarily guaranteed



IBM