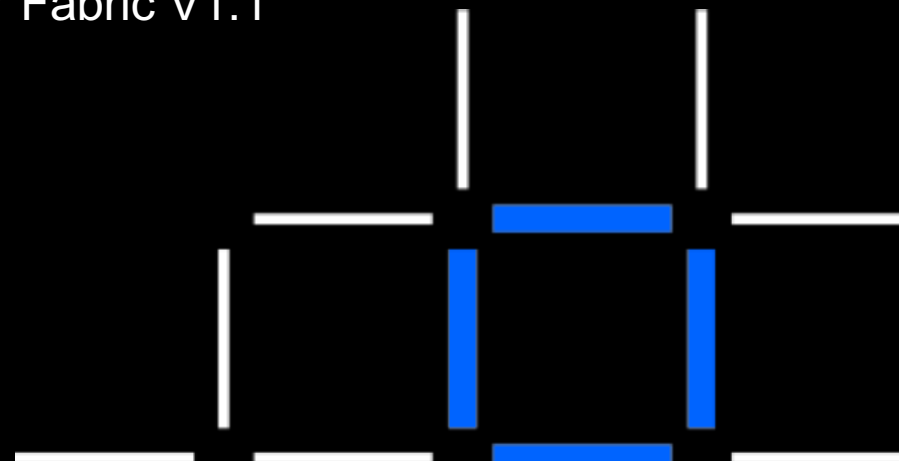







# What's New

A Technical Deep-Dive of new features in Hyperledger Fabric V1.1



Blockchain Explored Series

-  IBM Blockchain Platform Explored
-  Architectures Explored
-  Fabric Explored
-  Composer Explored
-  **What's New**


V0.9, 23 May 2018

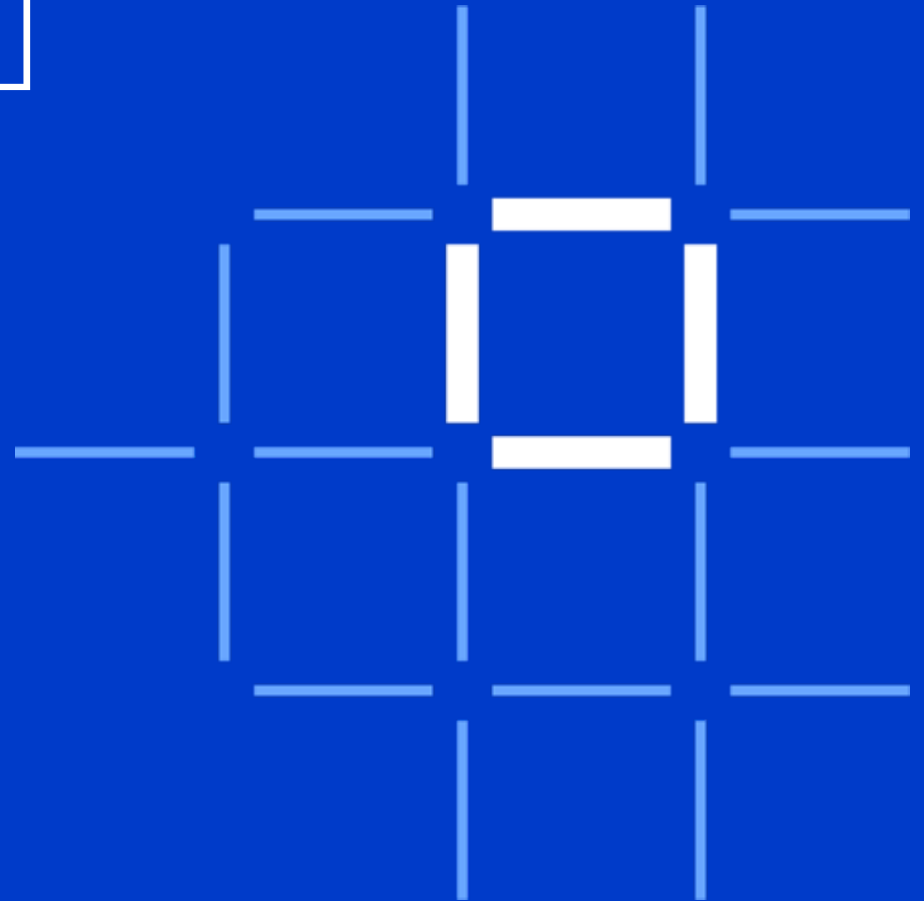
IBM Blockchain



# [ Hyperledger Fabric Roadmap and Architecture ]

 Fabric 1.1 new features

 Fabric 1.1 Experimental  
Features



# Hyperledger Fabric Roadmap

## V1 Alpha

- Docker images
- Tooling to bootstrap network
- Fabric CA or bring your own
- Java and Node.js SDKs
- Ordering Services - Solo and Kafka
- Endorsement policy
- Level DB and Couch DB
- Block dissemination across peers via Gossip

## V1 GA

- Hardening, usability, serviceability, load, operability and stress test
- Chaincode ACL
- Chaincode packaging & LCI
- Pluggable crypto
- HSM support
- Consumability of configuration
- Next gen bootstrap tool (config update)
- Config transaction lifecycle
- Eventing security
- Cross Channel Query
- Peer management APIs
- Documentation

## V1.1

- Node.js smart contracts
- Node.js connection profile
- Smart Contract APIs:
  - Encryption library
  - Txn submitter identity
  - Access control (using above)
- Performance & Scale
  - More orderers at scale
  - Parallel txn validation
  - CouchDB indexes
- Events
  - Per channel vs global
  - Block info minimal events
- CSR for more secure certs
- Serviceability
  - Upgrade from 1.0
- **Technical Preview features**
  - Private channel data
  - Finer grained access control on channels (beyond orgs)
  - ZKP features (ID Mixer)
  - Java for Smart contracts

## V1.2

- V1.1 Technical Preview features
  - Finalize Side DB - Private Data
  - Finalize Java chaincode
  - Finalize Fabric ACL mechanism
- Usability Features
  - e.g. Service discovery
- Technical Debt/Hygiene
  - e.g. testing frameworks
  - Parallel testing
  - More modular code
- Pluggable endorsement and validation
- State-based Endorsement
- Privacy-preserving state-based endorsement

Based on <https://wiki.hyperledger.org/projects/fabric/roadmap>

March 2017

July 2017

March 2018

June 2018 (quarterly)



**HYPERLEDGER**  
BLOCKCHAIN TECHNOLOGIES FOR BUSINESS

\* Dates determined by the Hyperledger community, subject to change

# Hyperledger Fabric Roadmap

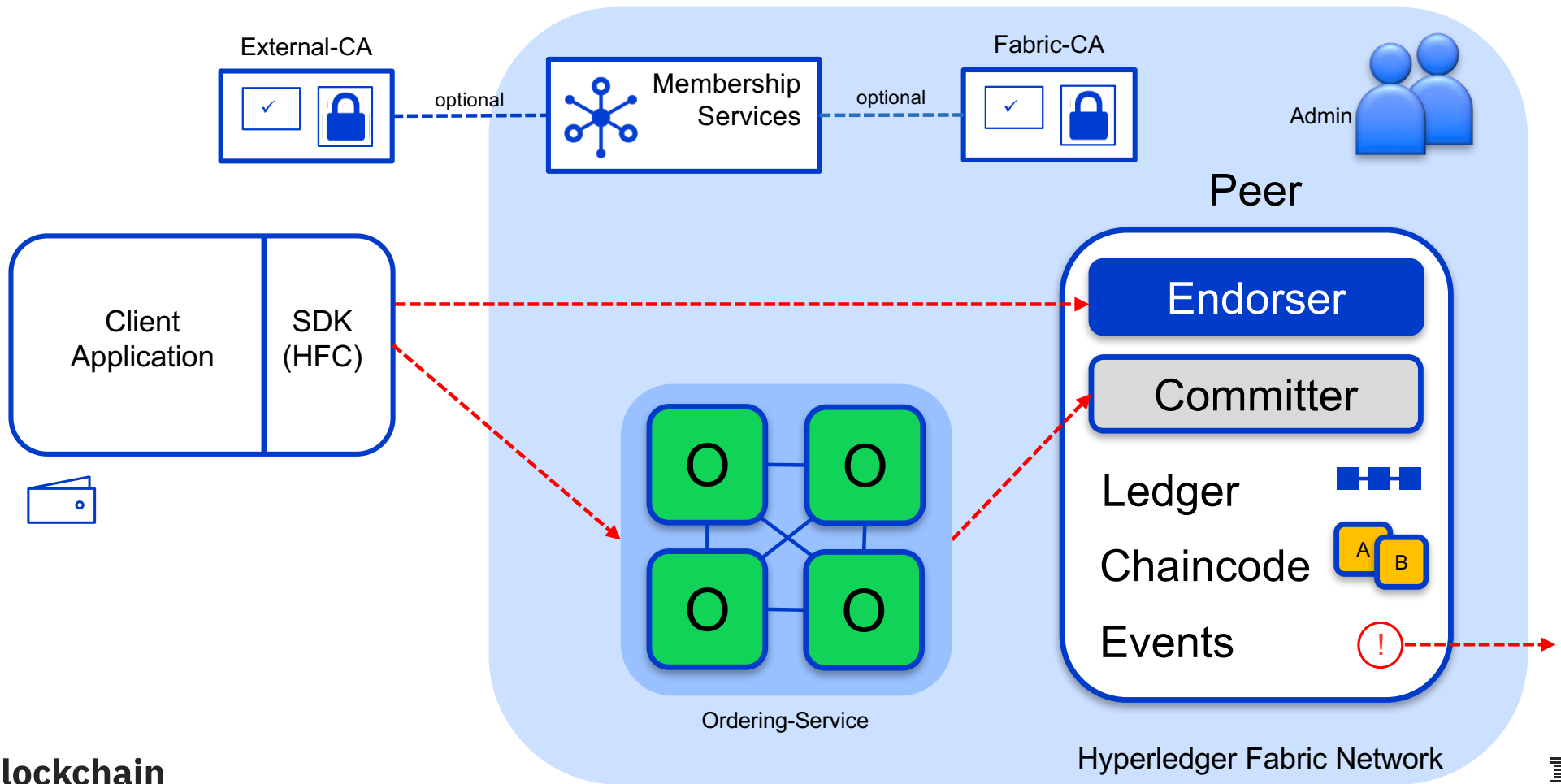
V1 GA	V1.1	V1.2 <sup>(*)</sup>	<i>Future<sup>(*)</sup></i>
<ul style="list-style-type: none"> <li>• Reliability, Servicability, Availability</li> <li>• Chaincode ACL</li> <li>• Chaincode Packaging &amp; LCI</li> <li>• Pluggable Crypto</li> <li>• HSM support</li> <li>• Next gen bootstrap tool (config update)</li> <li>• Config transaction lifecycle</li> <li>• Event security</li> <li>• Cross Channel Query</li> <li>• Peer management APIs</li> <li>• Documentation</li> <li>• Fabric CA or bring your own</li> <li>• Java and Node.js SDKs</li> <li>• Ordering Services - Solo and Kafka</li> <li>• Endorsement policy</li> <li>• Level DB and Couch DB</li> <li>• Block dissemination across peers via Gossip</li> </ul>	<ul style="list-style-type: none"> <li>• Node.js Smart Contracts</li> <li>• Node.js Connection Profile</li> <li>• Smart Contract APIs: <ul style="list-style-type: none"> <li>• Encryption Library</li> <li>• Txn Submitter Identity</li> <li>• Access Control (using above)</li> </ul> </li> <li>• Performance &amp; Scale <ul style="list-style-type: none"> <li>• More Orderers at scale</li> <li>• Parallel Txn Validation</li> <li>• CouchDB Indexes</li> </ul> </li> <li>• Events <ul style="list-style-type: none"> <li>• Per Channel vs global</li> <li>• Block Info Minimal Events</li> </ul> </li> <li>• CSR for more secure certs</li> <li>• Serviceability <ul style="list-style-type: none"> <li>• Upgrade from 1.0</li> </ul> </li> <li>• <i>Technical Preview features</i> <ul style="list-style-type: none"> <li>• Private Channel Data</li> <li>• Finer Grained access control on channels (beyond orgs)</li> <li>• ZKP Features (ID Mixer)</li> <li>• Java Smart Contracts</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>• Quarterly Updates</li> <li>• V1.1 Technical Preview features <ul style="list-style-type: none"> <li>• Finalize Private Channel Data</li> <li>• Finalize Fabric ACL mechanism</li> </ul> </li> <li>• Service Discovery</li> <li>• Technical Debt/Hygiene <ul style="list-style-type: none"> <li>• e.g. testing frameworks</li> <li>• Parallel testing</li> <li>• More modular code</li> </ul> </li> <li>• Pluggable endorsement and validation</li> <li>• Base Identity Mixer support</li> </ul>	<ul style="list-style-type: none"> <li>• State-based Endorsement</li> <li>• Side DB – Local Collections</li> <li>• Java chaincode</li> <li>• CouchDB safe pagination</li> <li>• Operational Metrics</li> <li>• Identity Mixer extensions</li> <li>• SDK improvements</li> <li>• MSP Refactoring</li> <li>• Chaincode Lifecycle Improvements</li> <li>• Raft/BFT consensus</li> </ul>
July 2017	March 2018	June 2018 <sup>(*)</sup>	September 2018 <sup>(*)</sup>

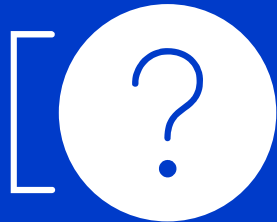


**HYPERLEDGER**  
BLOCKCHAIN TECHNOLOGIES FOR BUSINESS

Based on <https://wiki.hyperledger.org/projects/fabric/roadmap> (\*) Dates and content determined by the Hyperledger community and subject to change

# Hyperledger Fabric V1 Architecture



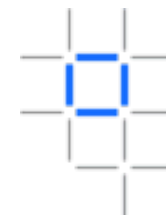


## Fabric 1.1 new features

*Rolling Upgrade Support*  
*Channel Events*  
*Couch DB Indexes*  
*Chaincode - Node.js*  
*Client Application – Common connection profile*  
*Application Level Encryption*  
*TLS*  
*Attribute Based Access Control*  
*Additional features*



# Rolling Upgrade Support



Allows components of the blockchain network to be updated independently

New “**capability requirements**” configuration in the channel determines the feature version level

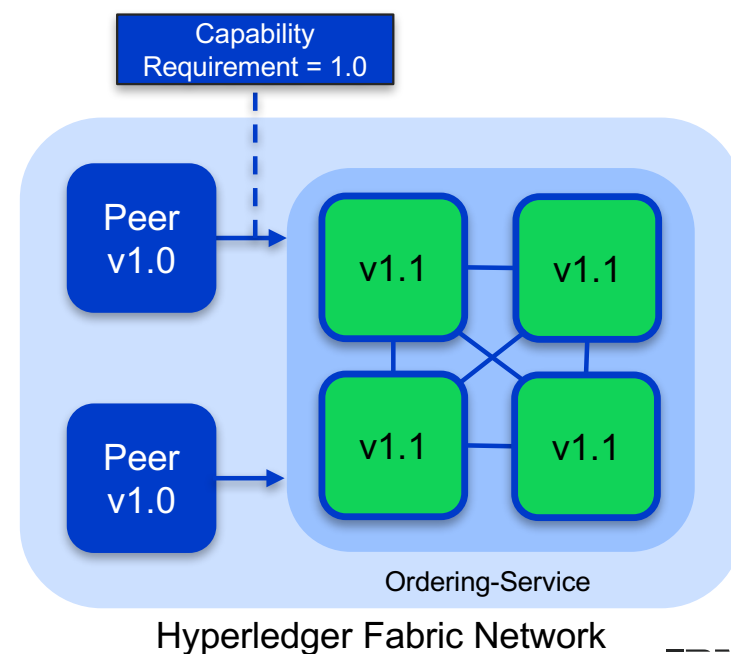
Separately configure:

- **Channel** – Capabilities of orderers and peers defined in the channel group
- **Orderer** – Capabilities of orderers only
- **Application** – Capabilities of peers only

Steps to upgrade a network (can be done in parallel):

- Orderers, Peers, Fabric-CAs
- Client SDK
- Enable 1.1 capability requirement
- Kafka
- Chaincode?

IBM **Blockchain**



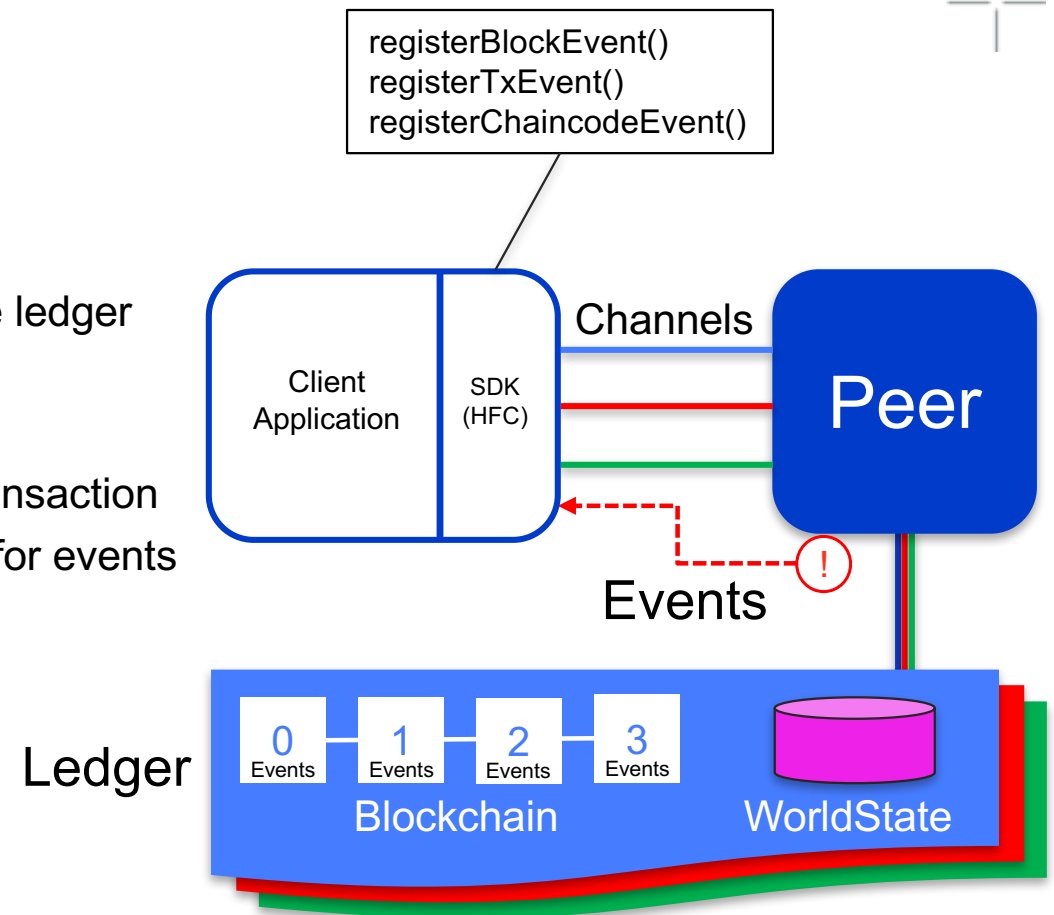
<https://jira.hyperledger.org/browse/FAB-5556>

IBM

# Channel Events

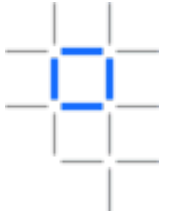
## Peers now deliver events per channel

- Rearchitected in Fabric 1.1
- Events captured during endorsement and stored in the ledger
- Peers can replay past events
- Applications can request old events to catch-up
- Events can include the entire block or be filtered by transaction
- Channel MSP defines which applications can register for events
- Applications register listeners for:
  - Block creation events
  - Transaction events
  - Custom chaincode events





# Couch DB Indexes

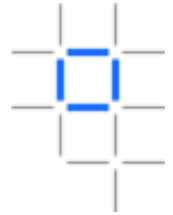


Indexes can be packaged with chaincode to improve query performance

- Indexes packaged alongside chaincode in the following directory:
  - */META-INF/statedb/couchdb/indexes*
- Each index must be defined separately in its own .json file
- When chaincode is first installed, the index is deployed to CouchDB on instantiation
- For subsequent chaincode installs the index is deployed on install
- Once the index is deployed it will automatically be used by CouchDB

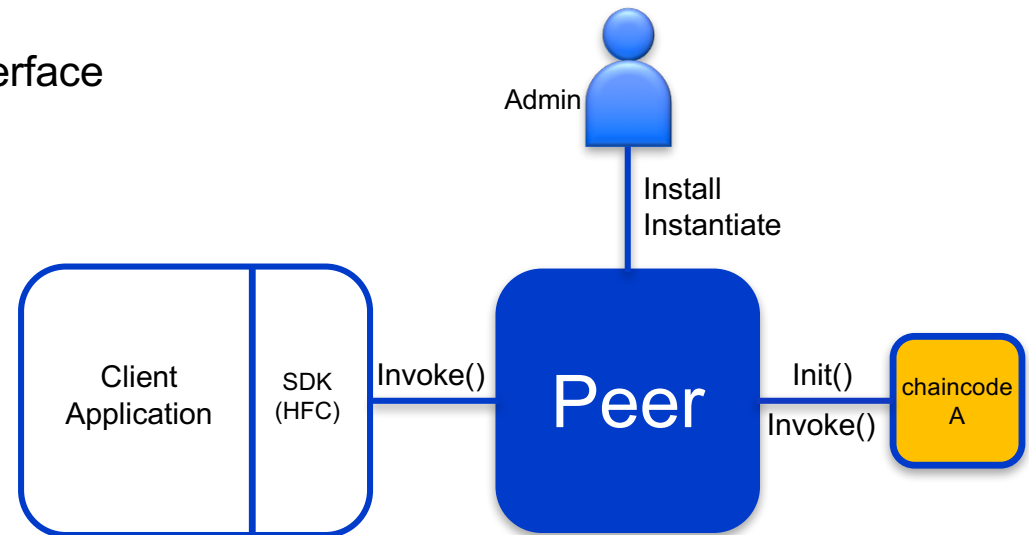
```
{
  "index": {
    "fields": ["docType", "owner"]
  },
  "ddoc": "indexOwnerDoc",
  "name": "indexOwner",
  "type": "json"
}
```

# Chaincode (a.k.a Smart Contract)



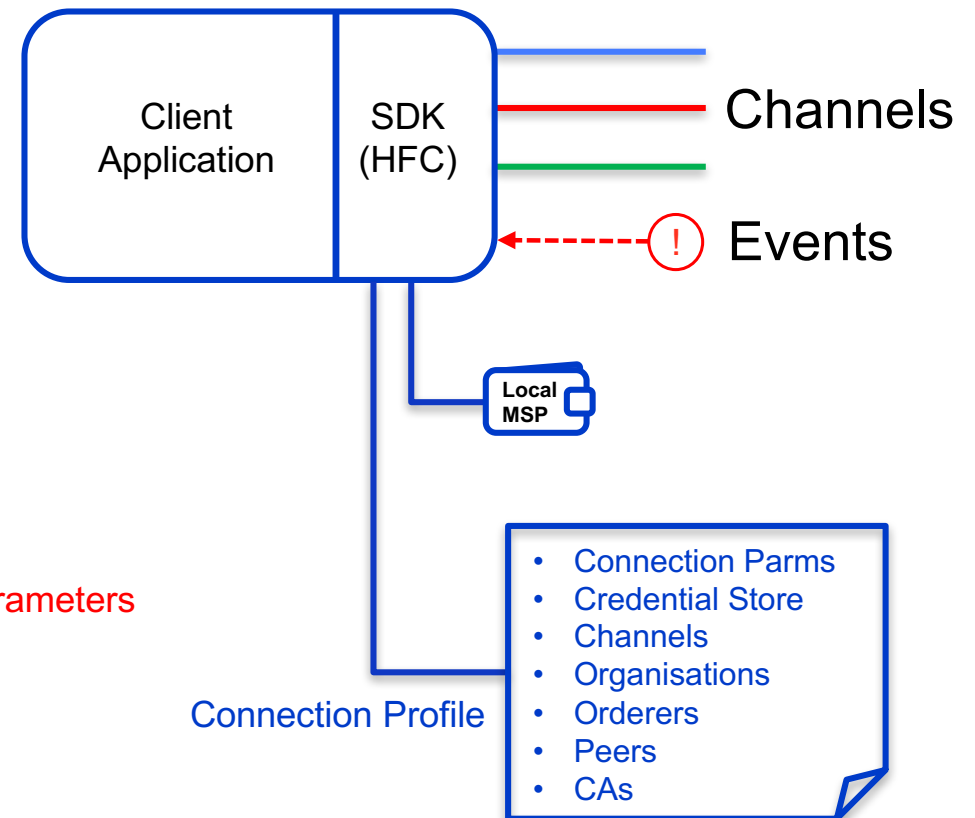
Chaincode contains business logic deployed to peers

- Installed on peers and instantiated on channels
- Run in secured docker images separate to the peer
- Interact with the worldstate through the Fabric shim interface
- Each chaincode has its own scoped worldstate
- Language support for:
  - Golang
  - **Node.js (new in Fabric 1.1)**
  - Java (Future FAB-8063)
- Implements:
  - Init() - Called on instantiate and upgrade
  - Invoke() – Called from client application



# 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
- Supported Languages
  - Node.js
  - Java
  - Golang
  - Python (future)
- **Common Connection Profile (New in Fabric 1.1)**
  - Includes all blockchain network end-points and connection parameters
  - Simplifies coding in the Fabric-SDK
  - Used to create a Business Network Card in Composer
  - Can be coded in yaml or JSON

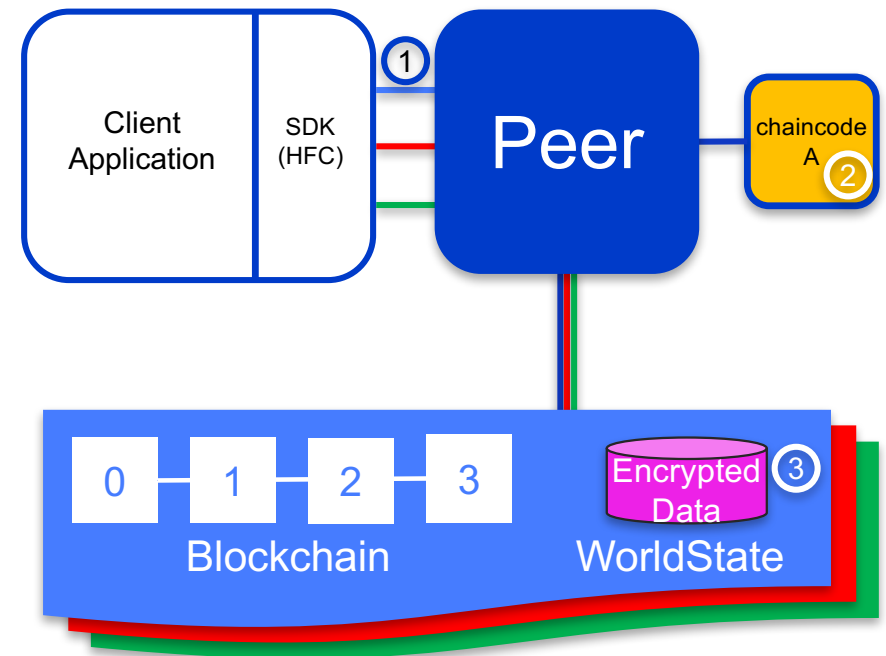


# Application Level Encryption

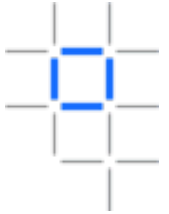
## Encrypt/Decrypt and sign data in chaincode

- Fabric includes an encryption library for use by chaincode
- New chaincode “entities” API provides interface to:
  - Encrypt (AES 256 symmetric or asymmetric)
  - Decrypt
  - Sign (ECDSA)
  - Verify
- Pass cryptographic key(s) to chaincode via **transient-data** field
- Support for Initialisation Vector (IV) allowing multiple endorsers to calculate same cypher text

1. Pass unencrypted data and keys to endorser
2. Chaincode encrypts data to put in worldstate
3. Encrypted data stored in worldstate

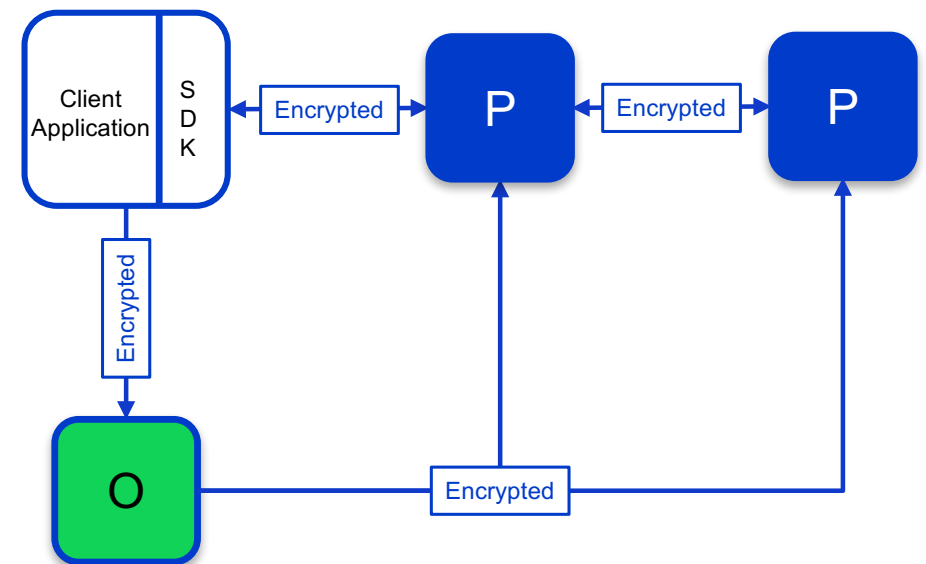


# Transport Layer Security

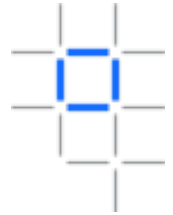


All communications within a Hyperledger Fabric network can be secured using TLS

- Peers and Orderers are both TLS Servers and TLS Clients
- Applications and commands are TLS Clients
- Fabric 1.0.x support for TLS Server authentication
- **Fabric 1.1 supports mutual TLS (client authentication)**



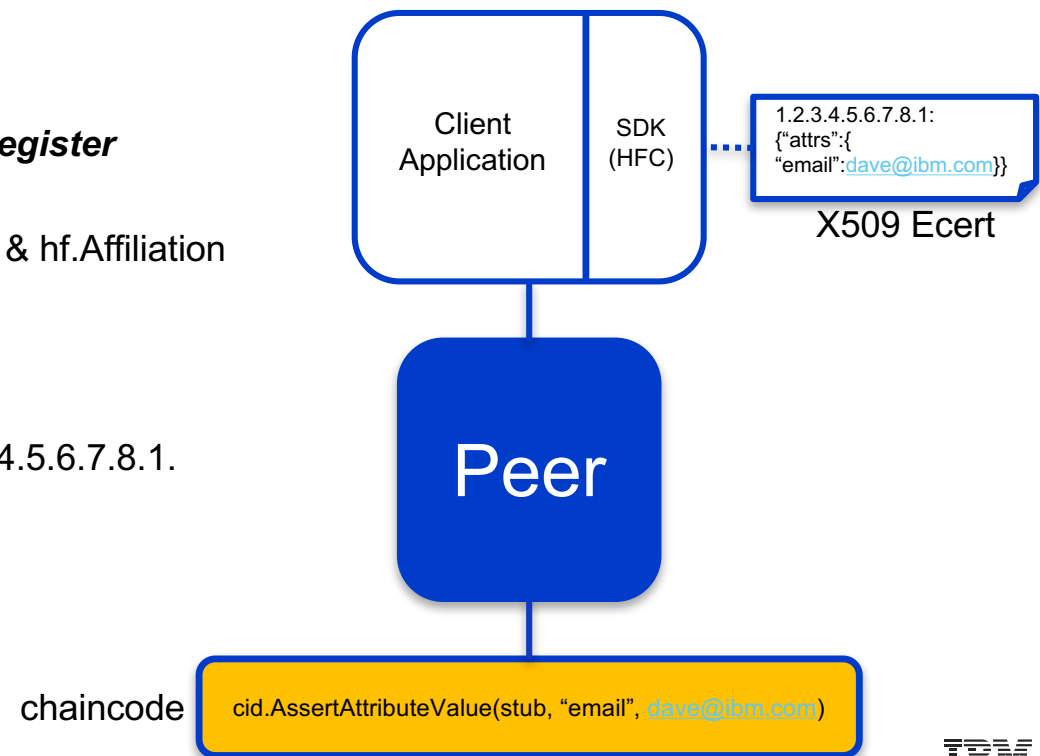
# Attribute Based Access Control



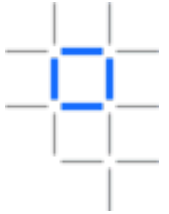
## Include identity attributes in enrollment certificates for chaincode

- Include attributes in **X509 enrollment certificates** (Ecerts)
- Defined as name/value pairs: email=dave@ibm.com
- Define mandatory and optional attributes with ***fabric-ca-client register***
- Specify attribute values with ***fabric-ca-client enroll***
- Ecerts automatically include attributes: hf.EnrollmentID, hf.Type & hf.Affiliation
- API provided by Client Identity chaincode Library:
  - cid.GetAttributeValue(stub, "attr1")
  - cid.AssertAttributeValue(stub, "myapp.admin", "true")
- Stored as an extension in the Ecert with an ASN.1 OID of 1.2.3.4.5.6.7.8.1.

```
1.2.3.4.5.6.7.8.1:  
{"attrs":{"attr1":"val1"}}
```



# Additional new features in Fabric 1.1



## **Generate a Certificate Revocation List (CRL) from Fabric CA server**

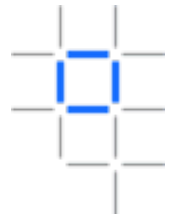
- fabric-ca-client support for `-gencrl` option outputs all revoked certs to a .pem file
- Support for both revoked and expired timeframes
- .pem file can be placed in local MSP and channel config blocks
- <https://jira.hyperledger.org/browse/FAB-5300>

## **Dynamic update of identities**

- Identities within Fabric-CA have the following fields: ID, Secret, Affiliation, Type, Maxenrollments, Attributes.
- Fabric 1.1 supports updating these fields, as well as creating and removing identities without a Fabric-CA server restart
- <https://jira.hyperledger.org/browse/FAB-5726>

## **Performance and Scale Improvements**

- Improvements in CouchDB (indexes), Orderer optimisations, Peer asynchronous updates to the ledger, Cache MSP identity validations.
- <https://jira.hyperledger.org/browse/FAB-6421>



# Further Information

Rolling Upgrade via configured capabilities - Support nodes of mixed versions in Fabric networks

- [http://hyperledger-fabric.readthedocs.io/en/release-1.1/upgrade\\_to\\_one\\_point\\_one.html](http://hyperledger-fabric.readthedocs.io/en/release-1.1/upgrade_to_one_point_one.html)
- [http://hyperledger-fabric.readthedocs.io/en/release-1.1/upgrading\\_your\\_network\\_tutorial.html](http://hyperledger-fabric.readthedocs.io/en/release-1.1/upgrading_your_network_tutorial.html)
- <http://hyperledger-fabric-ca.readthedocs.io/en/release-1.1/users-guide.html#upgrading-the-server>

Channel-based event service for blocks and block transaction events

- [http://hyperledger-fabric.readthedocs.io/en/release-1.1/peer\\_event\\_services.html](http://hyperledger-fabric.readthedocs.io/en/release-1.1/peer_event_services.html)
- <https://fabric-sdk-node.github.io/tutorial-channel-events.html>

Package CouchDB indexes with chaincode to enable efficient queries of ledger state

- [http://hyperledger-fabric.readthedocs.io/en/release-1.1/couchdb\\_as\\_state\\_database.html](http://hyperledger-fabric.readthedocs.io/en/release-1.1/couchdb_as_state_database.html)

Generate a Certificate Revocation List from Fabric CA

- <http://hyperledger-fabric-ca.readthedocs.io/en/release-1.1/users-guide.html>

Dynamically update Fabric CA Identities and Affiliations

- <http://hyperledger-fabric-ca.readthedocs.io/en/release-1.1/users-guide.html#dynamic-server-configuration-update>

Node.js Chaincode Support - the Hyperledger Fabric tutorials can be run with either Go chaincode or Node.js chaincode

- [http://hyperledger-fabric.readthedocs.io/en/release-1.1/build\\_network.html](http://hyperledger-fabric.readthedocs.io/en/release-1.1/build_network.html)
- [http://hyperledger-fabric.readthedocs.io/en/release-1.1/write\\_first\\_app.html](http://hyperledger-fabric.readthedocs.io/en/release-1.1/write_first_app.html)

Node.js SDK Connection Profile to simplify connections to Fabric nodes

- <https://fabric-sdk-node.github.io/tutorial-network-config.html>

Mutual TLS between Fabric nodes, and between clients and nodes

- [http://hyperledger-fabric.readthedocs.io/en/release-1.1/enable\\_tls.html](http://hyperledger-fabric.readthedocs.io/en/release-1.1/enable_tls.html)

Encrypt ledger data for confidentiality using the chaincode encryption library

- [https://github.com/hyperledger/fabric/tree/master/examples/chaincode/go/enccc\\_example](https://github.com/hyperledger/fabric/tree/master/examples/chaincode/go/enccc_example)
- [https://github.com/hyperledger/fabric/blob/master/examples/chaincode/go/enccc\\_example/utlis.go](https://github.com/hyperledger/fabric/blob/master/examples/chaincode/go/enccc_example/utlis.go)

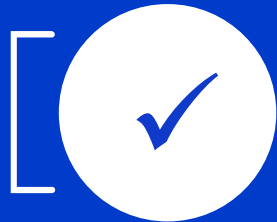
Attribute-based Access Control in chaincode

- <http://hyperledger-fabric-ca.readthedocs.io/en/release-1.1/users-guide.html#attribute-based-access-control>
- <https://github.com/hyperledger/fabric/tree/master/core/chaincode/lib/cid/>

Chaincode APIs to retrieve client identity for access control decisions

- <https://github.com/hyperledger/fabric/tree/master/core/chaincode/lib/cid/>



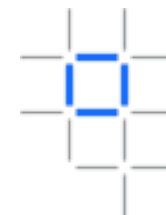


## Fabric 1.1 experimental features

*Private Data Collections*



# Experimental features in Fabric 1.1



Experimental features are enabled by recompiling Fabric 1.1 source

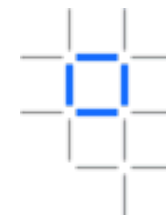
- export EXPERIMENTAL=true

It is expected that experimental features will GA in Fabric 1.2

Experimental features include:

- Private Data Channels
  - <https://jira.hyperledger.org/browse/FAB-1151>
- Java Chaincode
  - <https://jira.hyperledger.org/browse/FAB-1973>
- Identity Mixer to support unlinkability for signing transactions
  - <https://jira.hyperledger.org/browse/FAB-2005>
- Finer grained channel access control
  - <https://jira.hyperledger.org/browse/FAB-3621>

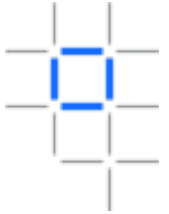
# Private Data Collections



Allows data to be private to only a set of authorized peers

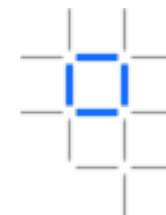
Fabric 1.0		Fabric 1.1
<ul style="list-style-type: none"><li>• Data privacy across channels only</li></ul>	→	<ul style="list-style-type: none"><li>• Data privacy within a channel</li></ul>
<ul style="list-style-type: none"><li>• Transaction proposal and worldstate read/write sets visible to all peers connected to a channel</li></ul>	→	<ul style="list-style-type: none"><li>• Transaction proposal and worldstate read/write sets available to only permissioned peers</li></ul>
<ul style="list-style-type: none"><li>• Ordering service has access to transactions including the read/write sets</li></ul>	→	<ul style="list-style-type: none"><li>• Ordering service has only evidence of transactions (hashes)</li></ul>
		<ul style="list-style-type: none"><li>• Complements Fabric 1.0 channel architecture</li><li>• Policy defines which peers have private data</li></ul>

# Private Data Collections - Explained



1. Private data:
  1. Excluded from transactions by being sent as 'transient data' to endorsing peers.
  2. Shared peer-to-peer with only peers defined in the collection policy.
2. Hashes of private data included in transaction proposal for evidence and validation.
  1. Peers/Orderers not in the collection policy have only hashes.
3. Peers maintain both a public worldstate and private worldstate.
4. Private data held in a transient store between endorsement and validation.

# Private Data Collections – Marble Scenario



## Privacy Requirements:

- No marble data should go through ordering service as part of a transaction
- All peers have access to general marble information
  - *Name, Size, Color, Owner*
- Only a subset of peers have access to marble *pricing* information

### Transaction

- Primary read/write set (if exists)
- Hashed private read/write set (hashed keys/values)

### Transaction

- Public channel data
- Goes to all orderers/peers

### Collection: Marbles

- Private Write Set
  - Name, Size, Color, Owner
- Policy: Org1, Org2**  
"requiredPeerCount": 1,  
"maxPeerCount": 2,  
"blockToLive": 1000000

### Collection: Marbles

- Private data for channel peers
- Goes to all peers but not orderers

### Collection: Marble Private Details

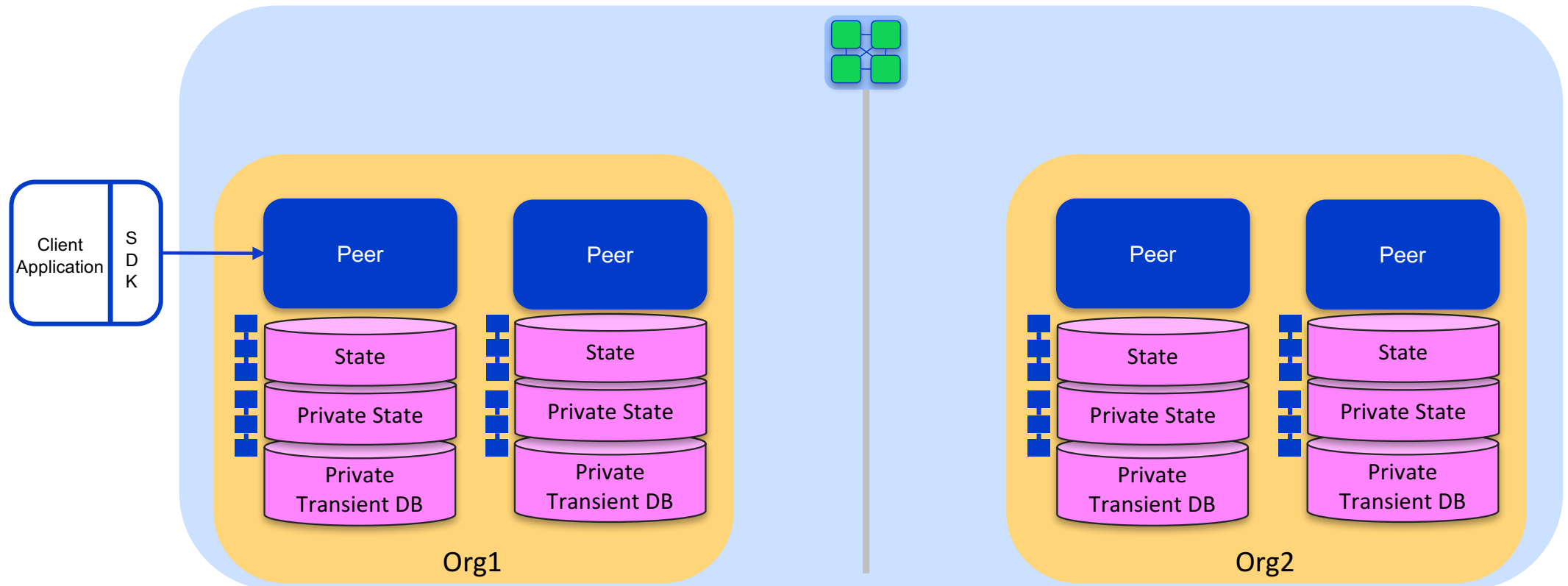
- Private Write Set
  - Price
- Policy: Org1**  
"requiredPeerCount": 1,  
"maxPeerCount": 1,  
"blockToLive": 3

### Collection: Marbles Private Details

- Private data for subset of channel peers
- Goes to subset of peers only

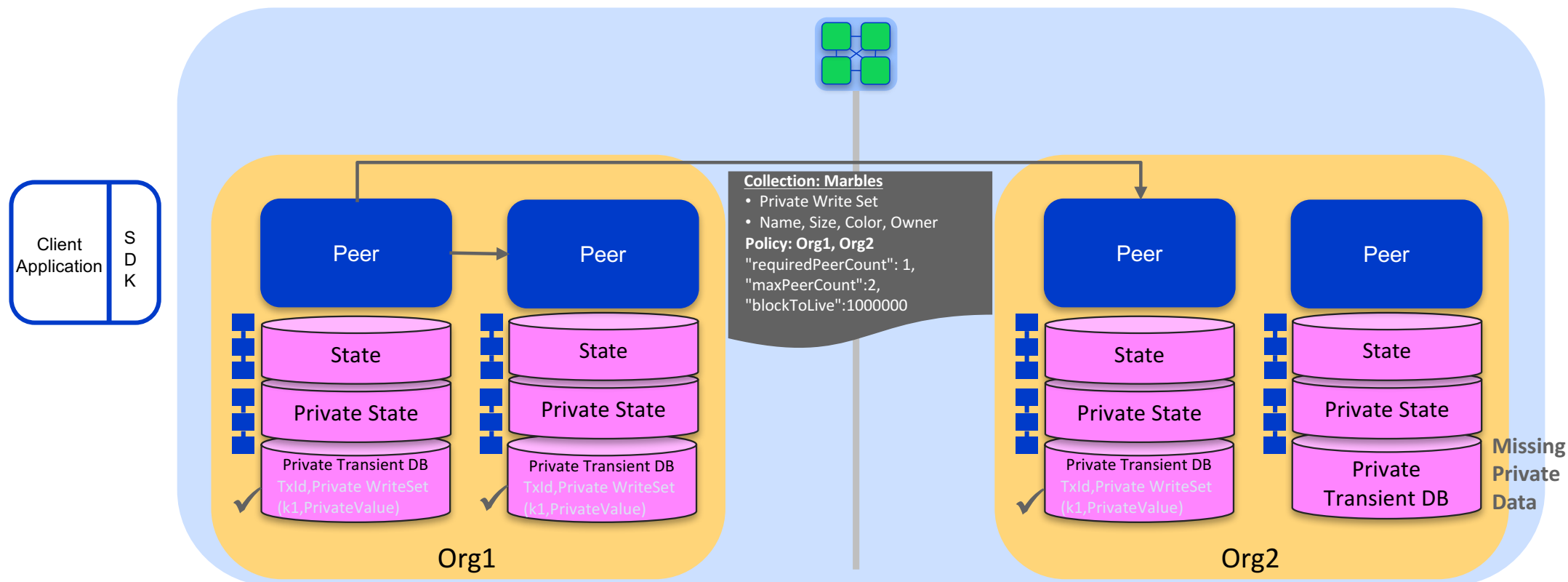
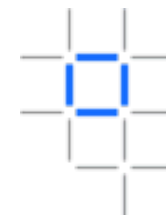
# Step 1: Propose Transaction

Client sends proposal to endorsing peer(s)



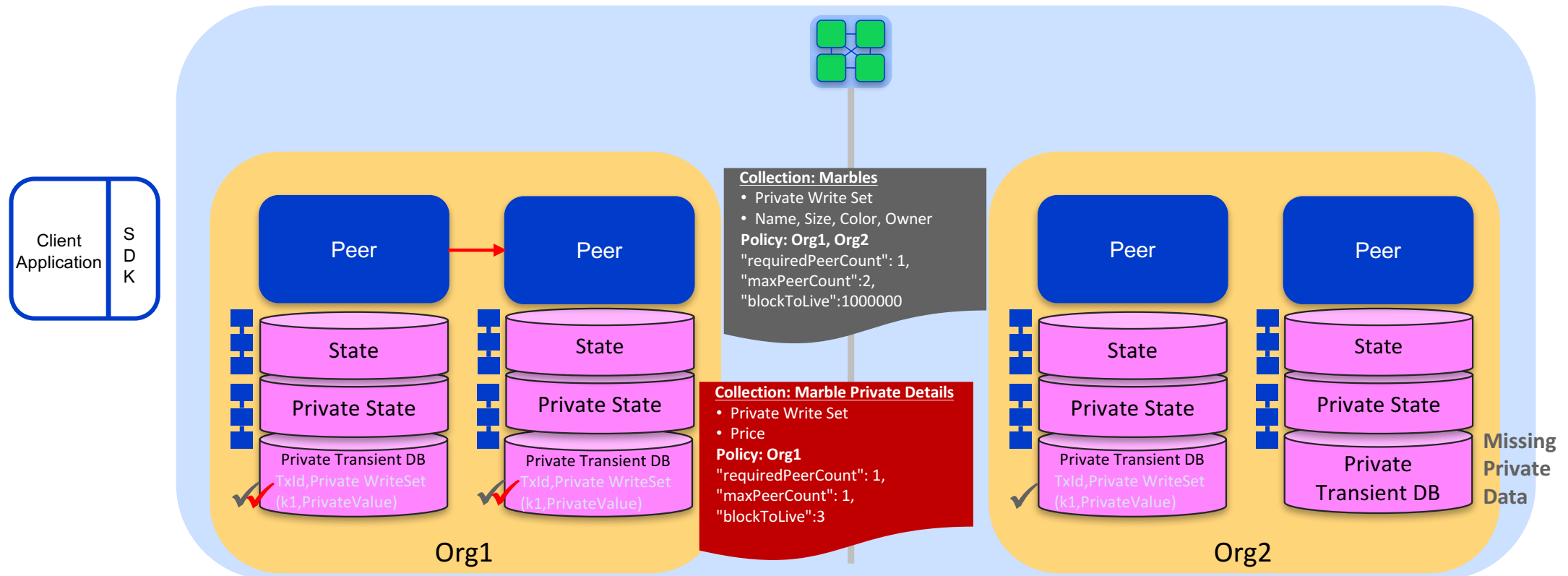
## Step 2a: Execute Proposal and Distribute 1<sup>st</sup> Collection

Endorsing peer simulates transaction and distributes **marbles collection** data based on policy



## Step 2b: Distribute 2<sup>nd</sup> Collection

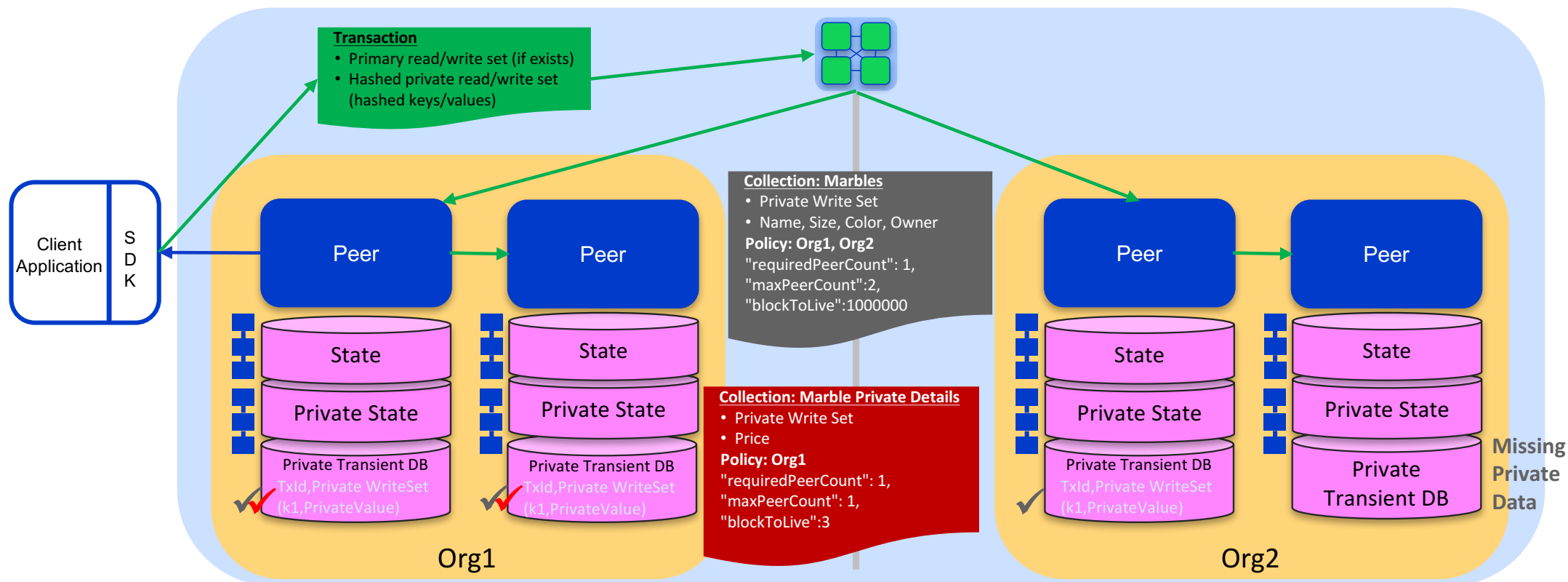
Endorsing peer distributes **marbles private details collection** data based on policy





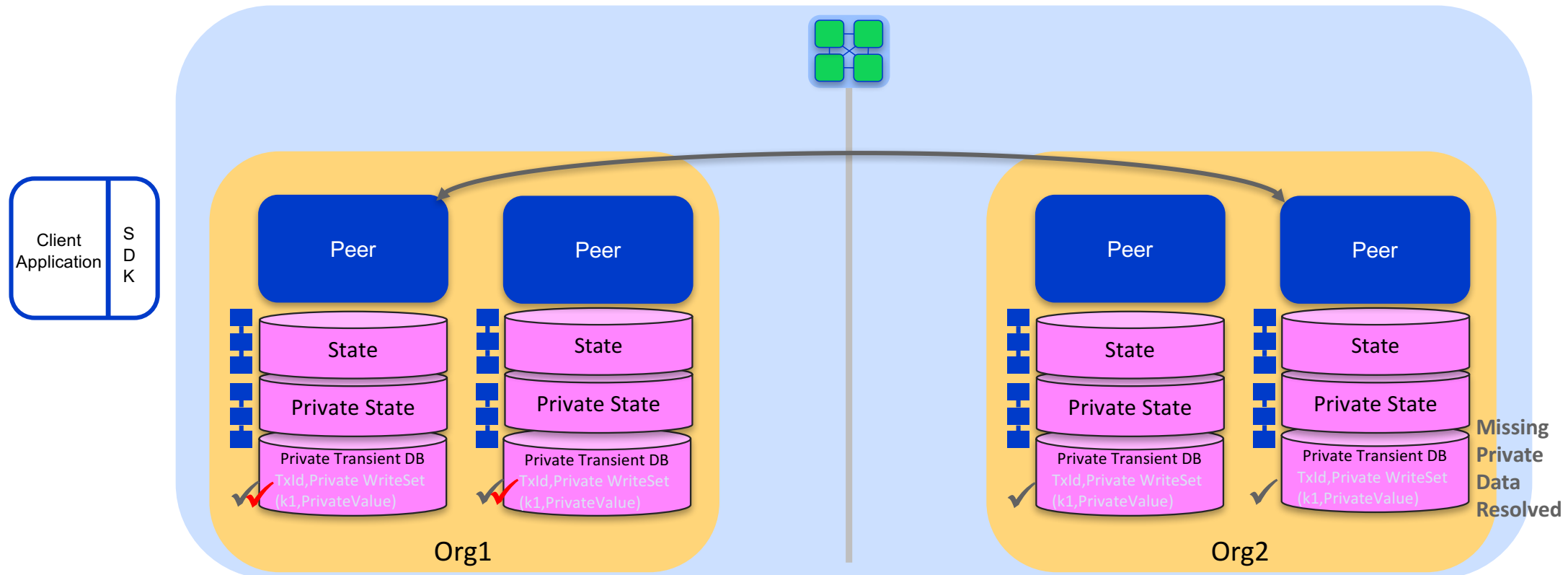
## Step 3: Proposal Response / Order / Deliver

Proposal response sent back to client, which then sends the proposal to the ordering service for delivery to all peers



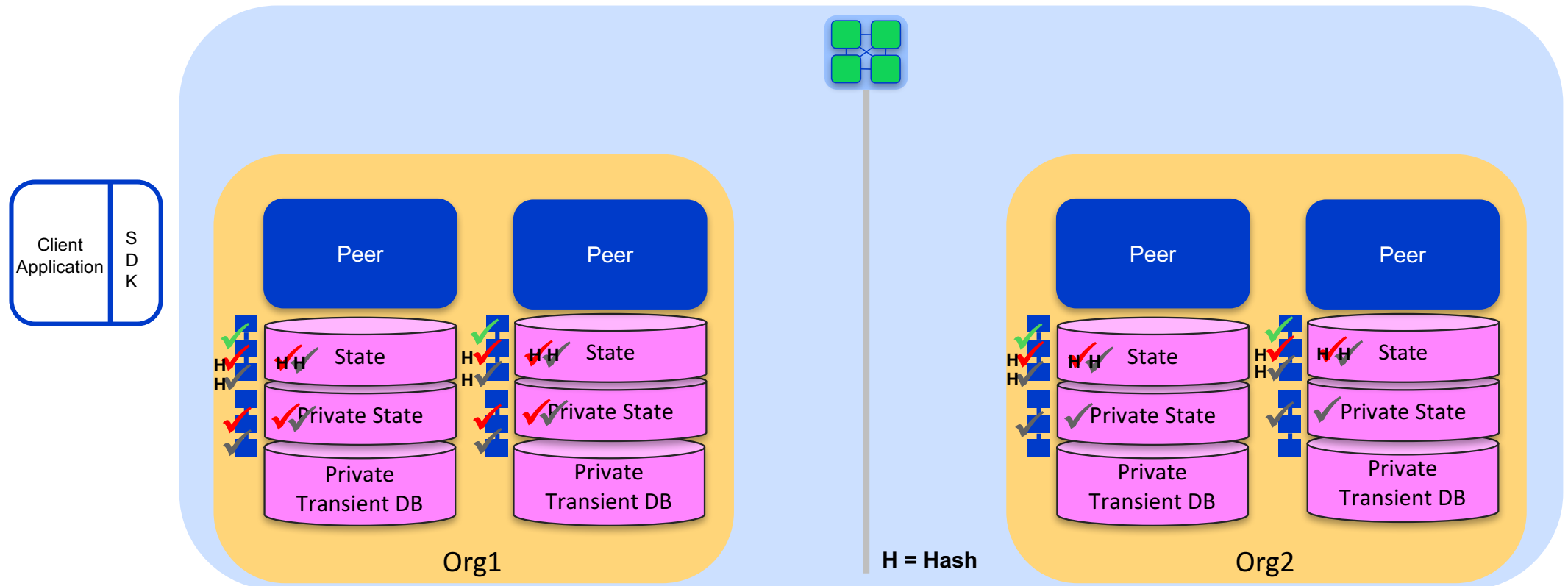
## Step 4: Validate Transaction

Peers validate transactions. Private data validated against hashes. Missing private data resolved with pull requests from other peers.



## Step 5: Commit

- 1) Commit private data to private state db.
- 2) Commit hashes to public state db.
- 3) Commit public block and private write set storage.
- 4) Delete transient data.



# Thank you

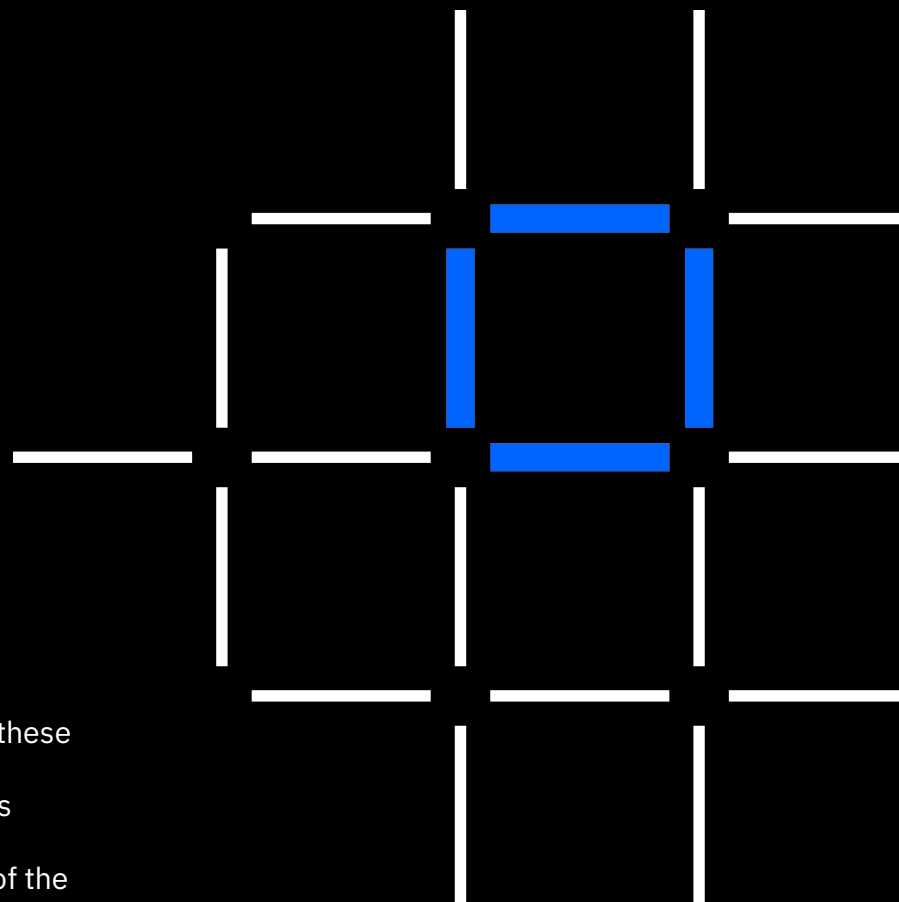
## IBM Blockchain

[www.ibm.com/blockchain](http://www.ibm.com/blockchain)

[developer.ibm.com/blockchain](http://developer.ibm.com/blockchain)

[www.hyperledger.org](http://www.hyperledger.org)

© 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.



The IBM logo, consisting of the letters "IBM" in a bold, sans-serif font, with horizontal stripes integrated into the letters.

