What's New

A Technical Deep-Dive of new features in Hyperledger Fabric V1.1 and V1.2

Barry Silliman
IBM Z Blockchain Enablement
Washington Systems Center
silliman@us.ibm.com

V1.2, 22 Aug 2018

IBM **Blockchain**

Hyperledger Fabric
Roadmap and Architecture

v1.1 Fabric 1.1 new features

v1.2 Fabric 1.2 new features

HYPERLEDGER FABRIC

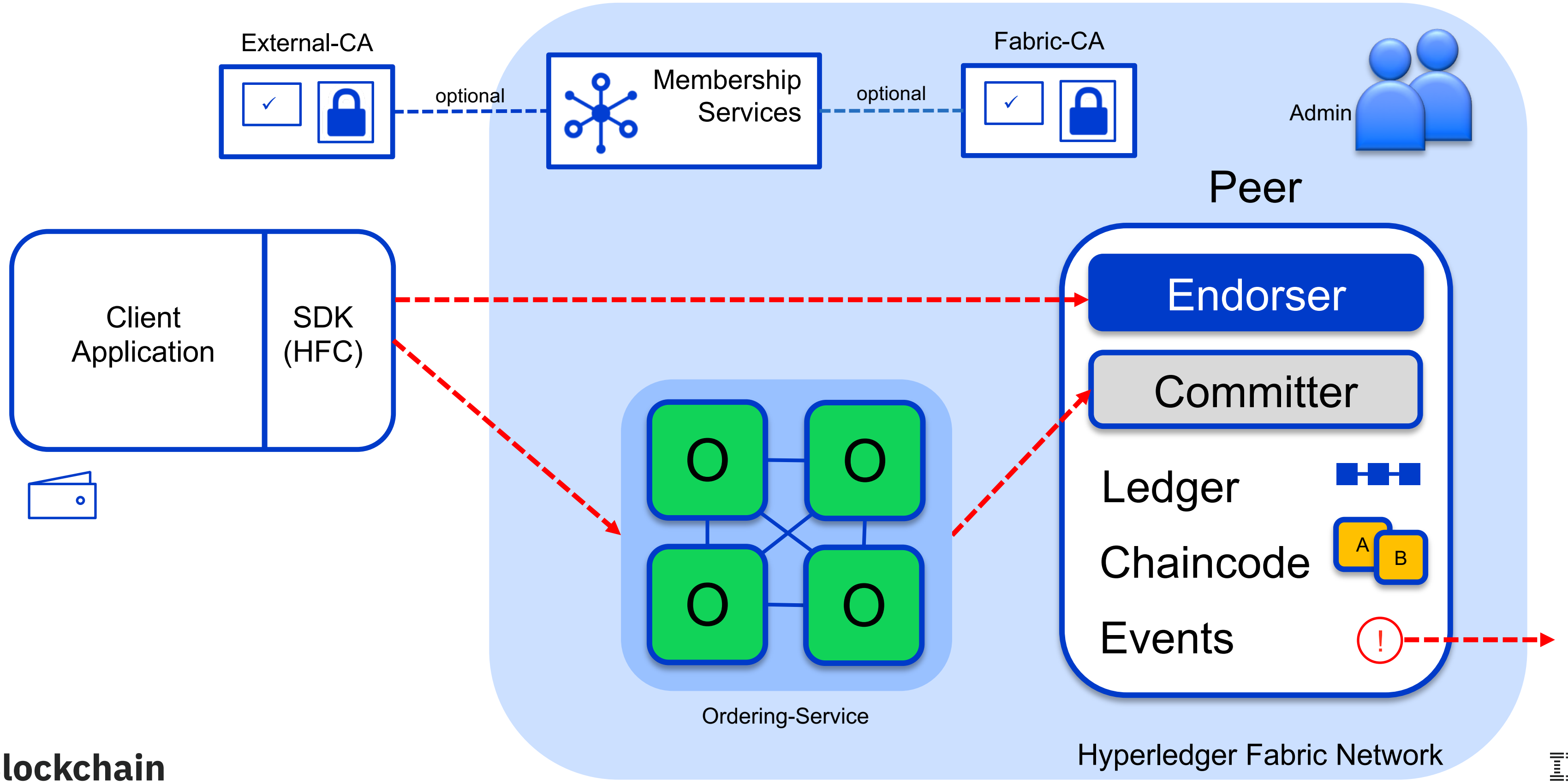
Roadmap

v1 GA	v1.1	v1.2	v1.3
▪ Docker images ▪ Tooling to bootstrap network ▪ Fabric CA or bring your own ▪ Channels for privacy ▪ Cross Channel Query ▪ Java and Node.js SDKs ▪ Ordering Services - Solo or Kafka ▪ Endorsement policy ▪ Level DB and Couch DB ▪ Block dissemination via Gossip ▪ Chaincode ACL ▪ Chaincode packaging & LCI ▪ HSM support ▪ Config transaction lifecycle ▪ Event security ▪ Peer management APIs	▪ Network administration: - Node.js connection profile ▪ Smart contract: - Node.js smart contracts - Encryption library - Attribute Based Access Control ▪ Performance & scale: - More orderers at scale - Parallel txn validation - CouchDB indexes ▪ Events: - Per channel vs global - Block info minimal events ▪ Membership services: - CSR for secure certificates ▪ Serviceability: - Upgrade from 1.0	▪ Network administration: - ACL mechanism per channel - Service discovery ▪ Consensus: - Pluggable endorsement and validation ▪ Smart Contract: - Private Data Collections (SideDB) ▪ Documentation: - Improved documentation and tutorials ▪ Serviceability: - Improvements and bug fixes	▪ Network administration: - CLI redesign - SDK improvements - Service Discovery remaining items ▪ Consensus: - State based endorsement ▪ Smart Contract: - Java chaincode - Burrow EVM support - Private Data remaining items - Chaincode query result pagination ▪ Membership services: - Identity Mixer ▪ Serviceability: - Improvements and bug fixes
July 2017	March 2018	June 2018	*Sept 2018* (quarterly)*

HYPERLEDGER
BLOCKCHAIN TECHNOLOGIES FOR BUSINESS

Based on https://wiki.hyperledger.org/projects/fabric/roadmap - Dates determined by the Hyperledger community - (*) Subject to change

Hyperledger Fabric V1 Architecture



v1.1

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

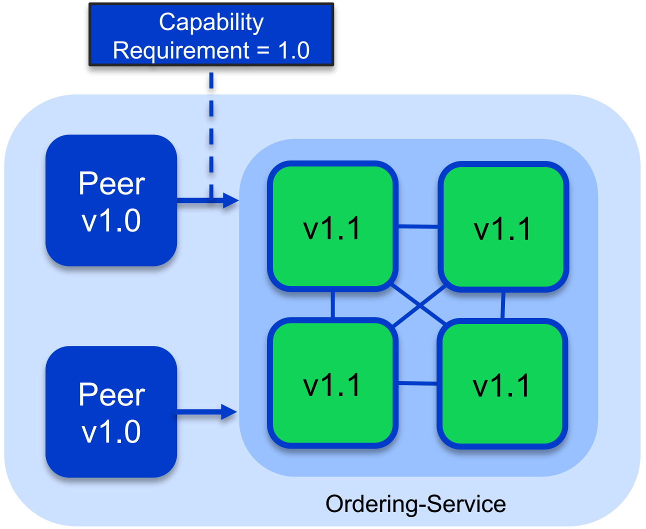
New "**capability requirements**" configuration in the channel determines the feature version level

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?



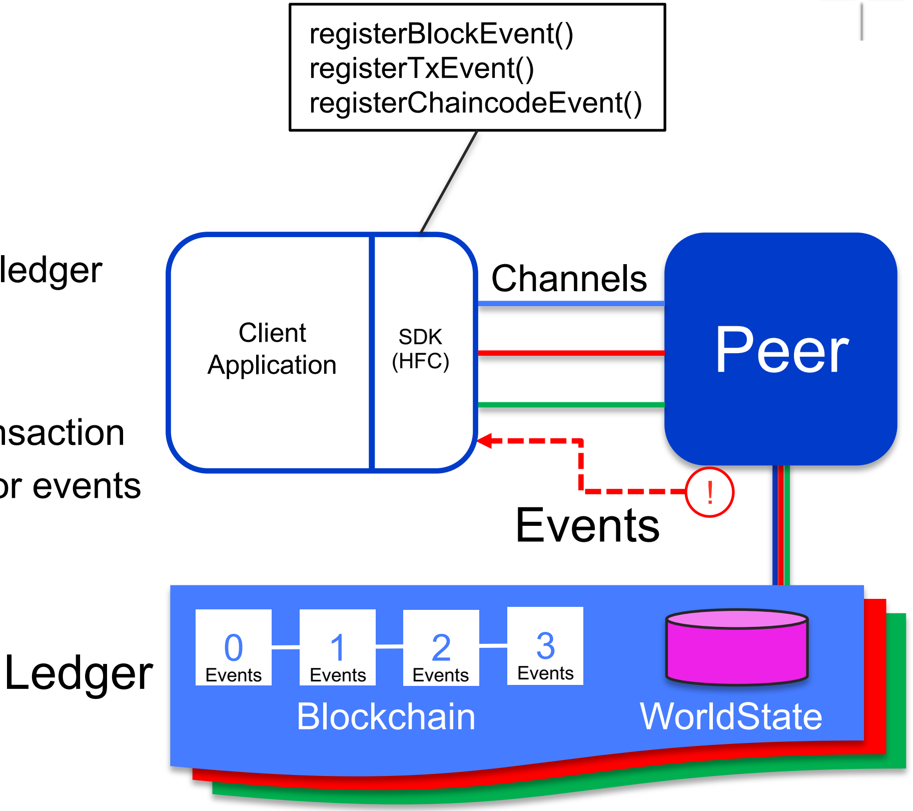
Hyperledger Fabric Network

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

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



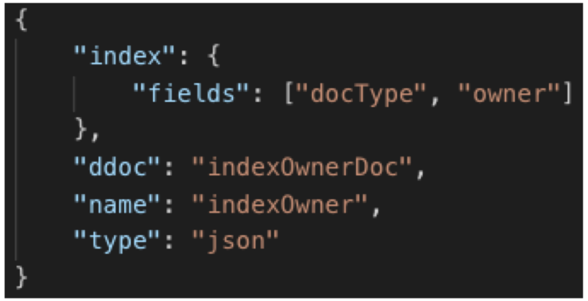
IBM **Blockchain**

https://jira.hyperledger.org/browse/FAB-6911 + https://jira.hyperledger.org/browse/FAB-6911

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
- Once the index is deployed it will automatically be used by CouchDB



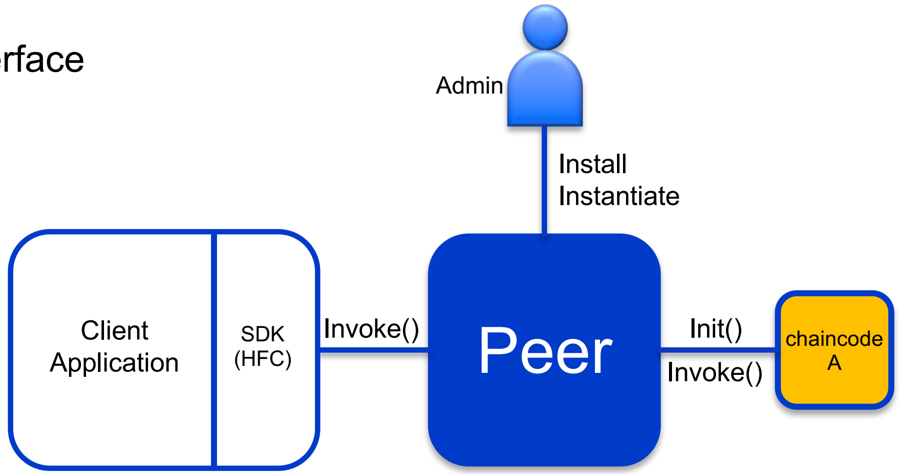
```
{
    "index": {
        "fields": ["docType", "owner"]
    },
    "ddoc": "indexOwnerDoc",
    "name": "indexOwner",
    "type": "json"
}
```

https://jira.hyperledger.org/browse/FAB-3067

Chaincode (a.k.a Smart Contract)

Chaincode contains business logic deployed to peers

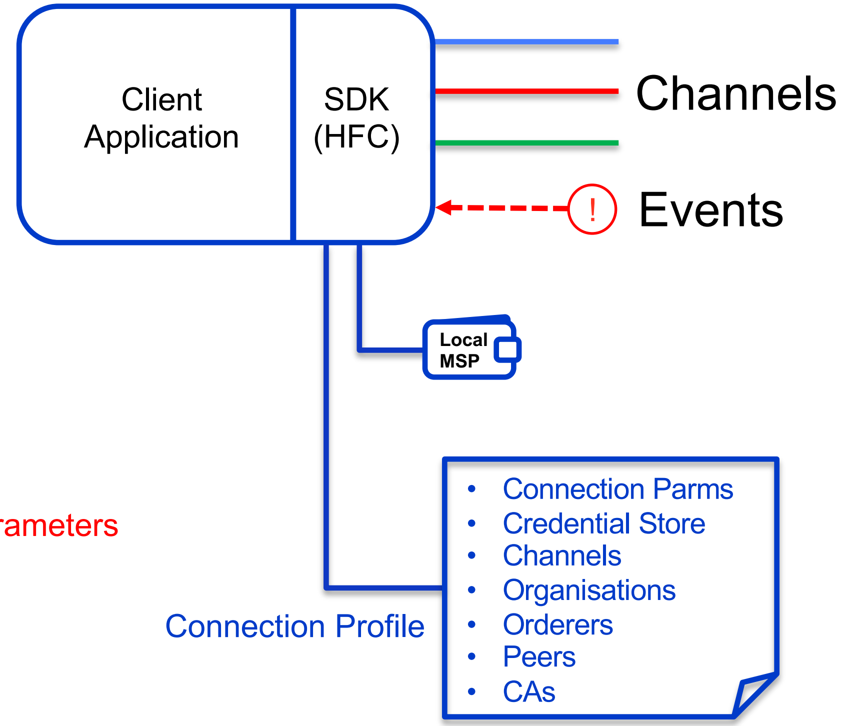
- Installed on peers and instantiated on channels
- Run in secured docker images separate to the peer
- Interact with the worldstate through the Fabric shim interface
- Each chaincode has its own scoped worldstate
- Language support for:
  - Golang
  - Node.js (new in Fabric 1.1)
  - Java (Future FAB-8063)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



https://jira.hyperledger.org/browse/FAB-2331

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



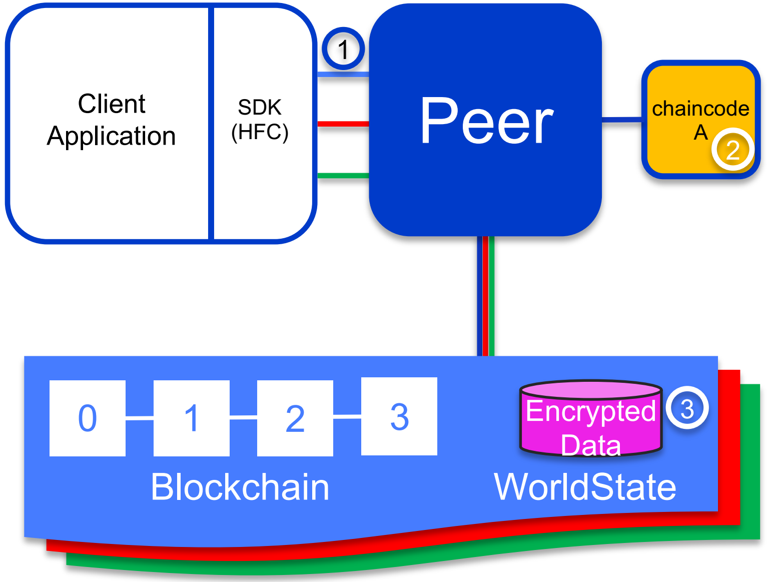
https://jira.hyperledger.org/browse/FAB-5363

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

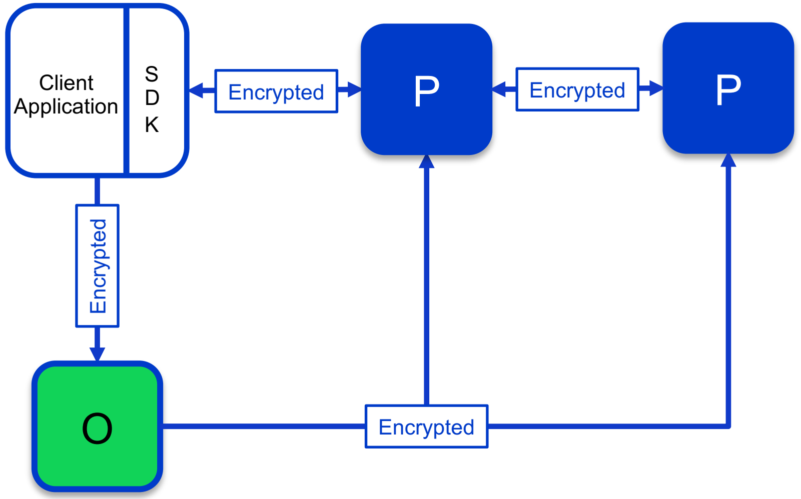


https://jira.hyperledger.org/browse/FAB-830

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)



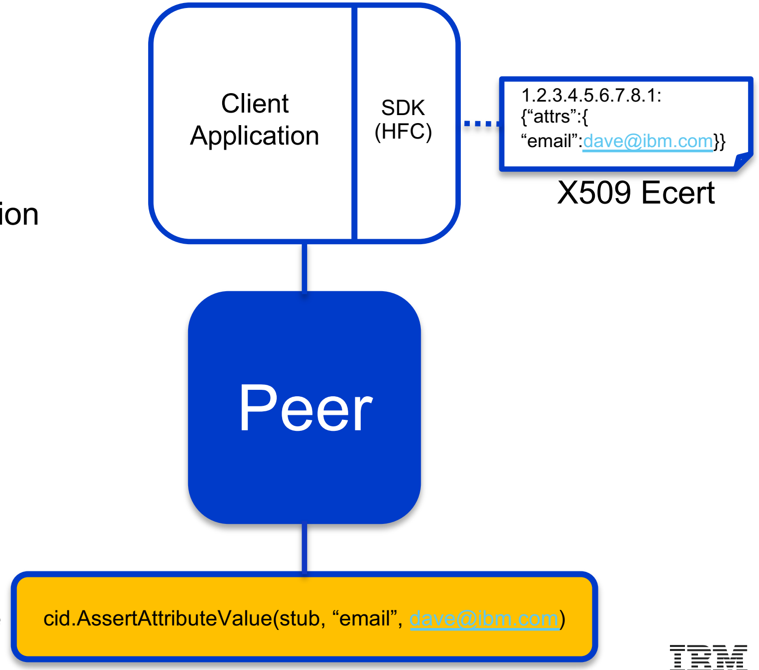
https://jira.hyperledger.org/browse/FAB-6715

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"}}
```



IBM **Blockchain**

https://jira.hyperledger.org/browse/FAB-5346 + https://jira.hyperledger.org/browse/FAB-6089

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/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
- 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

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/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

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/blob/master/examples/chaincode/go/enccc_example/utils.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/

v1.2

Fabric 1.2 new features

Private Data Collections
ACL mechanism per channel
Pluggable endorsement and validation
Service Discovery

Private Data Collections

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

Fabric 1.0		Fabric 1.2
• Data privacy across channels only	→	• Data privacy within a channel
• Transaction proposal and worldstate read/write sets visible to all peers connected to a channel	→	• Transaction proposal and worldstate read/write sets available to only permissioned peers
• Ordering service has access to transactions including the read/write sets	→	• Ordering service has only evidence of transactions (hashes)
		• Complements Fabric 1.0 channel architecture
		• Policy defines which peers have private data

https://jira.hyperledger.org/browse/FAB-8718

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.

https://jira.hyperledger.org/browse/FAB-1151

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

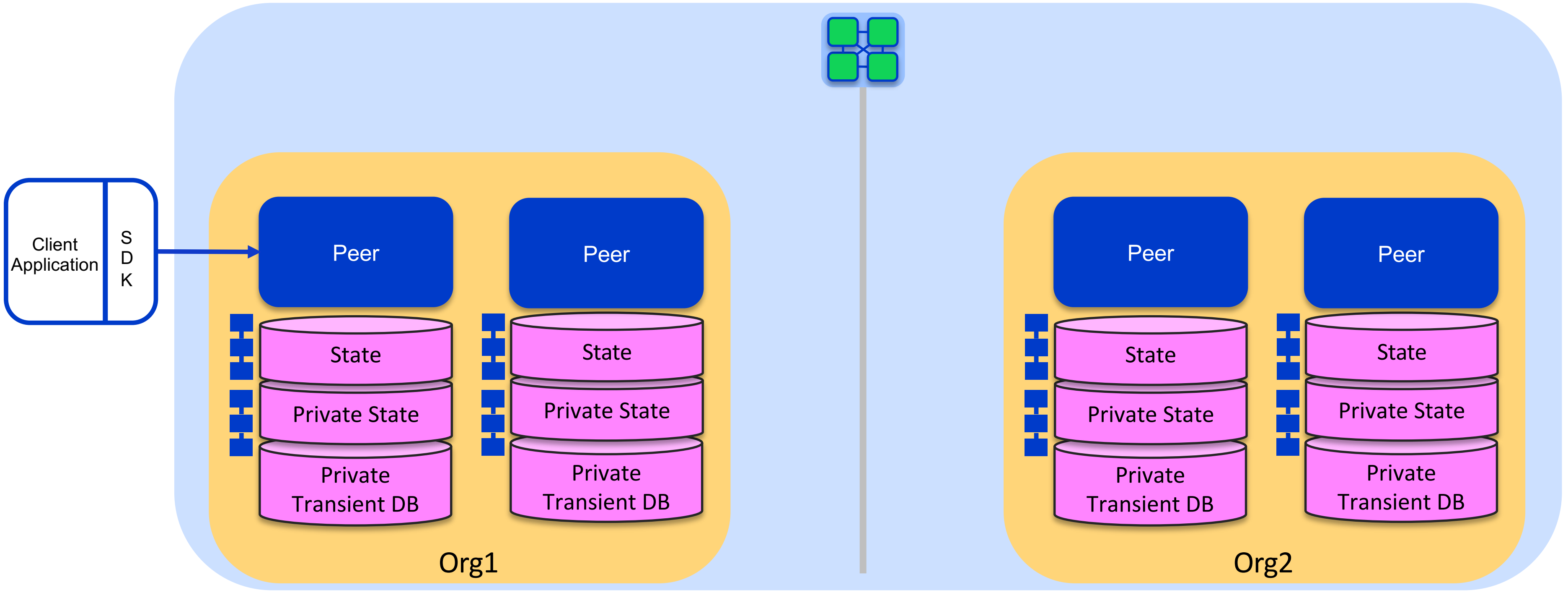
Collection: Marble Private Details
- Private Write Set
- Price

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

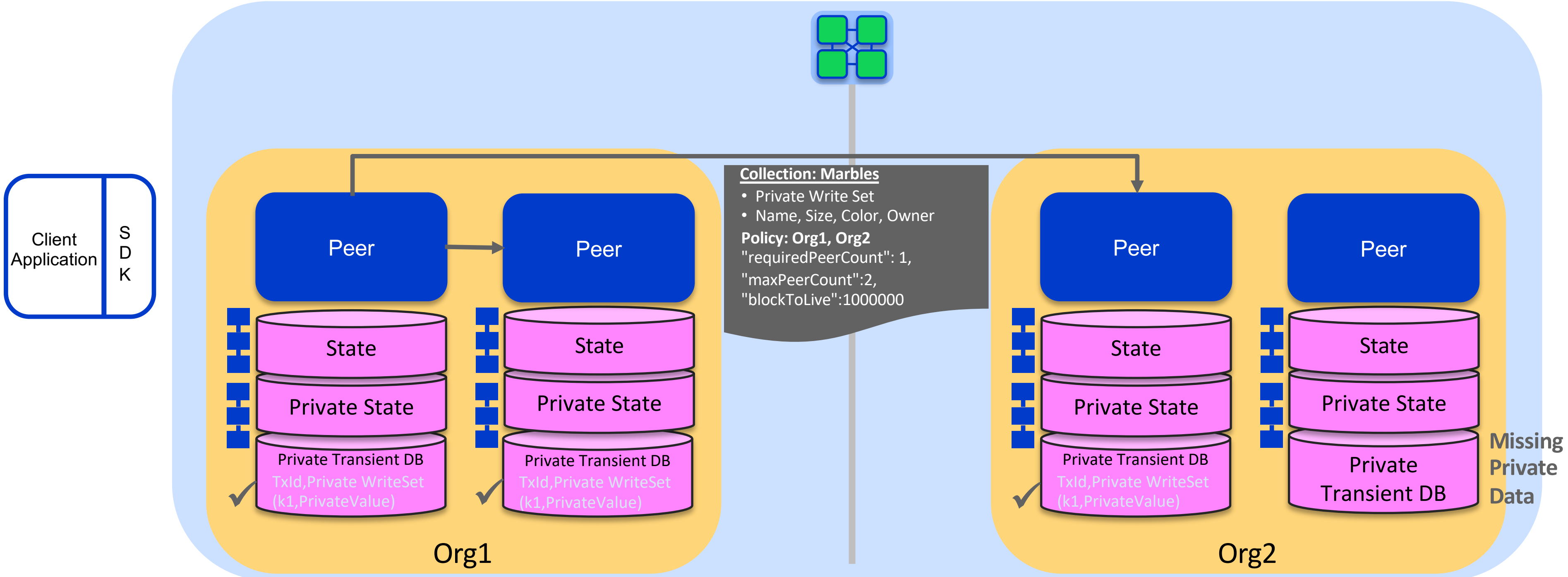
Step 1: Propose Transaction

Client sends proposal to endorsing peer(s)



Step 2a: Execute Proposal and Distribute 1st Collection

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



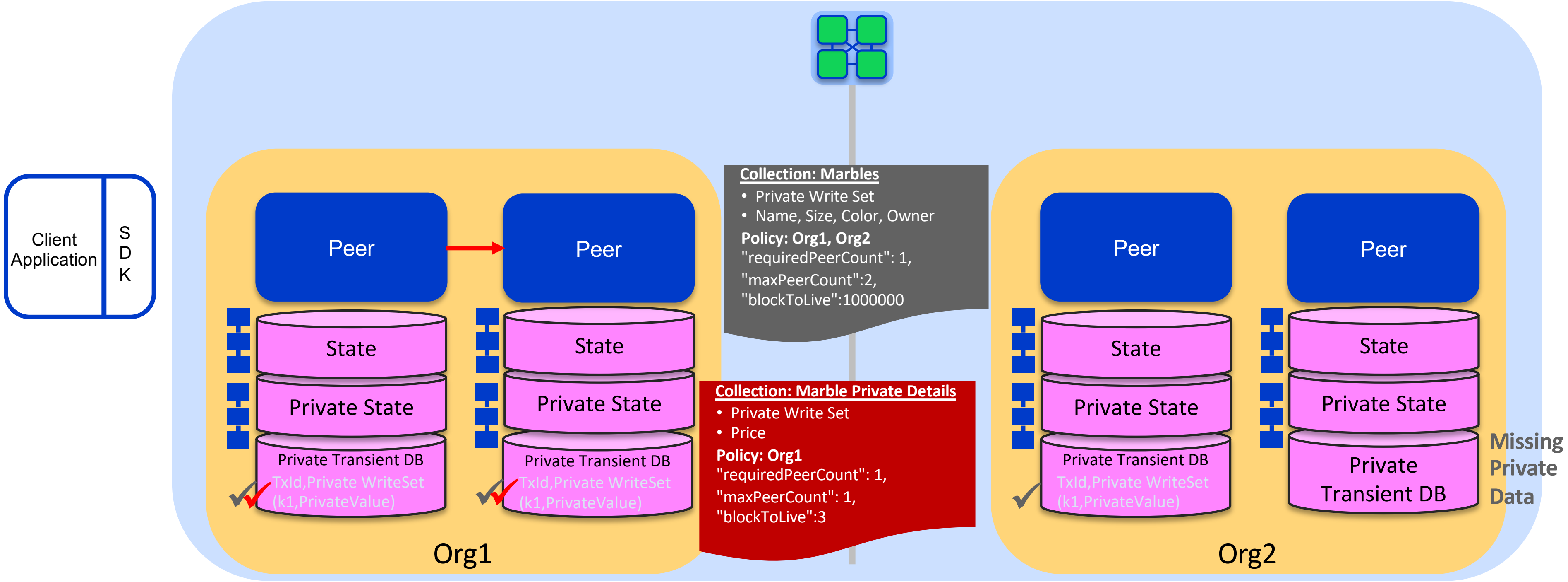
Collection: Marbles

- Private Write Set
- Name, Size, Color, Owner

Policy: Org1, Org2
"requiredPeerCount": 1,
"maxPeerCount":2,
"blockToLive":1000000

Step 2b: Distribute 2nd Collection

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



Collection: Marbles
- Private Write Set
- Name, Size, Color, Owner

Policy: Org1, Org2
"requiredPeerCount": 1,
"maxPeerCount":2,
"blockToLive":1000000

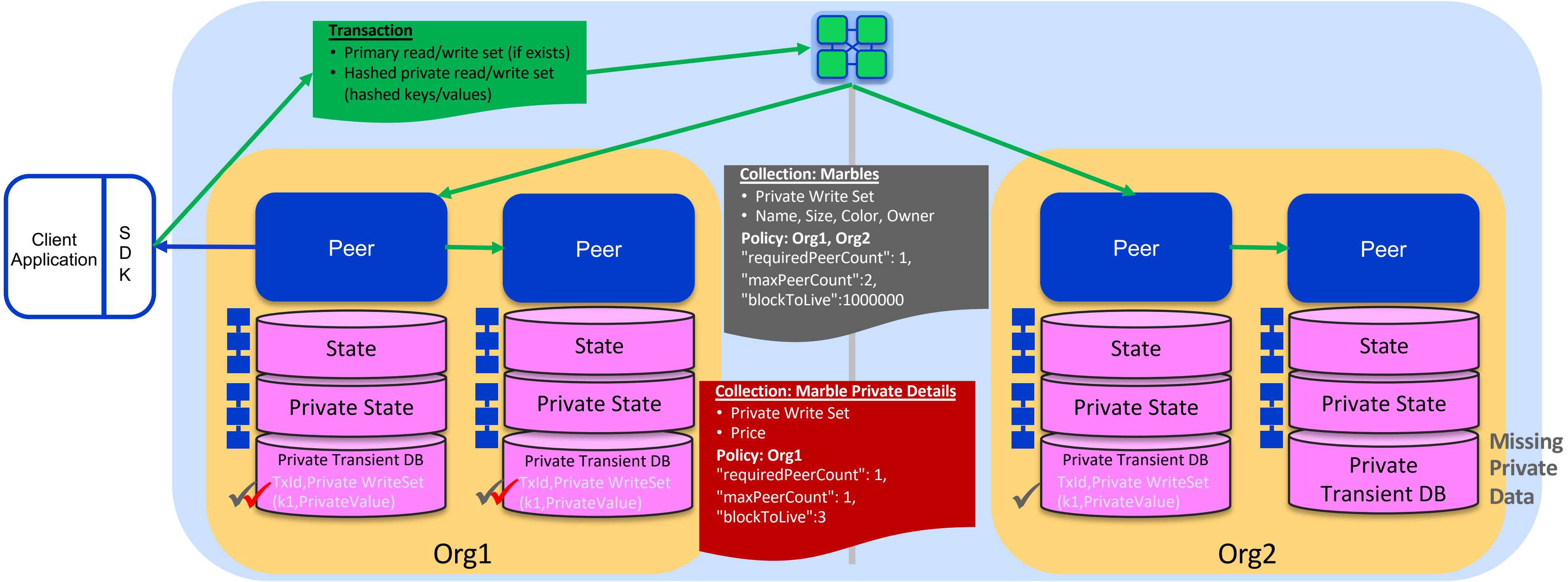
Collection: Marble Private Details
- Private Write Set
- Price

Policy: Org1
"requiredPeerCount": 1,
"maxPeerCount": 1,
"blockToLive":3

IBM **Blockchain**

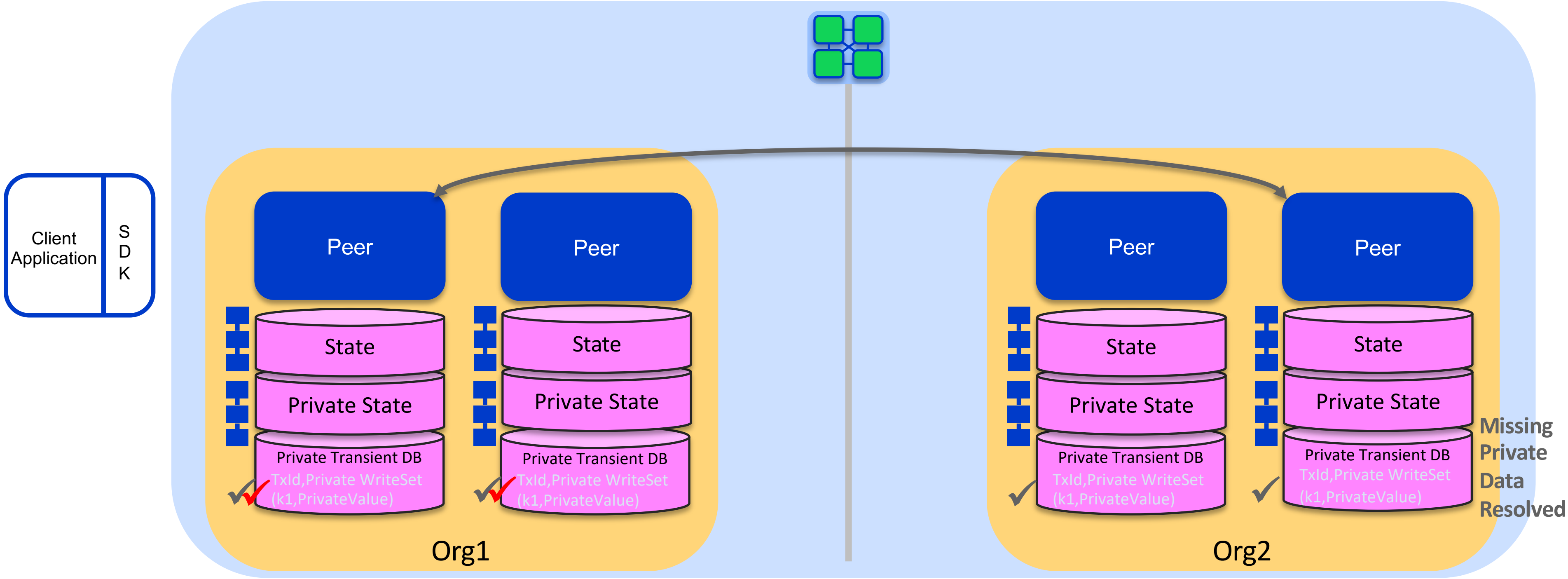
Step 3: Proposal Response / Order / Deliver

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



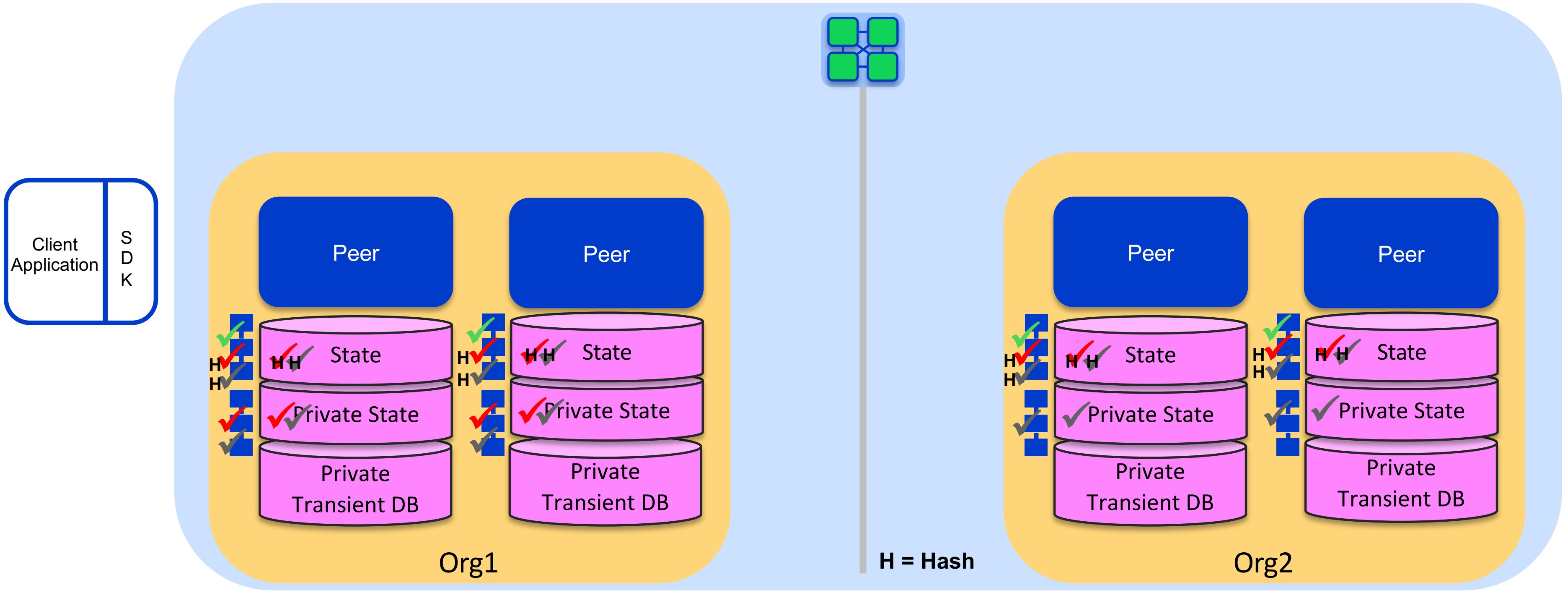
Step 4: Validate Transaction

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.

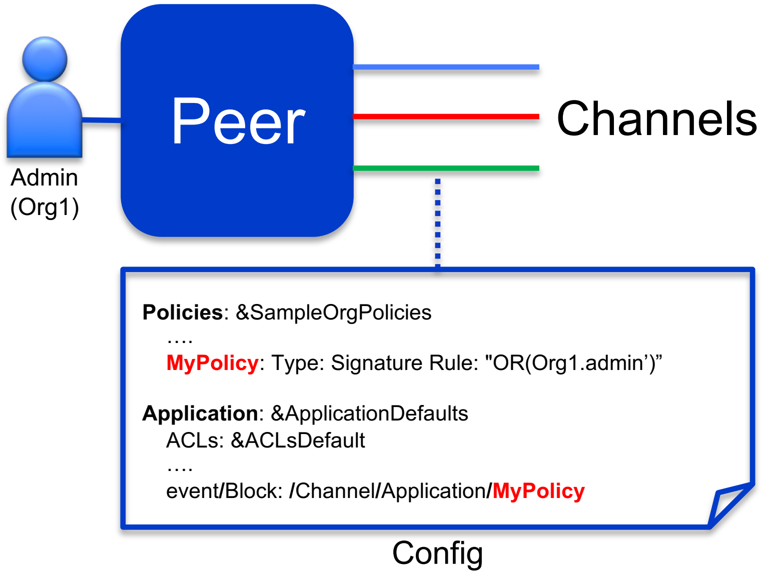


IBM **Blockchain**

ACL mechanism per channel

Support policy based access control for peer functions per channel

- Access control defined for channel and peer resources:
  - User / System chaincode
  - Events stream
- Policies specify identities and include defaults for:
  - Readers
  - Writers
  - Admins
- Policies can be either:
  - Signature : Specific user type in org
  - ImplicitMeta : “All/Any/Majority” signature types
- Custom policies can be configured for ACLs



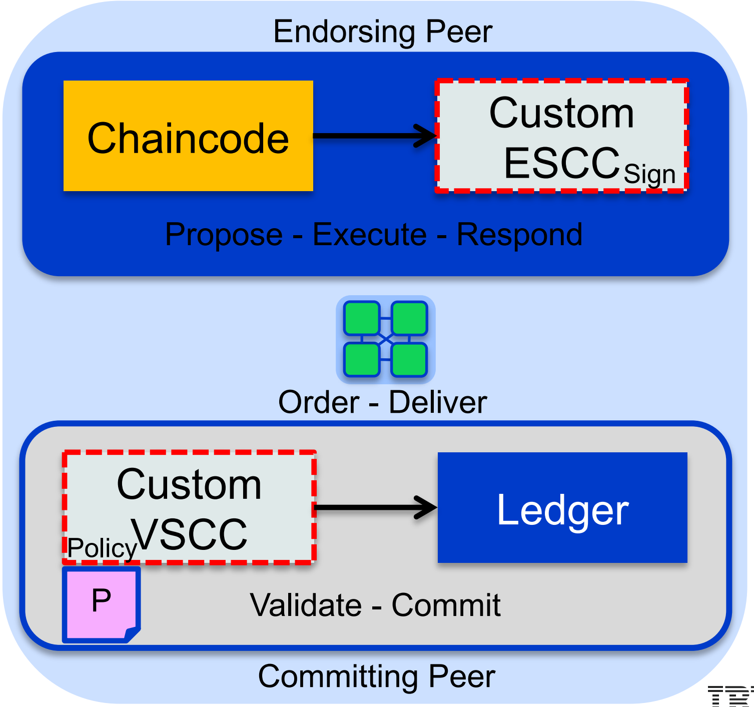
```
Policies: &SampleOrgPolicies
   ….
   MyPolicy: Type: Signature Rule: "OR(Org1.admin’)”

Application: &ApplicationDefaults
   ACLs: &ACLsDefault
   ….
   event/Block: /Channel/Application/MyPolicy
```

Pluggable endorsement and validation

Support for custom transaction endorsement and validation plugins

- Supports alternative transaction models for: State based endorsement, UTXO etc
- No need to recompile peer, **core.yaml** specifies additional golang plugins
- Support for custom:
  - **ESCC** : Endorsement System Chaincode
  - **VSCC** : Validation System Chaincode
  - **QSCC** : Query System Chaincode
  - **CSCC** : Configuration System Chaincode
  - **LSCC** : Lifecycle System Chaincode
- Chaincode associated with custom ESCC and VSCC at instantiation



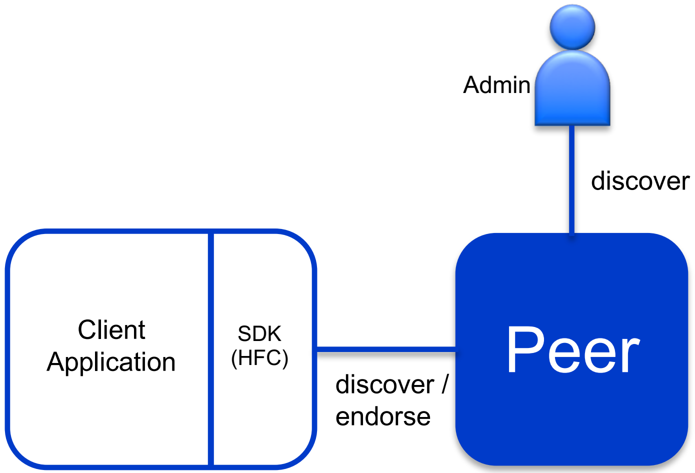
IBM Blockchain

https://jira.hyperledger.org/browse/FAB-8779

Service Discovery

Dynamically query peers to discover network service information

- Network metadata is shared between peers over GOSSIP
- Peers dynamically compute the following:
  - Configuration : MSP for all orgs in a channel
  - Peers : Peers that have joined a channel
  - Endorsers : Endorses for a specific channel/chaincode
- SDK sends dynamic query to peer to establish service connection information (including: endorsement policy, peers endpoints, TLS, CA and orderer endpoints).
- Administrator uses **discover** CLI to discover service information



https://jira.hyperledger.org/browse/FAB-8779

Further Information

Private Data Collections

- https://hyperledger-fabric.readthedocs.io/en/release-1.2/private-data/private-data.html

ACL mechanism per channel

- https://hyperledger-fabric.readthedocs.io/en/release-1.2/access_control.html

Pluggable endorsement and validation

- https://hyperledger-fabric.readthedocs.io/en/release-1.2/pluggable_endorsement_and_validation.html

Service Discovery

- https://hyperledger-fabric.readthedocs.io/en/release-1.2/discovery-overview.html
- https://hyperledger-fabric.readthedocs.io/en/release-1.2/discovery-cli.html

New tutorials

- https://hyperledger-fabric.readthedocs.io/en/release-1.2/whatsnew.html#new-tutorials

New documentation concepts

- https://hyperledger-fabric.readthedocs.io/en/release-1.2/whatsnew.html#new-conceptual-documentation

Fabric 1.1 experimental features

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

https://jira.hyperledger.org/browse/FAB-7746?jql=labels%20%3D%20Release-planning-1.1-TechPreview

Thank you

Barry Silliman
silliman@us.ibm.com

IBM **Blockchain**

www.ibm.com/blockchain

developer.ibm.com/blockchain

www.hyperledger.org