OSSユーザーのための勉強会 #24 Hyperledger

Hyperledger Fabric V1 の動作解説と活用例の紹介

IBM **Blockchain**

日本アイ・ビー・エム株式会社
IBMクラウド事業本部
IBM クラウド・マイスター
紫関 昭光

本日お話しすること

第一部

- **ブロックチェーン・アプリケーションの作り方**
- **Hyperledger Fabric V1 のコンセンサス・モデル**
- **Hyperledger Fabric V1 のチャネル、セキュリティ**
- **ブロックチェーン・クラウド環境を利用したコンソーシアムの運営**

第二部

- **Hyperledger Composer によるハンズオン**

Linux Foundation Hyperledger プロジェクト

- Linux Foundation により 2015年に発表され、Linux Foundationの史上最速で成長しており、現在 150以上のメンバーを擁している。
- Hyperledger: 様々な業界のビジネスに資するブロックチェーン・テクノロジーを推進するオープンソースかつオープンに統治された協同活動であり、Linux Foundation の下で行われている。
- Hyperledger Fabric: ブロックチェーン・フレームワークの一実装で、Hyperledgerプロジェクトの一つである。モジュラー・アーキテクチャによりアプリケーションやソリューションの開発の基盤となることを意図している。

共有台帳技術の実現を
どの一つの企業や業界が行うよりも
迅速かつ高度に可能にする

www.hyperledger.org

Premier

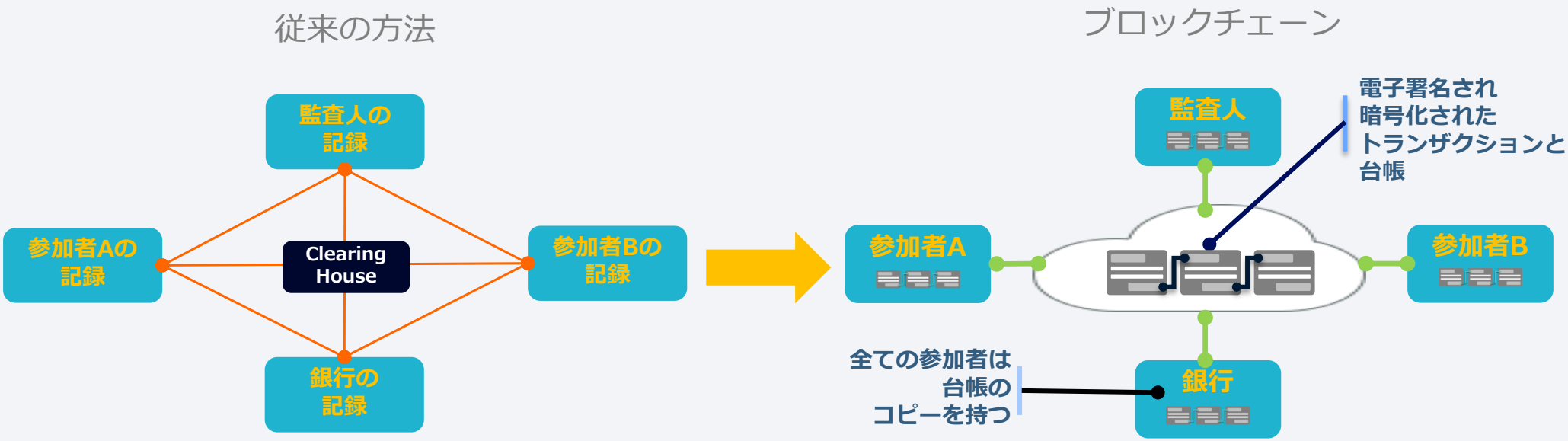
General

Associate

Source: https://www.hyperledger.org/about/members
Updated Sep 6, 2017

ブロックチェーンでビジネス・プロセスがどう変わるか？

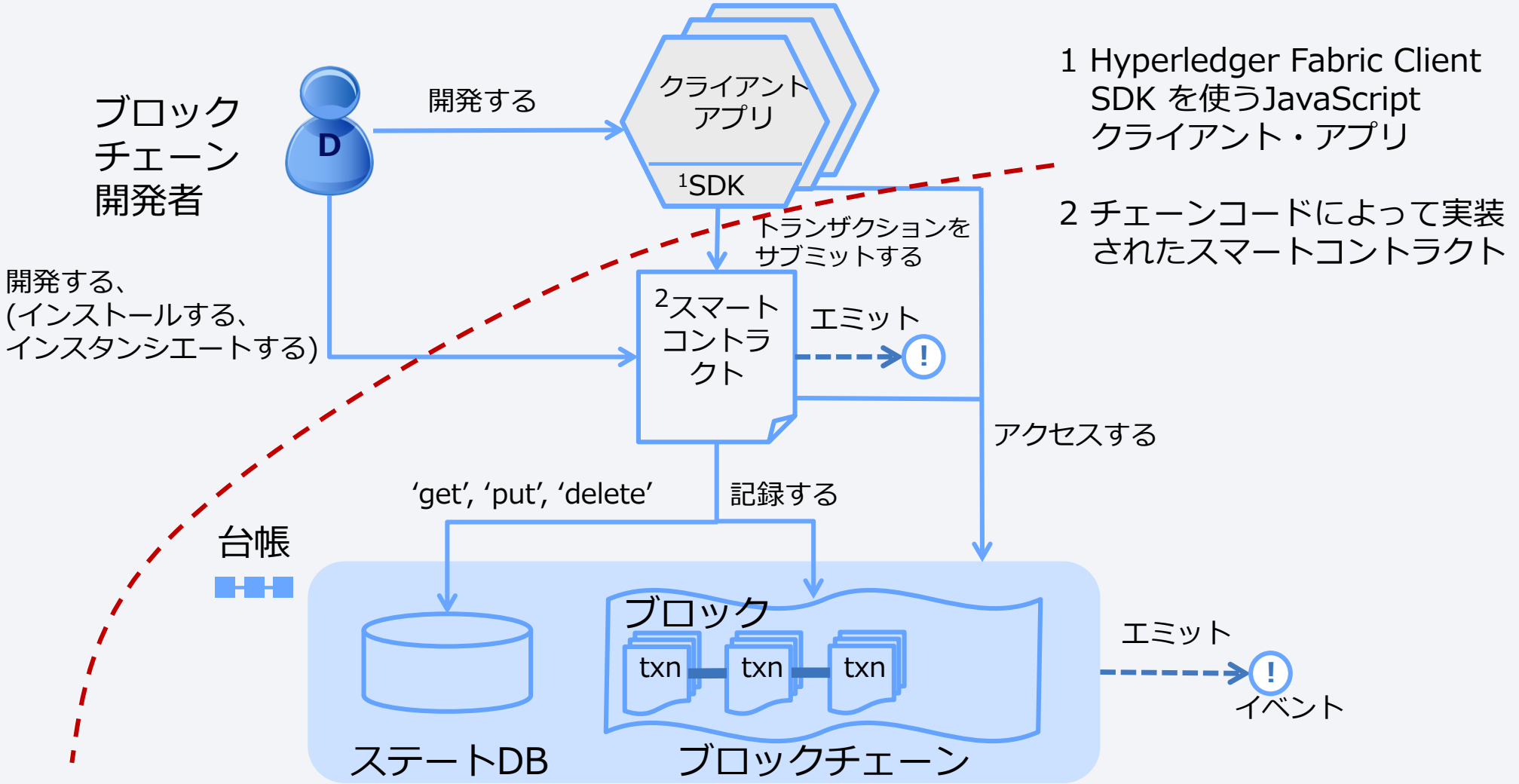
- ネットワーク上で事実を確定（ブロックチェーンに記録＝事実の確定）
- 複数参加者が一つの真実に基づき協業
- 予め合意したビジネス・ルール（スマートコントラクト）に則って取引を自動化
- 高可用性、セキュリティーとプライバシー保護、オープン・ソースによる低コスト



… 非効率、高コスト、脆弱

… コンセンサス、来歴、改竄不可能、ファイナリティ

Hyperledger Fabric アプリケーションと台帳の概要



ブロックチェーン・アプリケーション

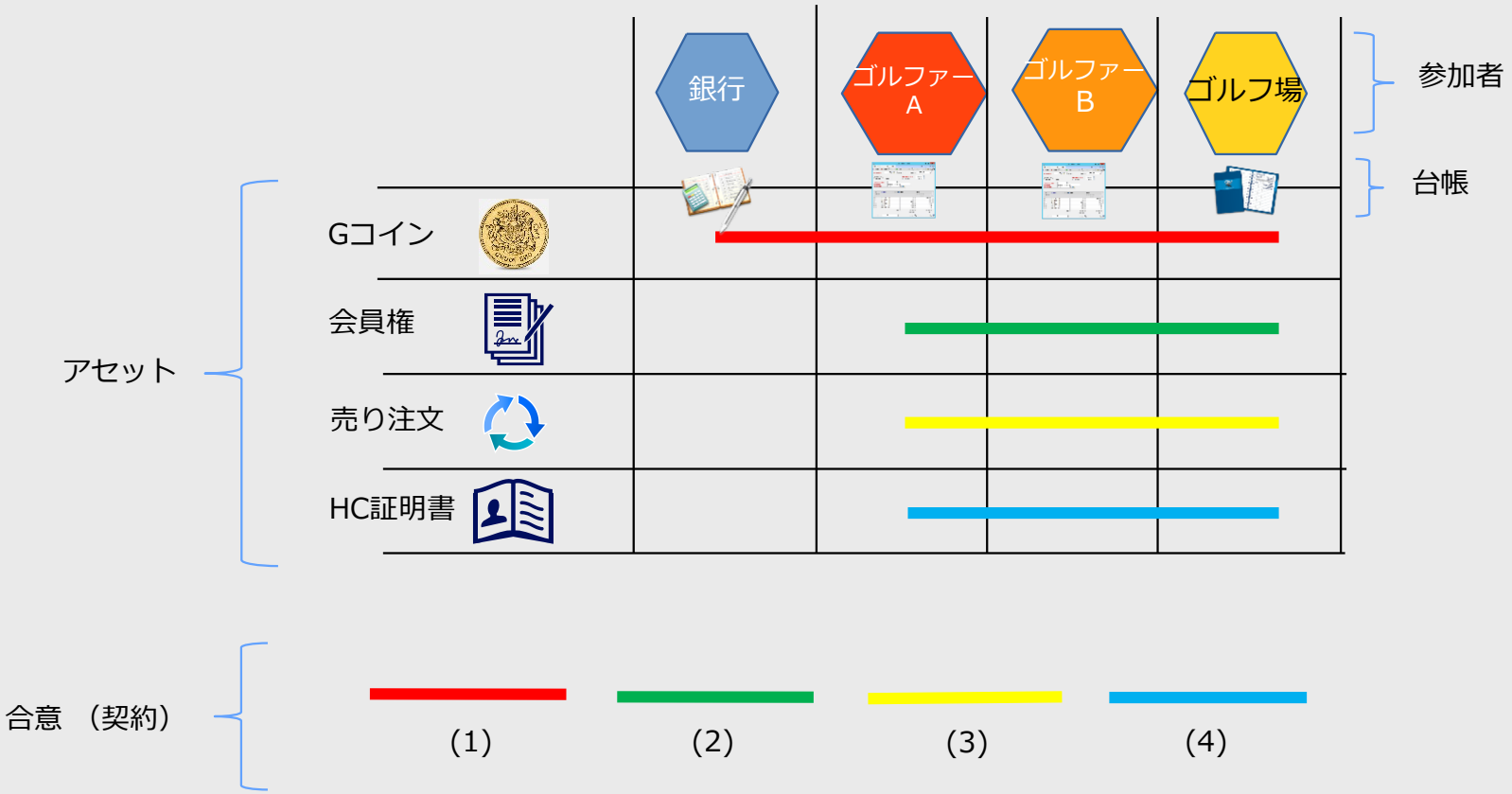
スマートコントラクト … 参加者同士で同意したビジネス・ネットワークのルール（**法律**）

- ビジネス・ロジックはチェーンコードに含まれる。プログラム言語は選択可能。
- スマートコントラクトの各呼び出しがブロックチェーン・トランザクションとなる。

アプリケーション … 参加者が自由に作れる（**個々の企業のサービス、約款**）

- スマートコントラクトを呼び出し、台帳を読み書きする
- トランザクションの記録に直接アクセスすることも可能
- イベントを処理することも可能

ユースケース例： ゴルフ会員権売買



ビジネス・ルール

合意事項 （チェーンコードの前提）

(1) Gコインの購入と換金 （アセット： Gコイン）

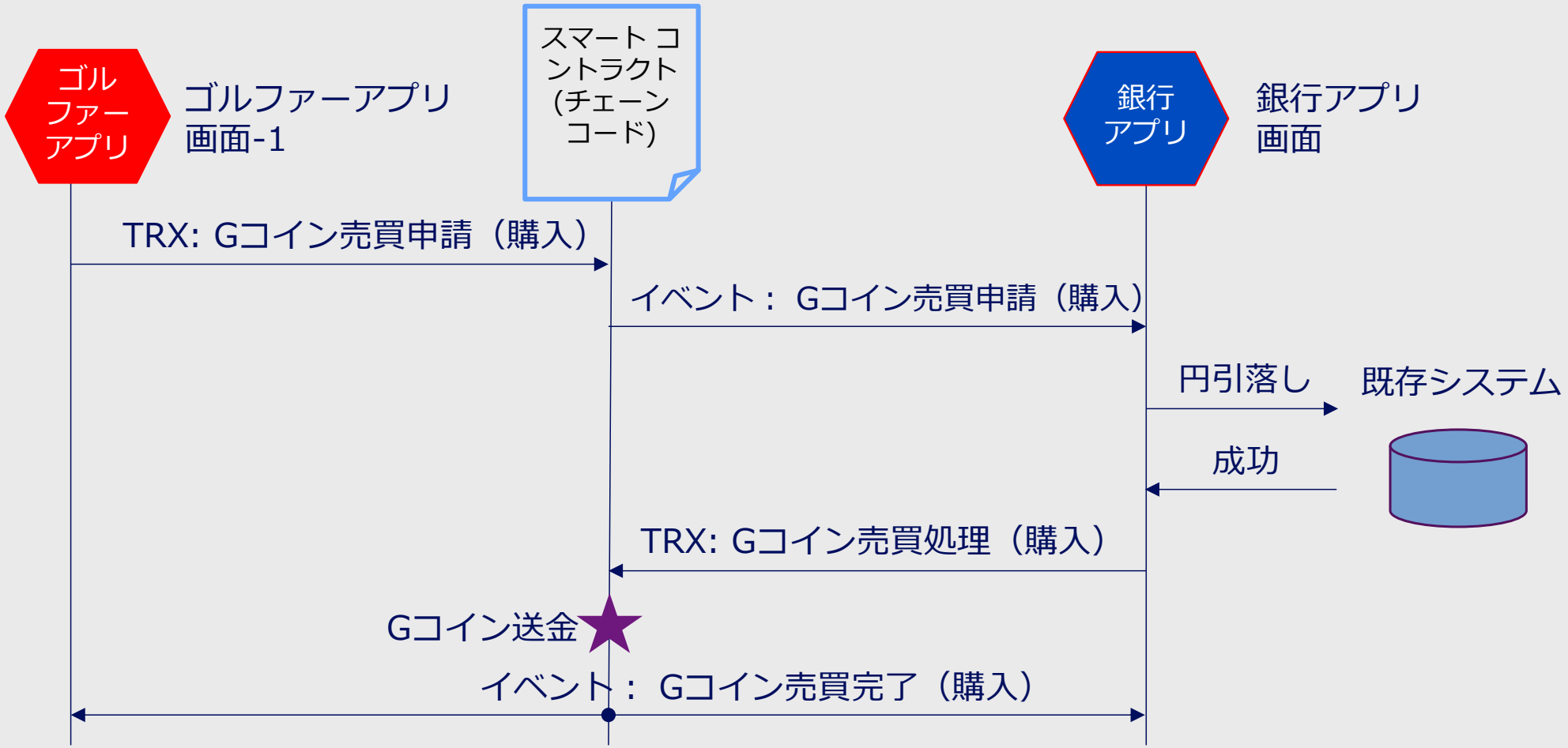
(2) 会員権の売買 （アセット交換： Gコイン ⇔ 会員権）

(3) 会員権の売り注文（アセット共有： 売り注文）

(4) HC証明書の発行 （アセット交換： HC証明書）

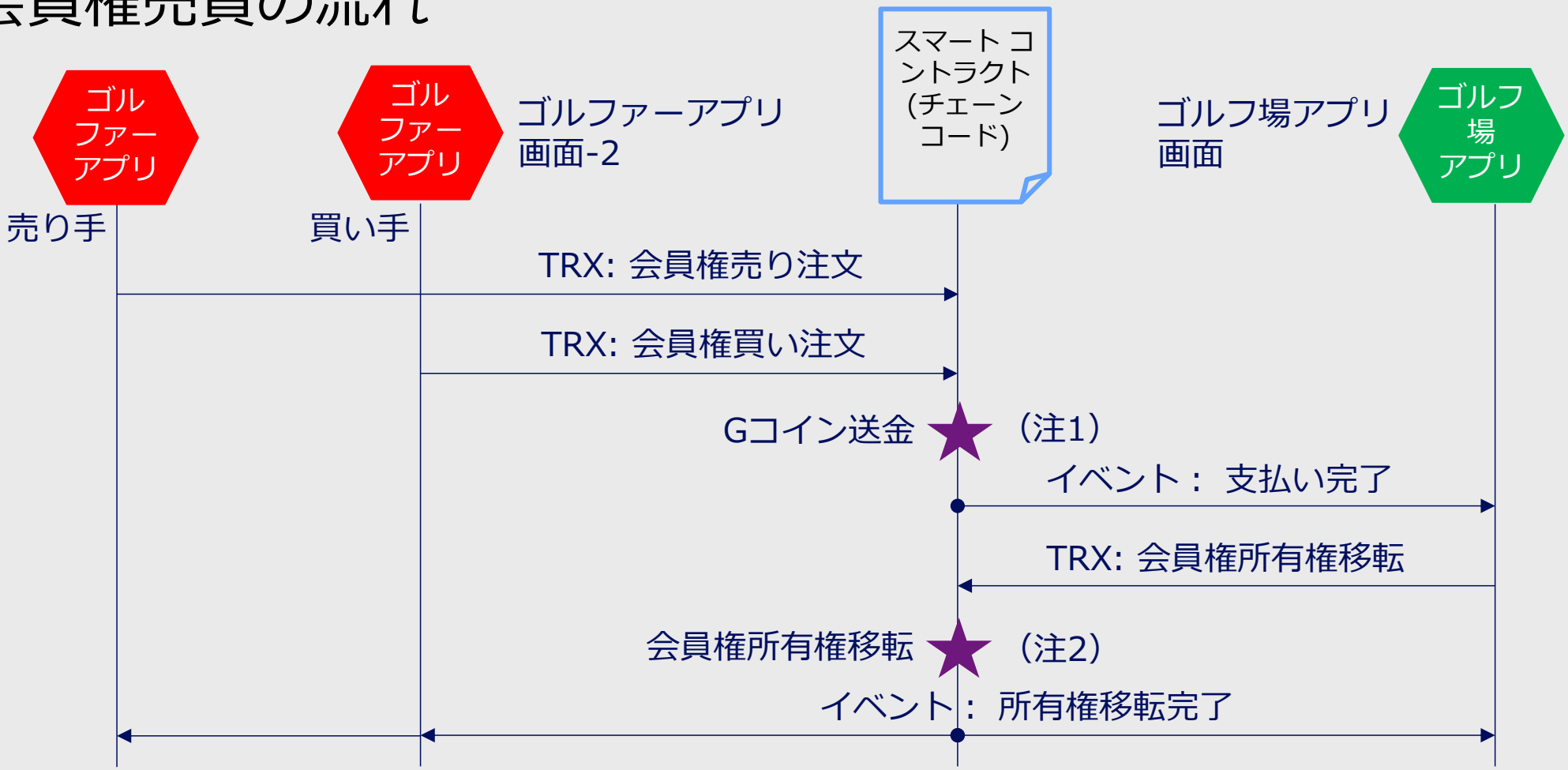
→ チェーンコードとして実装される
→ トランザクション実行はチェーンコードを呼び出すことで行われる

Gコイン購入の流れ



- 円引落し確認後にGコインが銀行からゴルファーに送金される

会員権売買の流れ



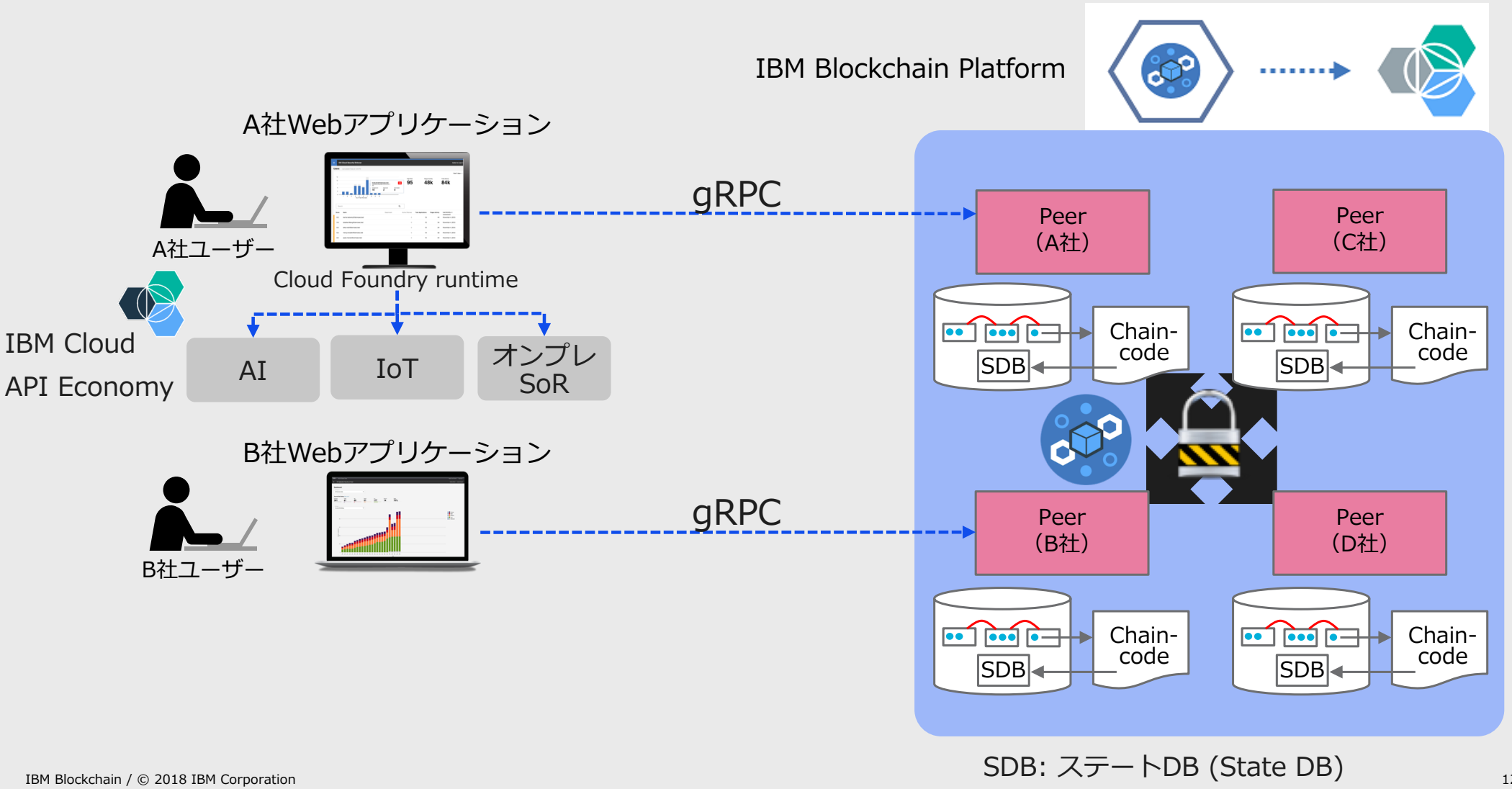
（注1）会員権のゴルフ場による電子署名、入会条件を確認後、買い手から売り手とゴルフ場にGコインを送金
（注2）会員権の所有者を買い手に変更し、ゴルフ場の電子署名を行う。

会員権売り注文に関するビジネス・ルール

- ビジネス・ルール
  ① 会員権売り注文に記されている売り手とトランザクション発行者が一致している。
  ② 会員権売り注文はレジストリーに未登録の新規の注文である。
  ③ 会員権売り注文の売り注文受付日にトランザクションのシステム・タイム・スタンプをコピーする。
  ④ 買い注文受付日はブランクにする。
  ⑤ 譲渡完了日はブランクにする。
  ⑥ 会員権売り注文をレジストリーに追加（add）する。

- 考えられる悪意あるアプリケーションの検出
  - 他人になりすまして会員権売り注文を行う。
    → ルール①でエラーになる。
  - 処理済みの会員権売り注文を再利用することで過去の注文を改ざんする。
    → ルール②⑥でエラーになる。
  - 売り注文受付日、買い注文受付日、譲渡完了日に不正な値を記入する。
    → ルール③④⑤で正しく修正される。

ブロックチェーン・クラウド環境　IBM Blockchain Platform

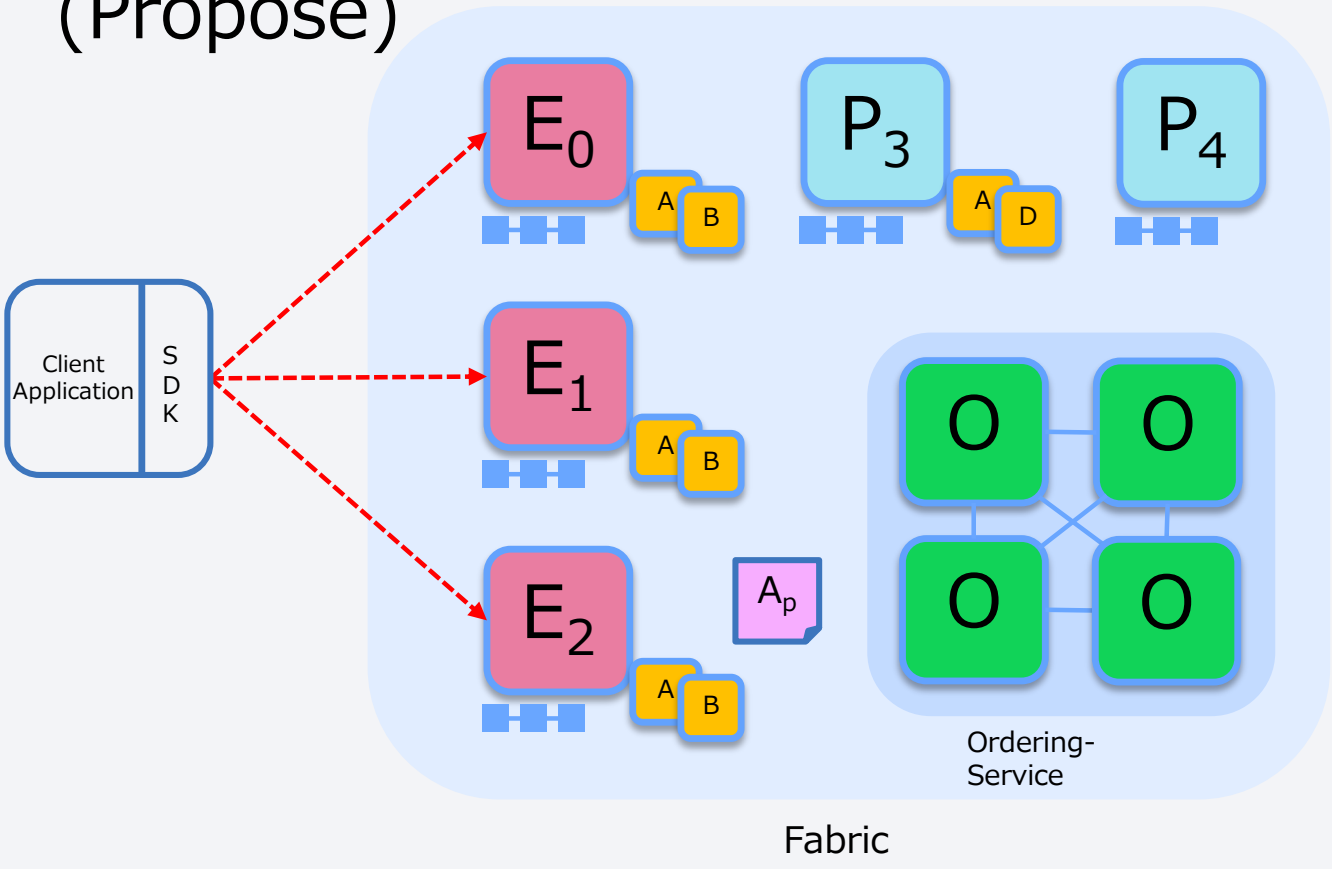


SDB: ステートDB (State DB)

Hyperledger Fabric ノードと役割

	**Committing Peer**: 台帳を保持し、トランザクションをコミットする。 スマートコントラクト（チェーンコード）を持っていてもよい。
	**Endorsing Peer**: 特別なCommitting peer。エンドースメントのためのトランザクション・プロポーザルを受け、エンドースメントまたは却下を返す。 スマートコントラクトは必須。
	**Ordering Nodes (service)**: トランザクション・ブロックを台帳に追加することを承認し、Committing および Endorsing peer nodes に伝える。スマートコントラクトも台帳も持っていない。

トランザクションの例 (1/7) トランザクションの申請 (Propose)

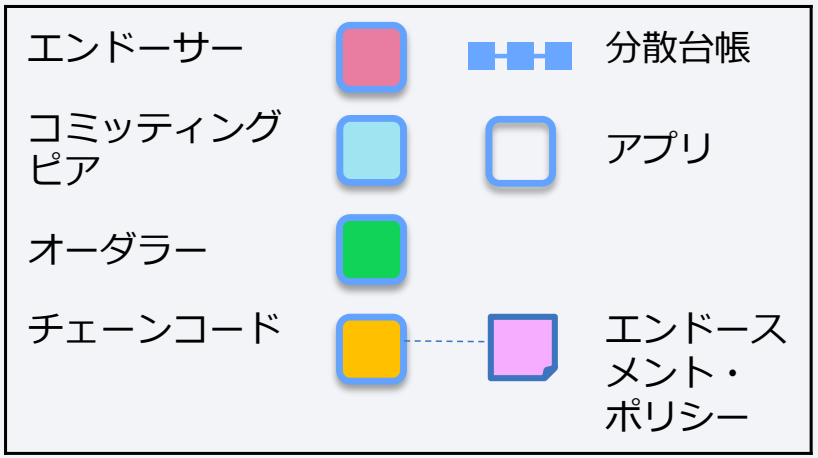


アプリがトランザクションを申請

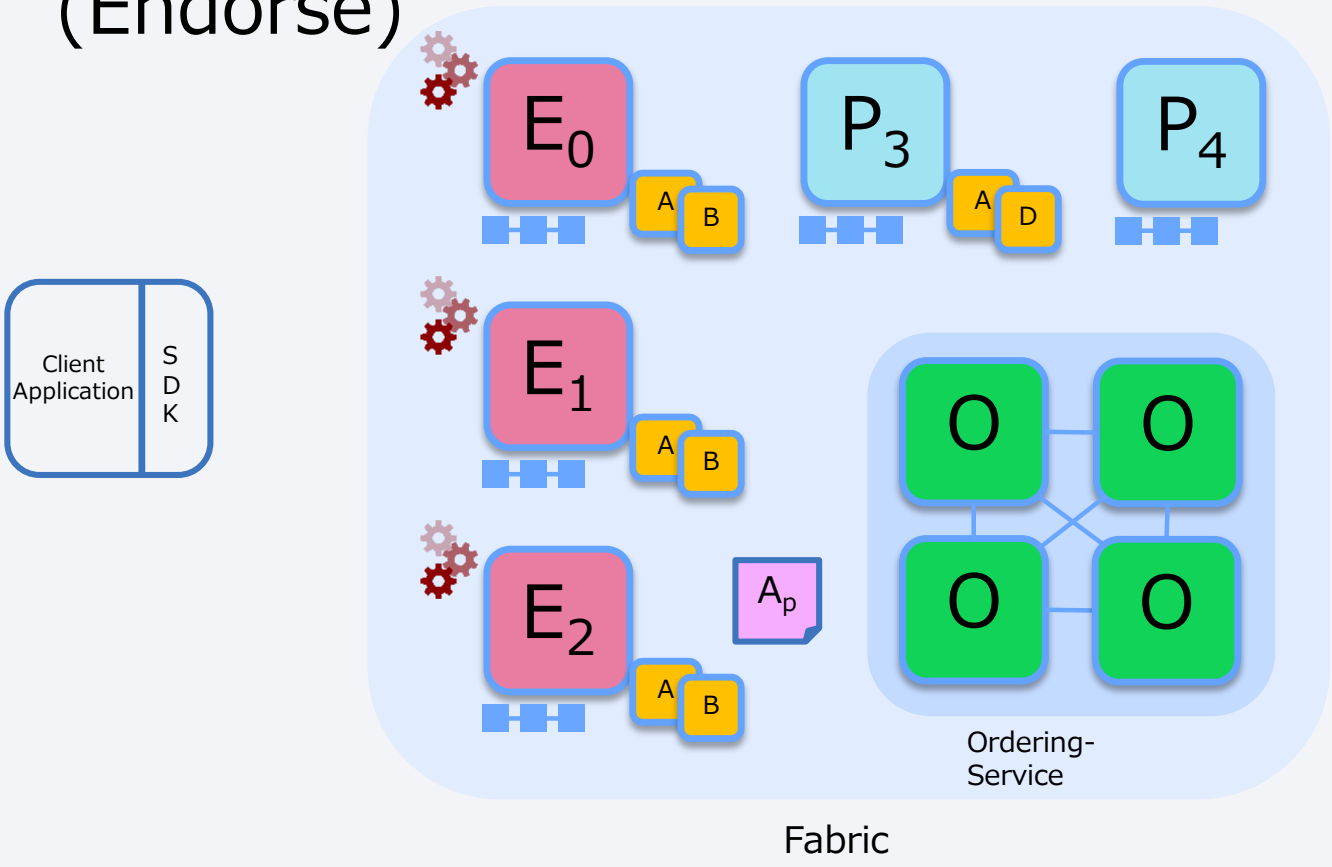
エンドースメント・ポリシー:
- “E_0, E_1, E_2 のサインが必要”
- (P_3, P_4 は含まれない)

クライアントはスマートコントラクトAを実行するトランザクションをエンドーサー {E_0, E_1, E_2}に申請する

凡例:



トランザクションの例 (2/7) トランザクションを仮実行 (Endorse)



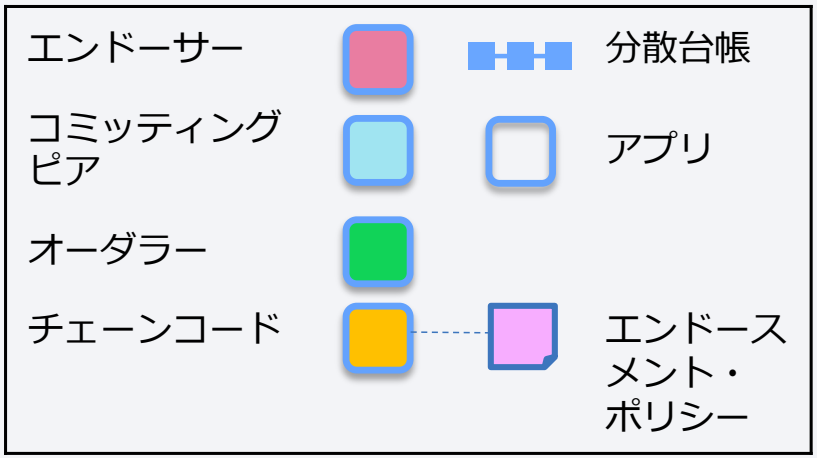
エンドーサーがプロポーザルを仮実行

エンドーサー E_0, E_1 & E_2 はそれぞれに申請されたトランザクションを仮実行する。

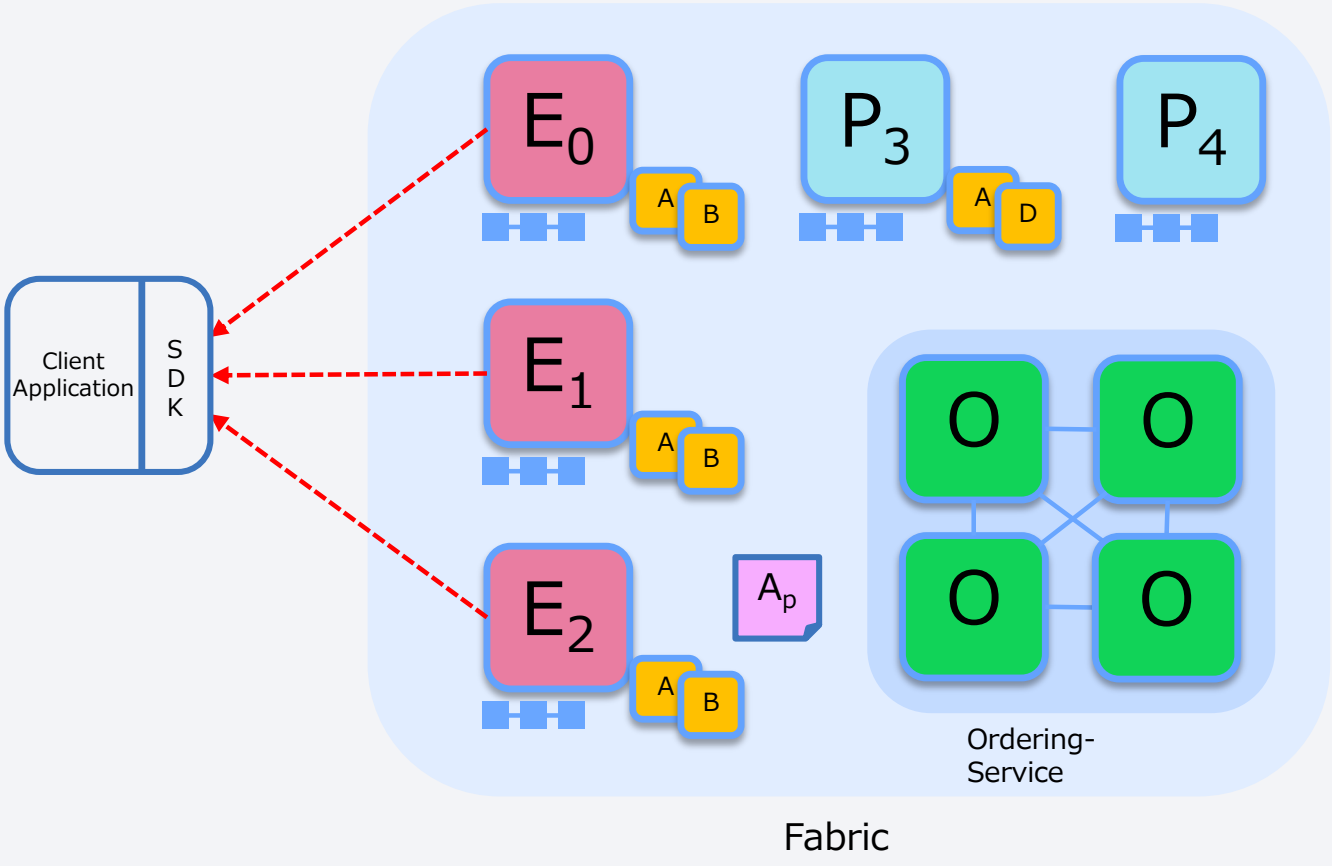
実行結果は分散台帳に記録されないが、Read/Write セットとしてアプリケーションに戻される。

トランザクションは署名や暗号化が可能。

凡例:



トランザクションの例 (3/7) 申請への応答

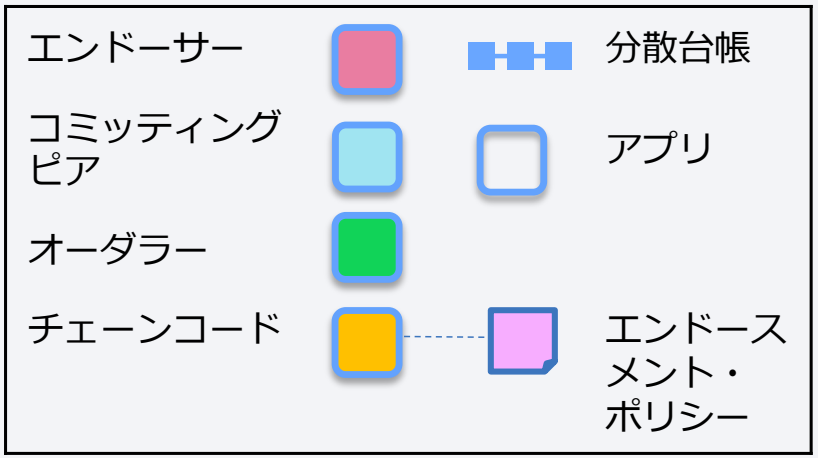


アプリケーションは応答を受診

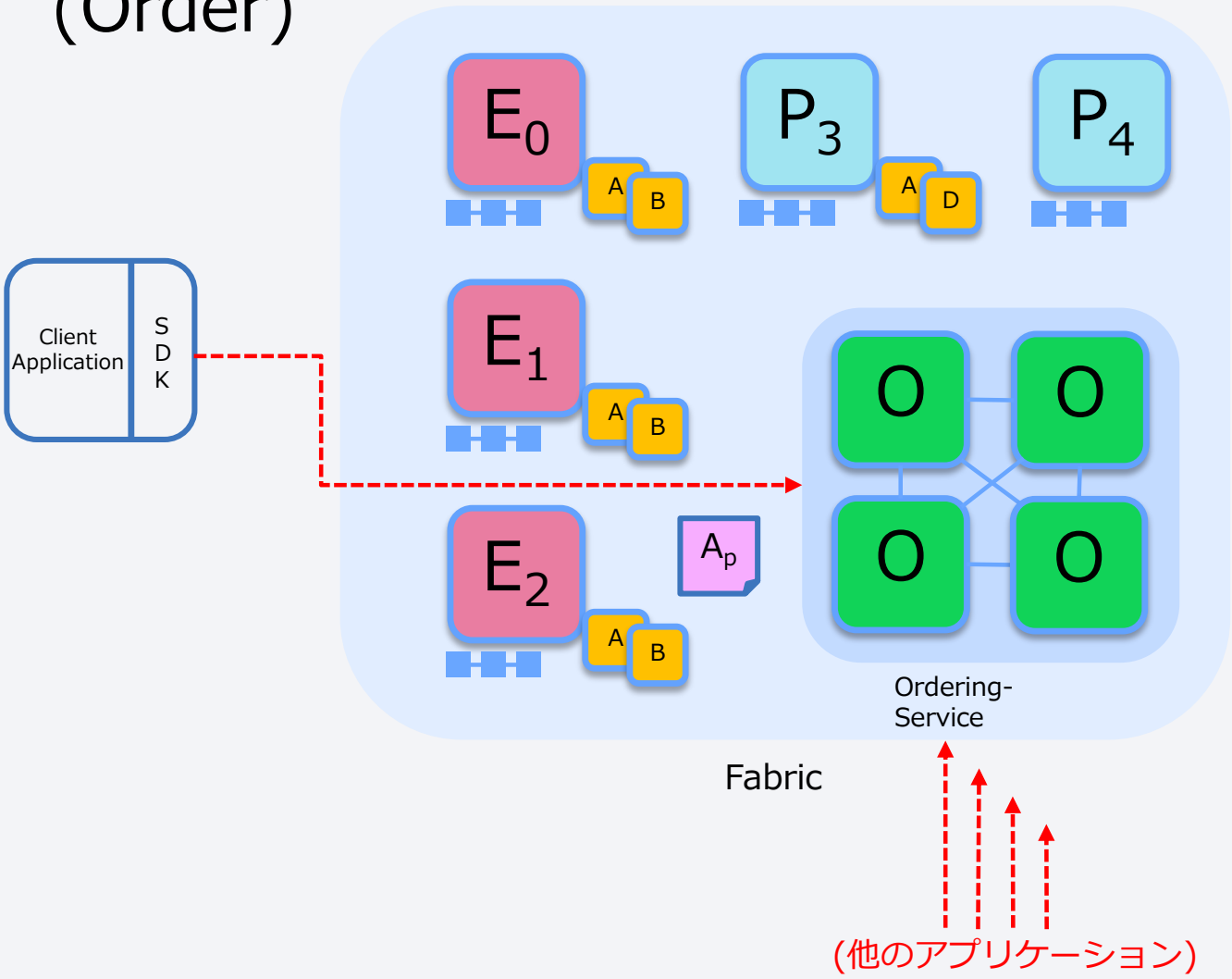
Read/Write セットが非同期にアプリケーションに返される。

Read/Write セットはエンドーサーにより署名され、レコードバージョン番号を含む。

凡例:



トランザクションの例 (4/7) トランザクションの順序付け (Order)

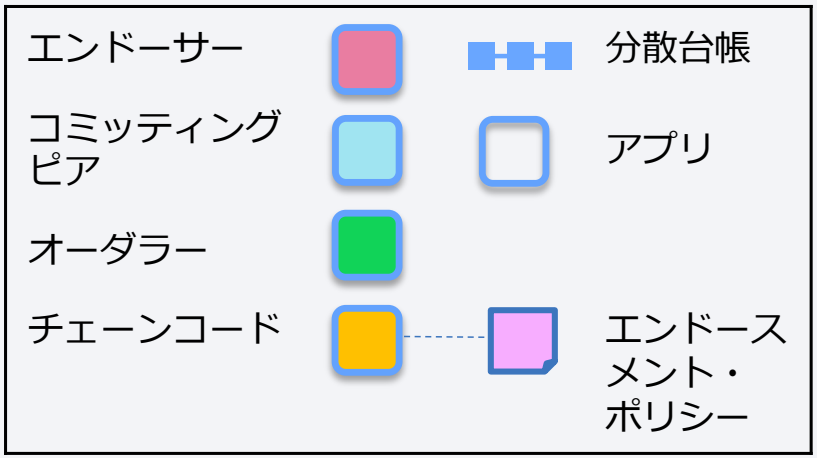


アプリは応答をオーダラーに送信

アプリケーションはエンドーサーから受け取った情報を順序付けのためにオーダラーへ送る。

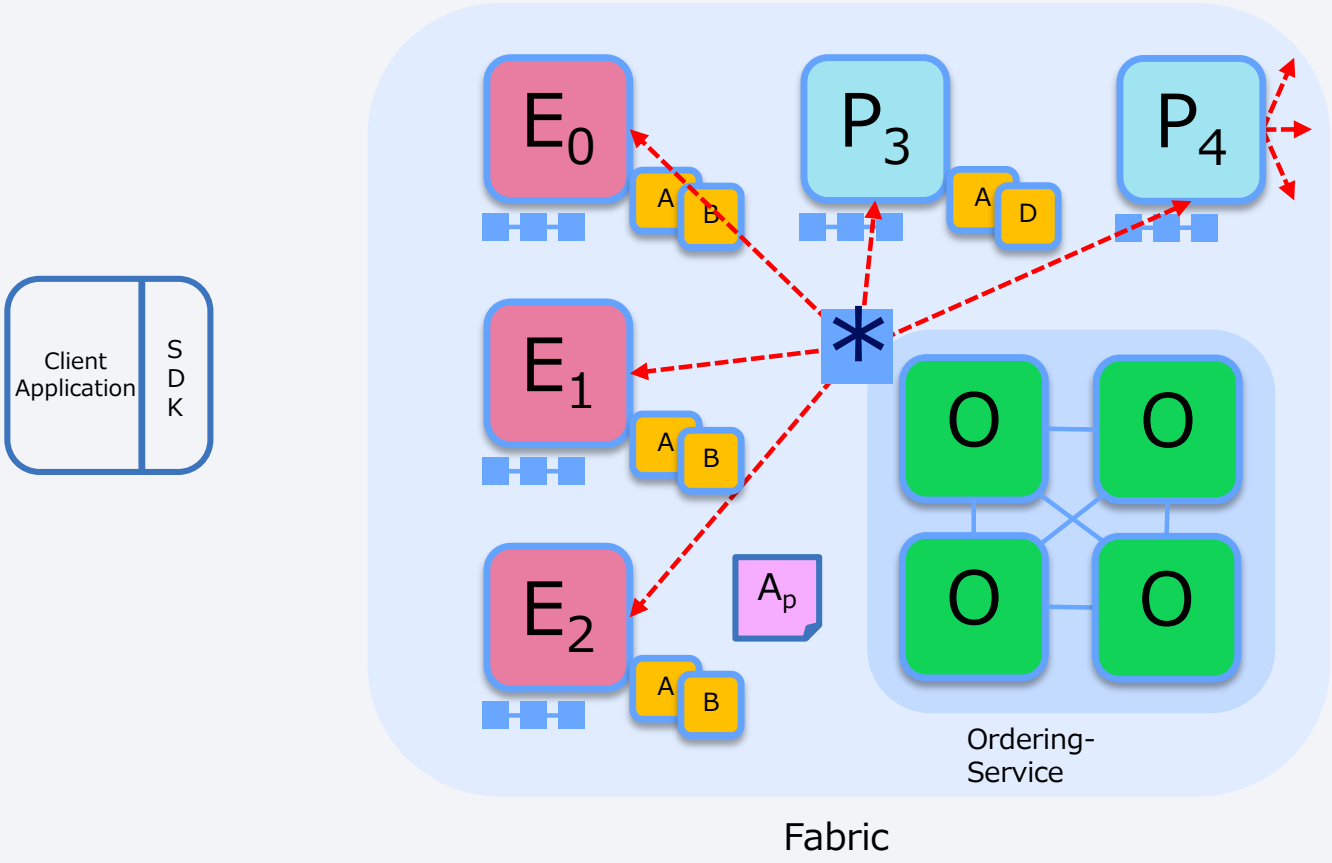
順序付け (Ordering) は、ブロックチェーンネットワークの中で、他のアプリケーションから送られてたトランザクションと並行して行われる。

凡例:



トランザクションの例 (5/7) トランザクションの配信

全Committing peersへ通知

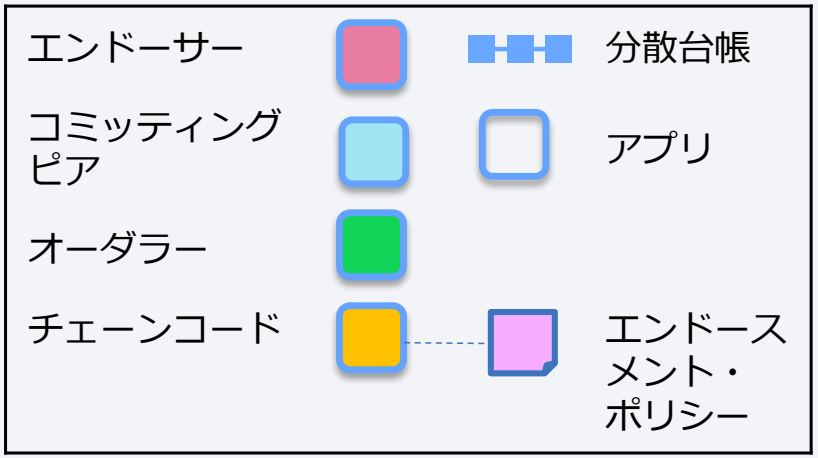


オーダラーはトランザクションをブロックにパッケージしてコミッティングピアへ配信する。

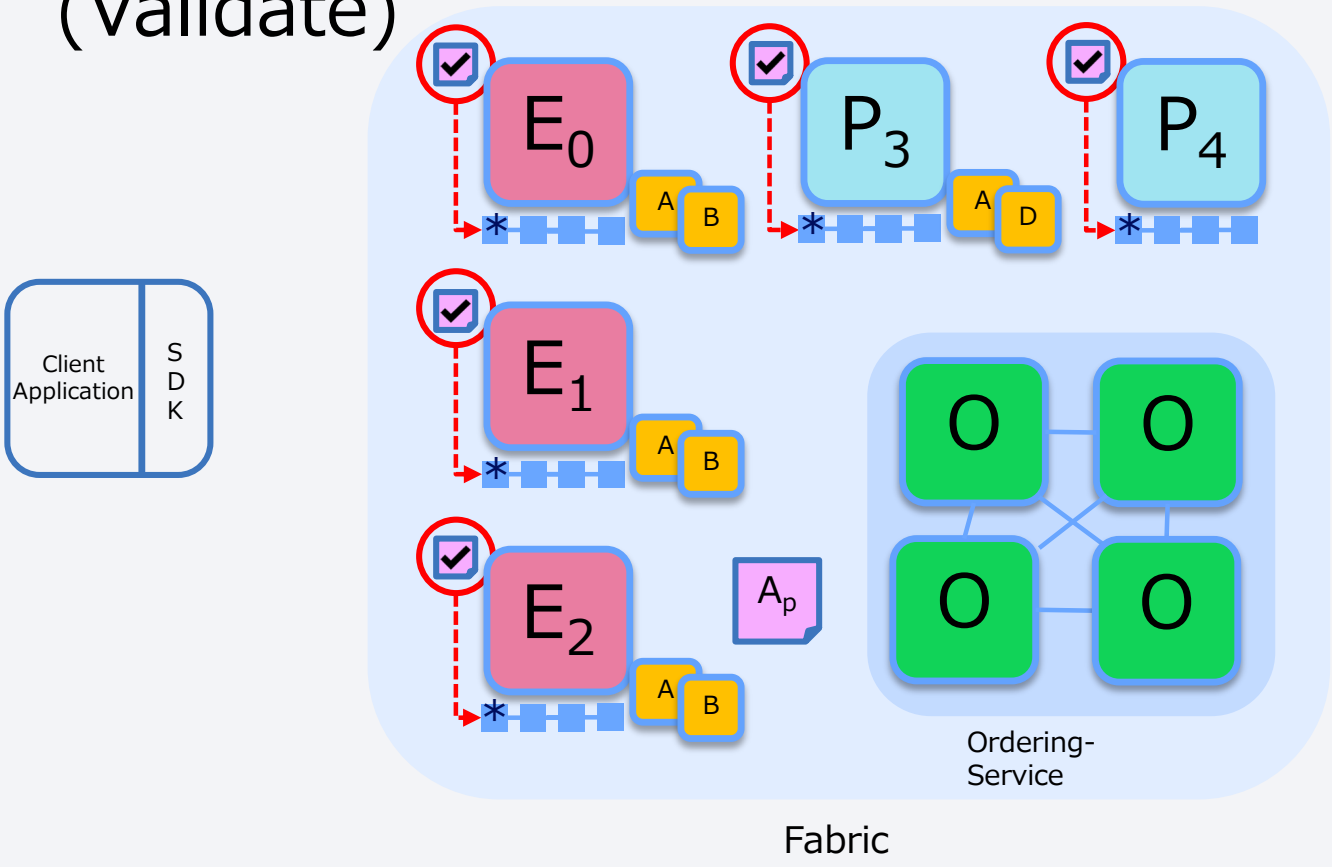
複数のオーダリングアルゴリズムから選べる:

- SOLO (Single node, development)
- Kafka (Crash fault tolerance)

凡例:



トランザクションの例 (6/7) トランザクションの検証 (Validate)

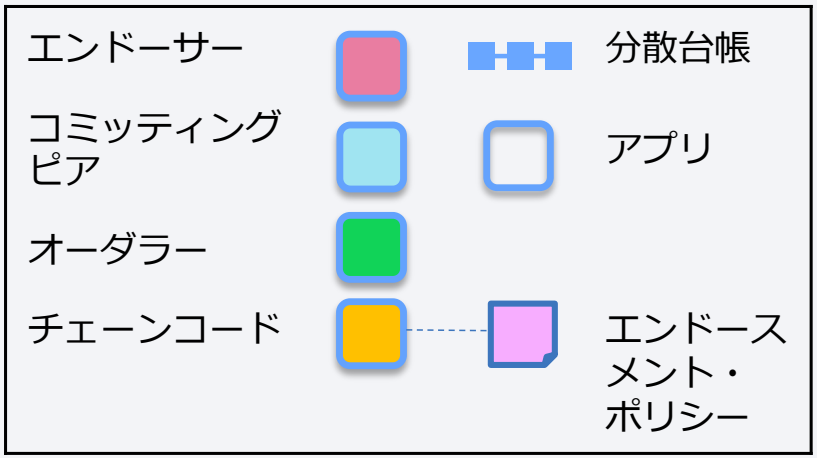


コミッティングピアはトランザクションを検証

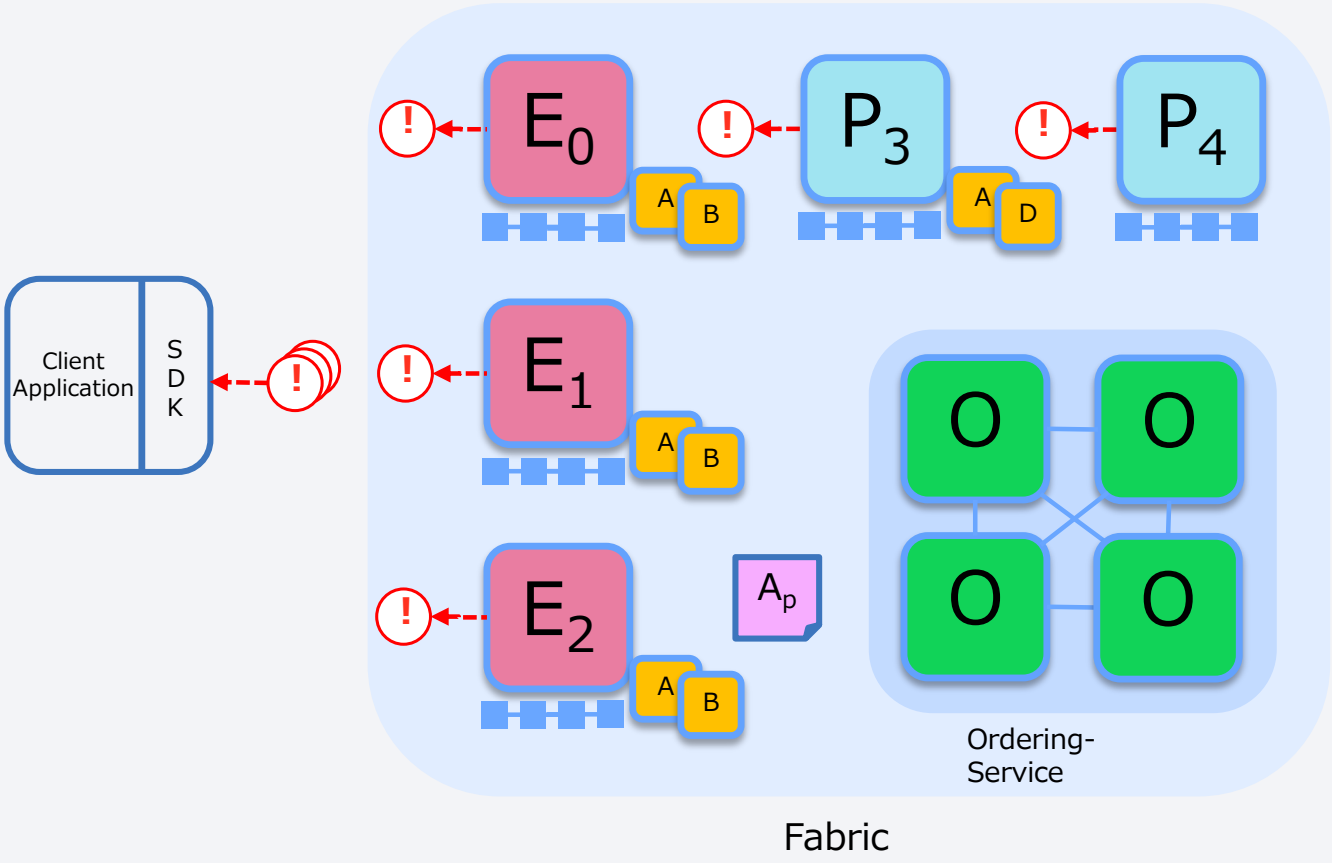
各コミッティングピアはエンドースメント・ポリシーにしたがってトランザクションを検証する。

また Read/Write セットがまだ有効であることを確認し、問題なければ台帳へ書き込む。

凡例:



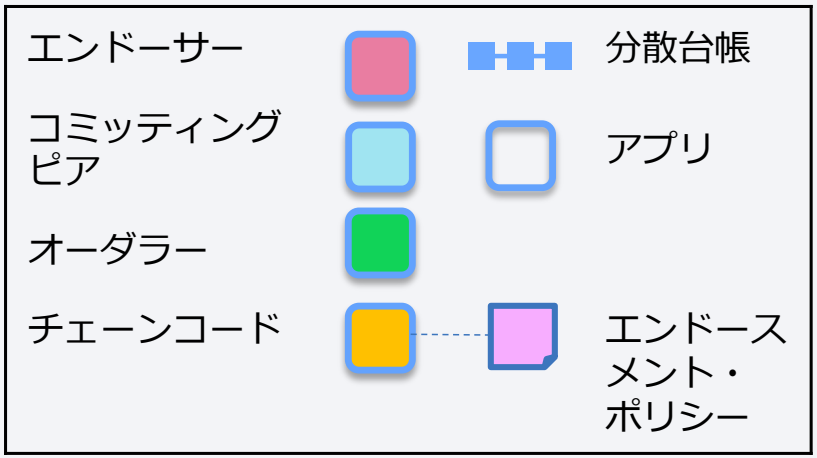
トランザクションの例 (7/7) トランザクションの通知



コミッティングピアはアプリへ通知

アプリケーションはトランザクション申請の結果の通知を受けることが出来る。

凡例:



Bitcoin の場合

Bitcoin ブロックチェーンのコンセンサス方式

全てのフルノードは、検証リストに従って各トランザクションを各々で検証する
→ トランザクションの正当性を確認する

マイニングノードは、それらのトランザクションを各々で新しいブロックに詰め込み、合わせてProof of Wok アルゴリズムにより計算を行ったことを示す
→ ブロックの作成と Proof of Work競争への勝利宣言をする

全てのノードは、各々で新しいブロックを検証し、ブロックチェーンに繋ぐ
→ 競争の勝者による正しいブロックであることを検証し、チェーンに繋いで登録する

全てのノードは、Proof of Work による計算の蓄積が最大のチェーン（一番長いチェーン）を各々で選ぶ
→ チェーンが分岐している場合は一番長いチェーンに新しいブロックを繋ぐ

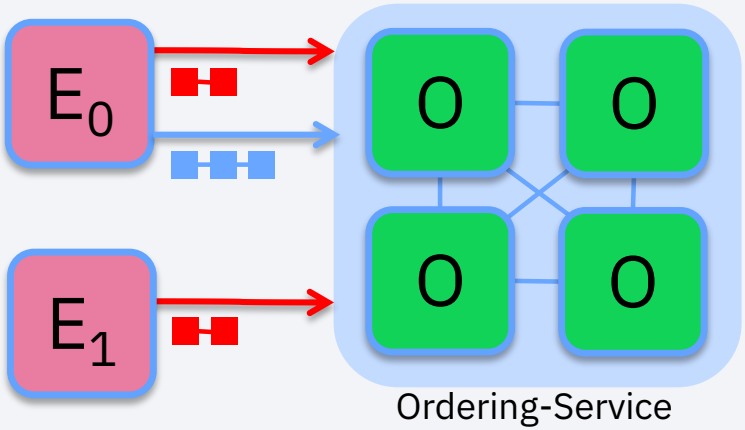
チャネルとオーダリングサービス

ブロックチェーンネットワークをチャネルに分割できるようになりました。

分散台帳、チェーンコードインスタンスはチャネルごとに存在します。

チャネル

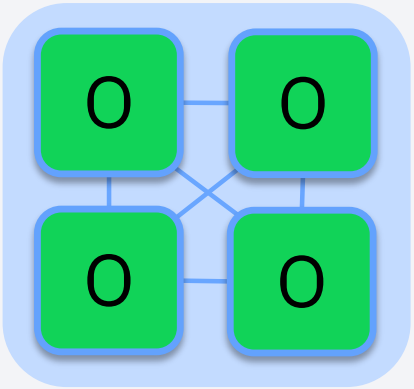
チャネル毎にトランザクションを異なる台帳で管理



- 台帳はチャネルごとに存在する
  - 台帳はネットワークの全Peer で共有可能
  - 台帳を特定の参加者だけで共有することも可能
- チェーンコードはワールドステートにアクセスが必要なPeer にインストールされる
- チェーンコードはPeer用にチャネルごとにインスタンス化される
- Peerは複数のチャネルに参加可能
- 並行処理によるパフォーマンスとスケーラビリティが向上

オーダリングサービス

オーダリングサービスはトランザクションをブロックにパッケージしてPeerへ送る。
コミュニケーションはチャネルを通じて行う。

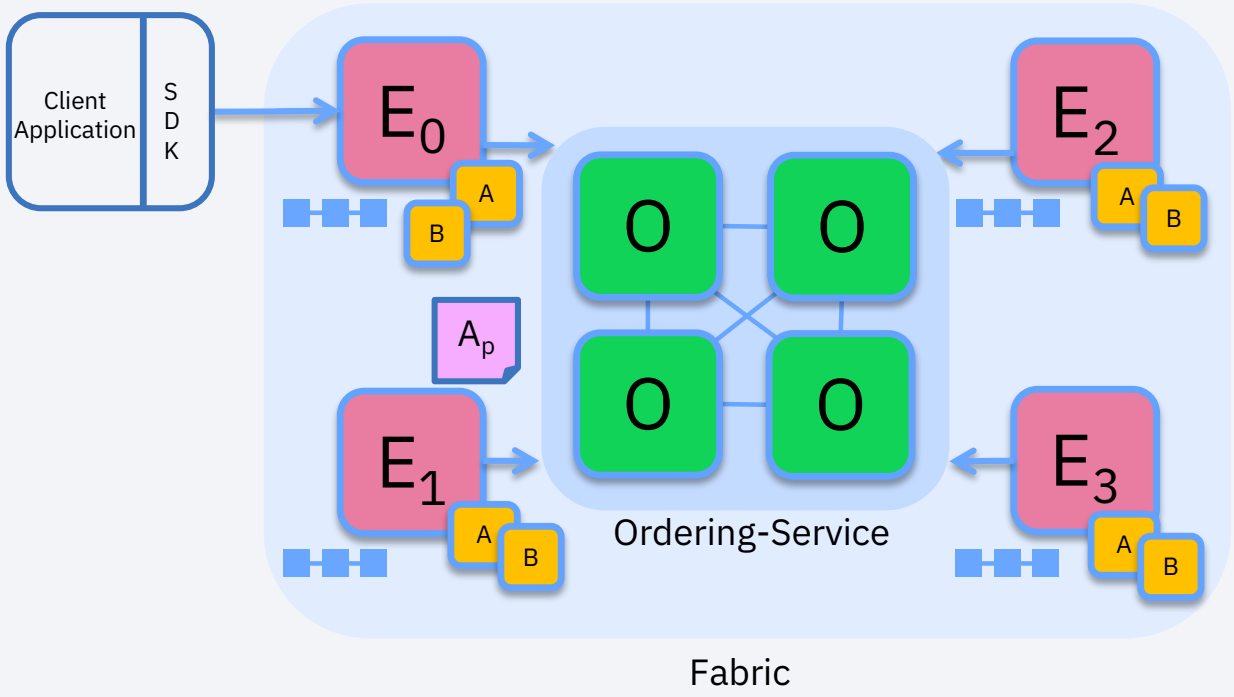


Ordering-Service

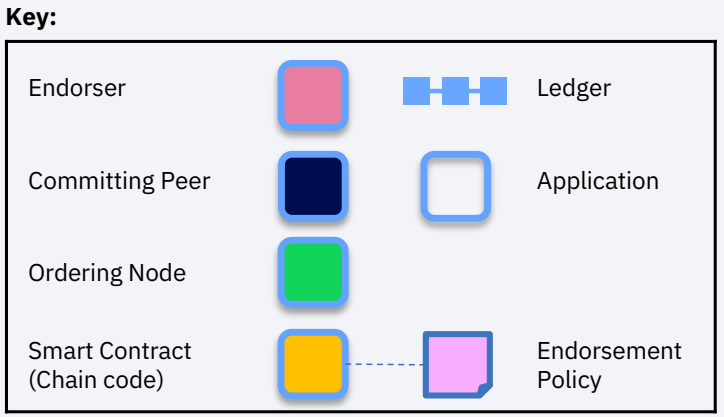
オーダリングサービスの構成：

– **SOLO**
  • 開発用のシングルノード
– **Kafka：**クラッシュ・フォルト・トレラント・コンセンサス
  • 最小ノード構成 3～n
  • 奇数ノードを推奨

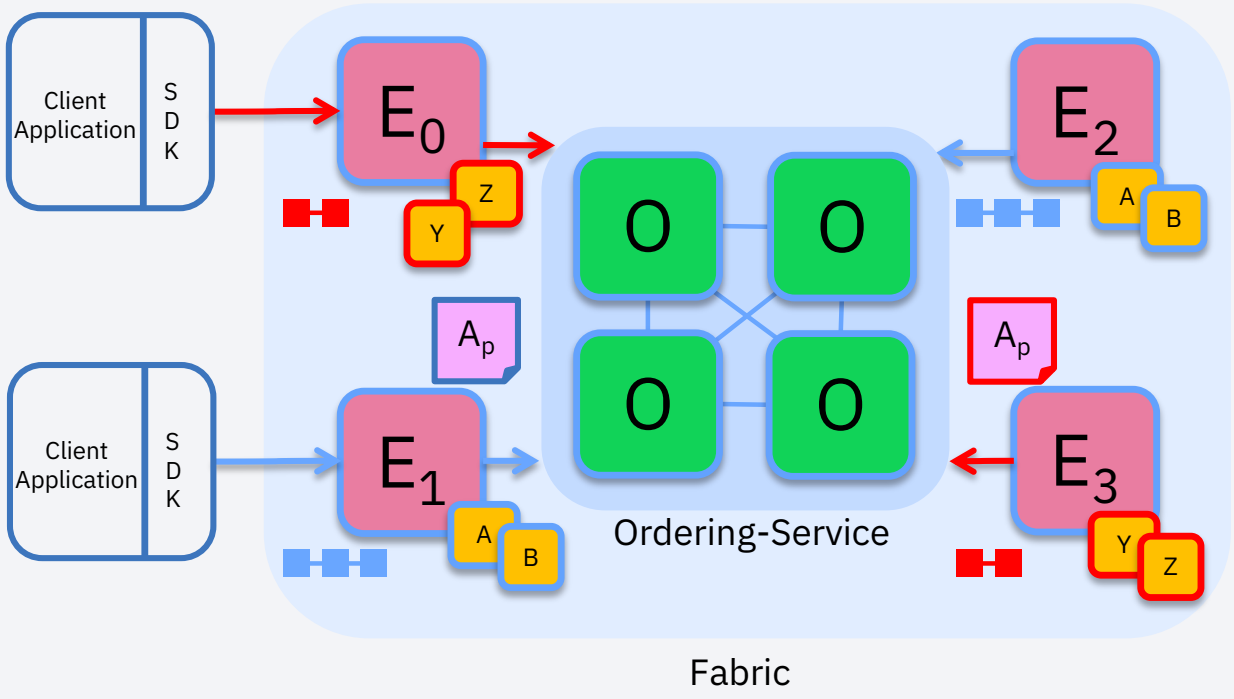
一つのチャネルによるエンドースメント



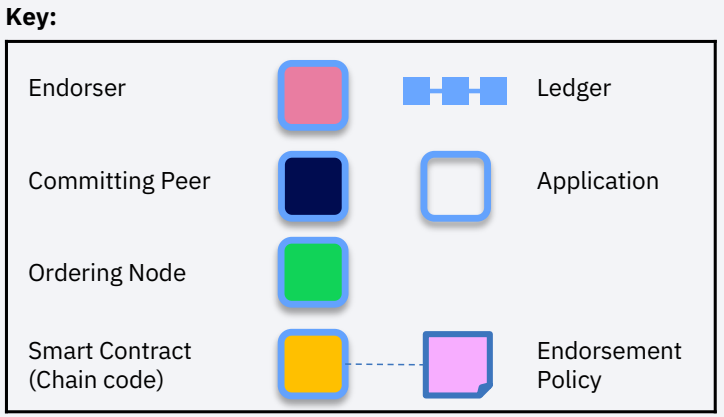
- 0.6 PBFTモデルと同様
- 全てのPeerが同じシステムチャネル（青）にコネクトする
- 全てのPeerは同じチェーンコードと台帳を保持する
- Peers E_0, E_1, E_2, E_3 によるエンドースメント。



複数チャネルによるエンドースメント



- Peers E_0 , E_3 は**赤**のチャネルにコネクトしてチェーンコード**Y**、**Z** を利用
- Peers E_1, E_2 は**青**のチャネルにコネクトしてチェーンコード**A**、**B** を利用



Endorsement Policies

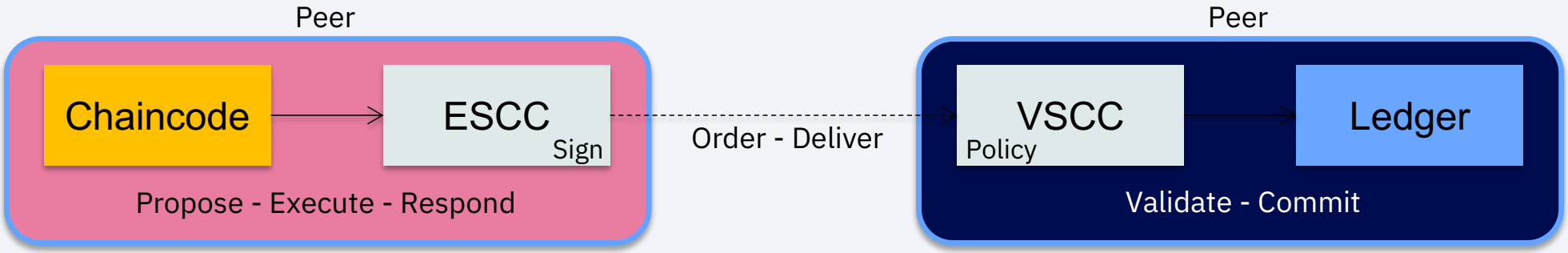
エンドースメントの成立条件はポリシーによって設定可能です。

エンドースメントポリシーはチェーンコードがチャネルに対してインスタンス化されたときに指定されます。

エンドースメントポリシーとは？

エンドースメントポリシーはトランザクションが承認される条件を規定する。トランザクションはポリシーに従って承認された場合に限って正当と考えられる。

- チェーンコードはそれぞれエンドースメントポリシーに関連付けられる
- デフォルトの実装では簡単な宣言型の言語でポリシーを記述する
- ESCC (Endorsement System ChainCode) はエンドーシングピアにてプロポーザルへのレスポンスに署名する
- VSCC (Validation System ChainCode) はエンドースメントを検証する



エンドースメントポリシーの指定方法

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

This command instantiates the chaincode *mycc* on channel *mychannel* with the policy AND('Org1MSP.member’)
-c: constructor message

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 required, and **ROLE** is either “member” or “admin”.

エンドースメントポリシーの例

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')

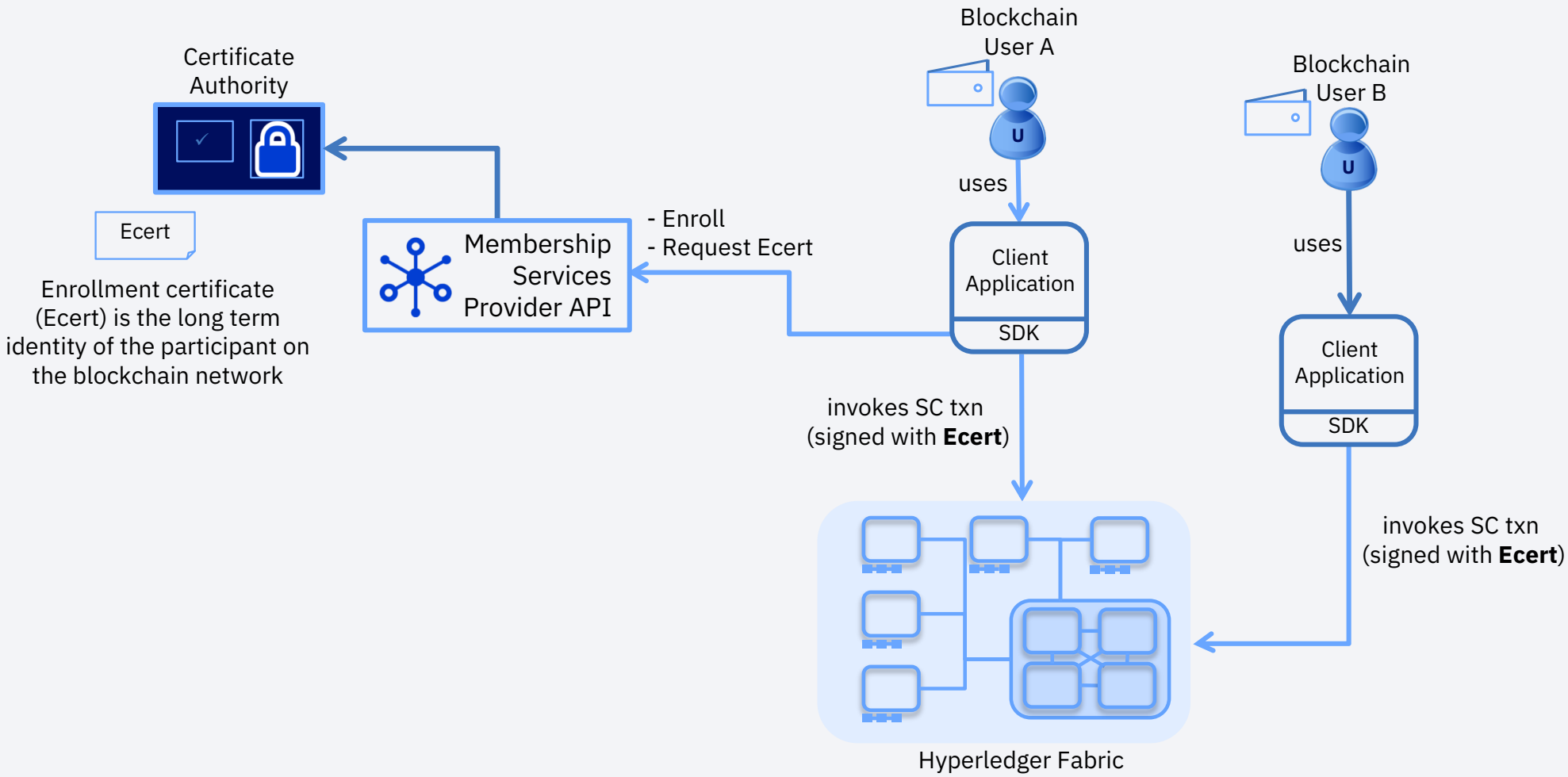
Request either one signature from a member of the Org1 MSP or (1 signature from a member of the Org2 MSP and 1 signature from a member of the Org3 MSP)

– OR('Org1.member', AND('Org2.member', 'Org3.member'))

パーミッション型の台帳アクセス

トランザクションとＩＤのプライバシー保護

メンバーシップサービスの概要



トランザクションとＩＤのプライバシー保護

エンロールメント証明書 Ecerts

– 長期ID

– オフラインでの獲得や持ち込みID (bring-your-own-identity) も可能

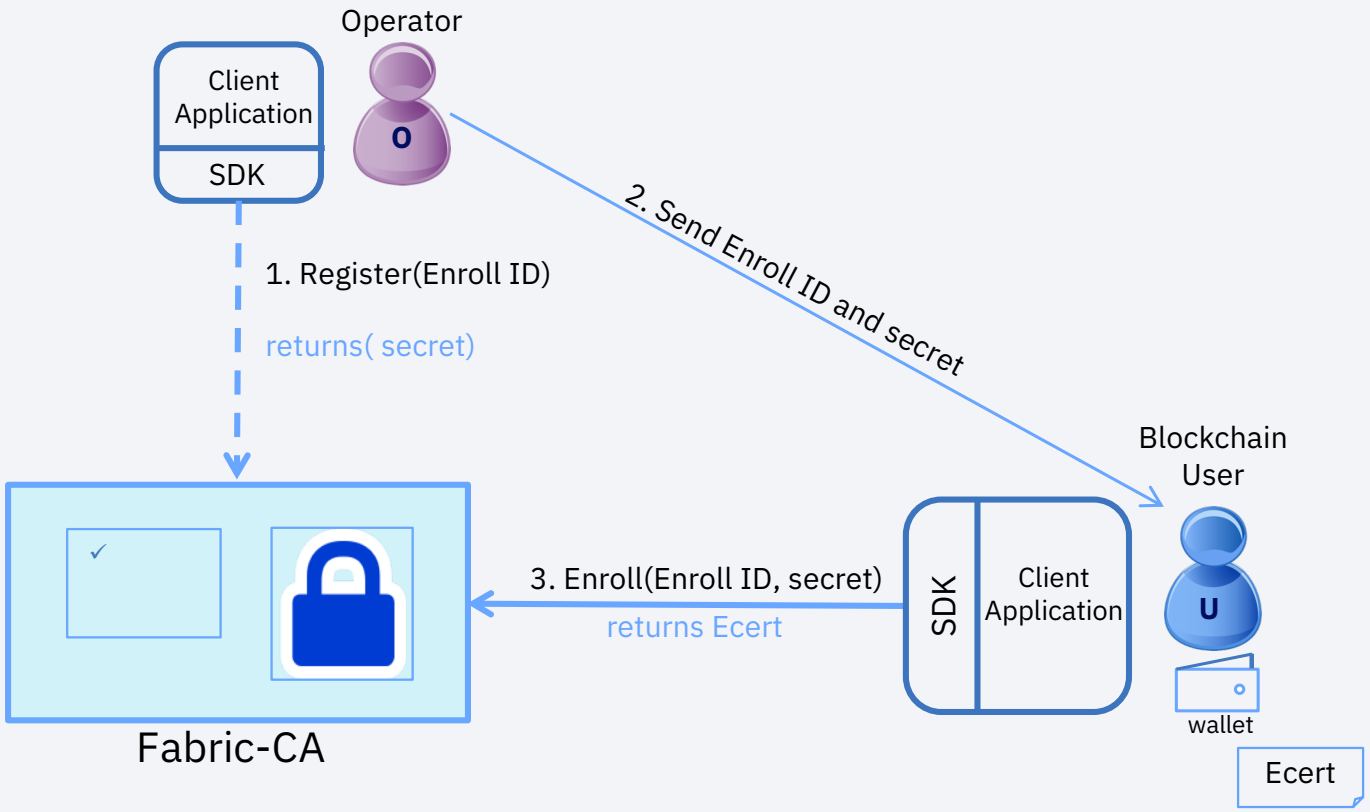
パーミッション型のやり取り

– ユーザーは Ecerts を使って署名する

メンバーシップサービス

– 証明書のプロバイダーへの抽象レイヤー

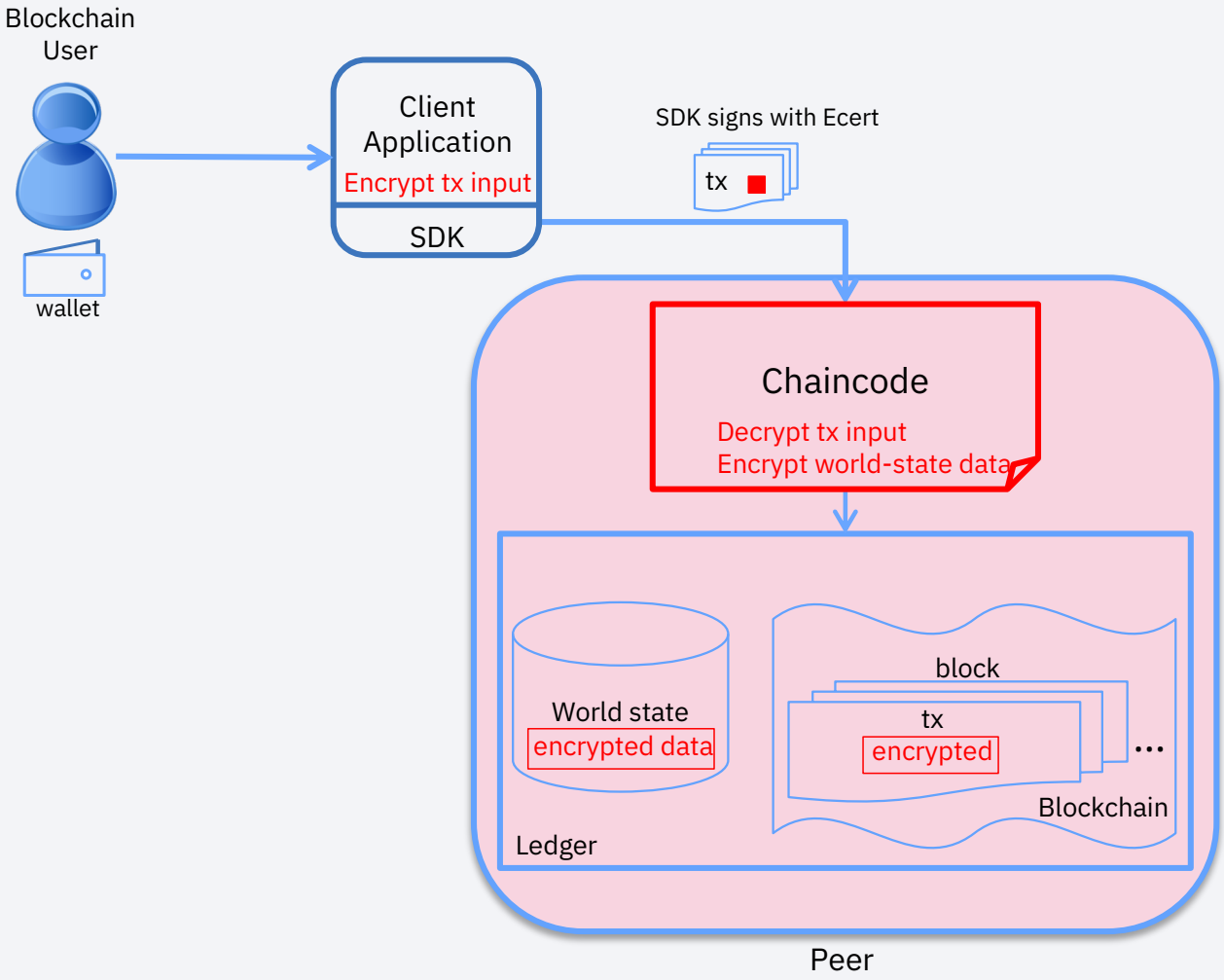
新ユーザーの登録とエンロールメント



登録とエンロールメント

- 管理者は新ユーザーをエンロールメントIDで登録
- ユーザーはエンロールして証明書を受け取る
- 更にオフラインでの登録とエンロールメントのオプションも用意されている

アプリケーションレベルでの暗号化



データの暗号化

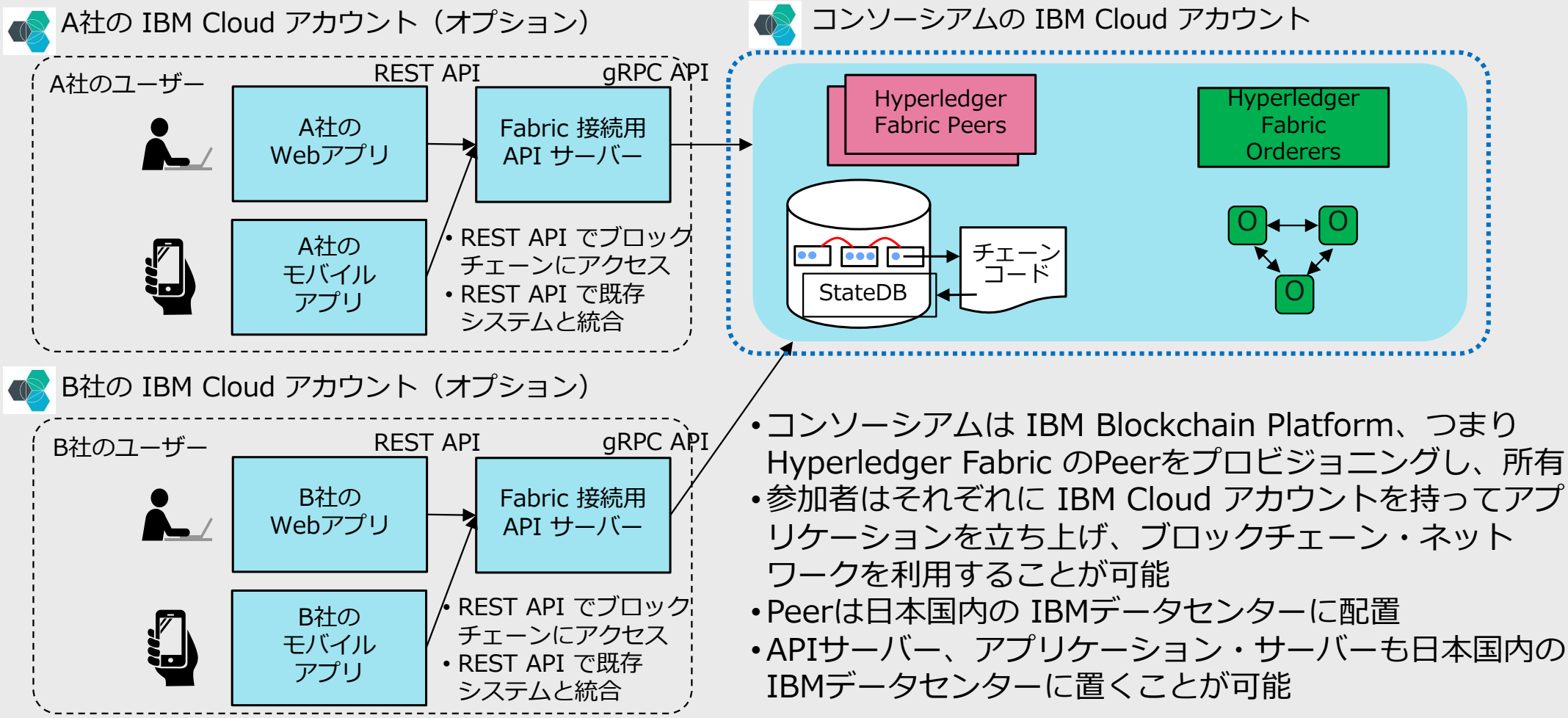
アプリケーション領域での処理

色々な暗号化オプション：
- トランザクションデータ
- チェーンコード*
- ワールドステートデータ

チェーンコードは暗号鍵と一緒にデプロイされたり、クライアントアプリケーションからのトランザクションの一部として *transient* data field (台帳にストアされない)を利用して受けとることが出来る

*アプリケーションチェーンコードの暗号化には追加のシステムチェーンコードの開発が必要になる

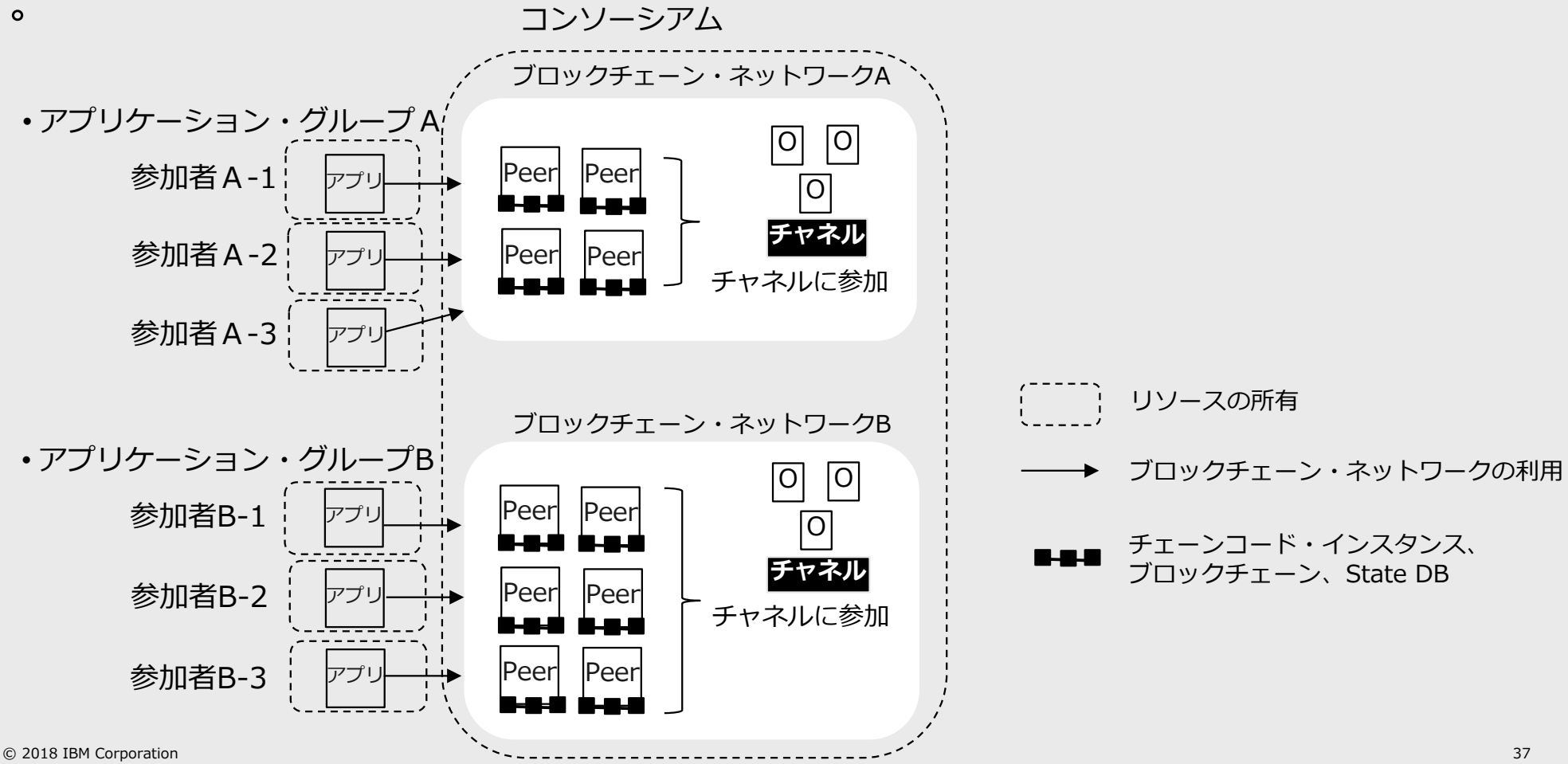
ブロックチェーン・クラウド環境を利用したコンソーシアムの運営例



- コンソーシアムは IBM Blockchain Platform、つまり Hyperledger Fabric のPeerをプロビジョニングし、所有
- 参加者はそれぞれに IBM Cloud アカウントを持ってアプリケーションを立ち上げ、ブロックチェーン・ネットワークを利用することが可能
- Peerは日本国内の IBMデータセンターに配置
- APIサーバー、アプリケーション・サーバーも日本国内の IBMデータセンターに置くことが可能

ブロックチェーン・クラウド環境によるリソースの分離 （パターン１）

システム上に複数のブロックチェーン環境を構築し、環境を分離することが可能です。



ブロックチェーン・クラウド環境によるリソースの分離 （パターン２）

単一のブロックチェーン･ネットワーク環境に、複数のチャネル定義し、環境を分離することが可能です。