

Hyperledger Fabricで構築するビジネスのための ブロックチェーン・システム

Hyperledger Fabric V1 と Composer の最新情報

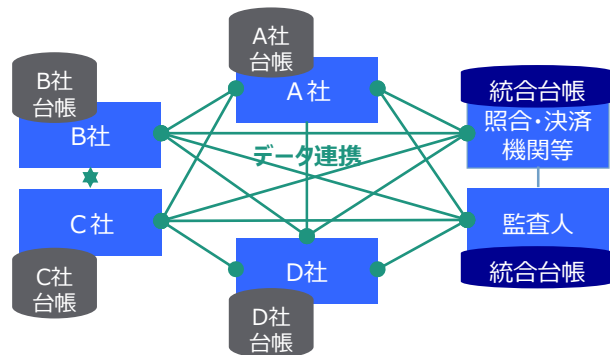
日本アイ・ビー・エム株式会社

クラウド事業本部 ブロックチェーンリーダー

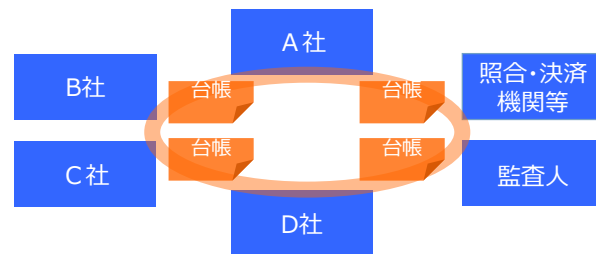
紫関 昭光

ビジネスのためのブロックチェーン

- ブロックチェーンとは、分散し、共有され、複製される台帳
- ビジネス・ネットワーク上の参加者が、同一の記録（台帳）を共有できます
- ビジネス・フローのスピードが高速化し、コストを削減し、リスクを低減し、信頼性が向上します



従来
の台帳管理



非集中・共有型の台帳管理
(Decentralization)

ブロックチェーンの適用のユースケースが多様化

- ビットコインを支える技術として知られたブロックチェーン技術は、各種デジタル資産の保管、資産の所有権移転を安全かつ確実に実施できる技術であると考えられており、各業界で適用が検討されています。

仮想通貨



「ビットコインの国内取引高は2016年1-6月期に4300億円、7月は単月で2000億円を突破」した(2016/8/16 日本経済新聞電子版)。日本国内では、改正資金決済法も整備され、仮想通貨の両替所市場は今後も規模の拡大が予想される。

国際送金



ブロックチェーンによる非集中型の送金サービスにより、透明性やスピードの向上、コストの削減が可能になる。
仮想通貨の両替所を使った海外送金サービスなども可能である。

証券取引



多くの証券取引所で、ブロックチェーンによるポスト・トレーディング処理の効率化など、証券市場の改革が検討されている。
IBMはJPXと共同に低トランザクション市場を想定したブロックチェーン技術の実証実験を行った。

資産管理



不動産登記簿など、ブロックチェーンを資産台帳として使用することで、資産に関するすべてのステークホルダーが、直接その資産の情報を見たり、取引をすることが可能になる。

契約管理



ブロックチェーンによって契約管理ができれば、契約が進行するに従い最新状態が記録され、改ざん不可能な監査証跡が保存される。
ただし、スマートコントラクトを契約書として利用するには、電子署名法など関連法について当局への確認が必要であろう。

保険



保険加入者は、仲介者を介さずにダイレクトに契約を結ぶことができ、保険市場の透明性および正確性が向上する可能性がある。例えば、ロンドン保険市場をブロックチェーンによって改革しようとしている企業もある。

貿易金融



貿易金融には平均12社が関与し、27の文書がやりとりされる複雑なプロセス。
スマートコントラクトによる自動化により、効率性、スピード、正確さが向上し、リスクを低下させる可能性がある。

サプライチェーン マネージメント



複雑なサプライチェーン・ネットワークにまたがる取引をブロックチェーン上に記録することで、プロセスのエンド・ツー・エンドでの可視化・効率化が可能になり、製品の来歴管理が可能になる。

IoT



IBMが提唱する「デバイス・デモクラシー」では、「ブロックチェーンはIoTの世界において相互に連携するデバイス間のトランザクションを管理・調整する枠組みとして非常に有用である。」と謳っている。

本人確認



個人認証の重要性が高まっているが、いまだ運転免許証のコピーを利用するといった方法が行われている。ブロックチェーンを使えば、低コストで安全に本人確認情報を共同所有するしくみが構築できると期待されている。

自立分散型組織



ブロックチェーンは、社会貢献を目的としたソーシャルビジネスのための持続可能なプラットフォームとして利用できる。これにより、全ての人の公平なサービスの提供が期待される。

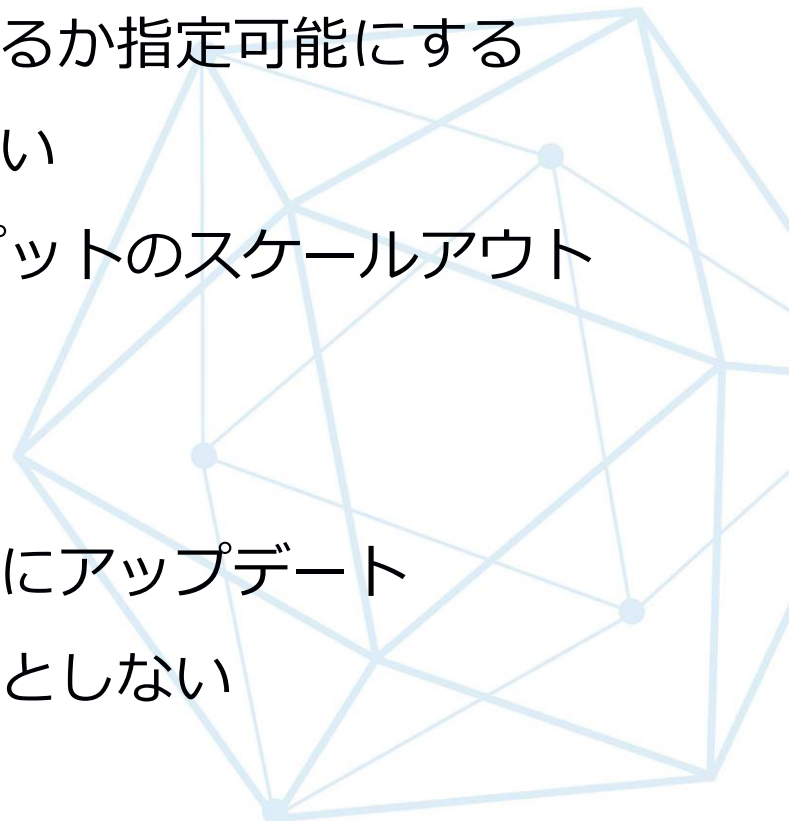
スマートグリッド



中央集権型のシステム構成では、柔軟な分散エネルギー市場の構築は難しい。
小売・送電網自由化のもとでは、ブロックチェーン技術を使った分散エネルギーの市場の構築が期待される。

Hyperledger Fabric V0.x PoC から学んだこと

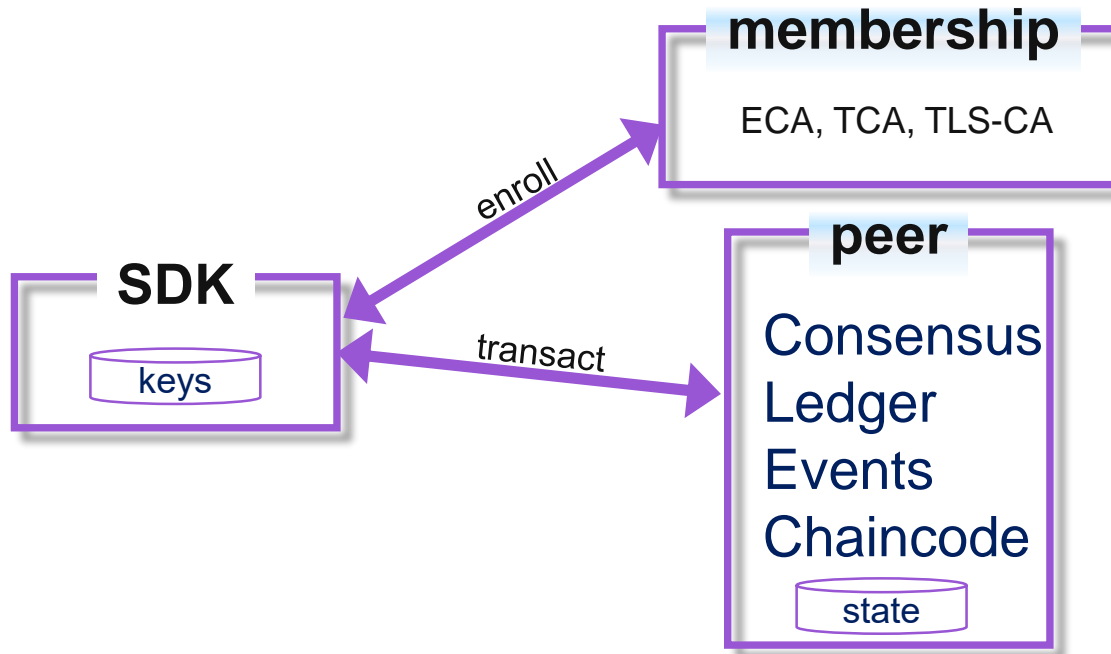
- 誰がトランザクションをエンドースするか指定可能にする
- 取引のデータは関係者だけが持てばよい
- Peer数とトランザクションのスループットのスケールアウト
- 非決定論的なトランザクションの排除
- 台帳への豊富なクエリーをサポート
- ファブリックとチェーンコードを動的にアップデート
- メンバーシップサービスを単一障害点としない



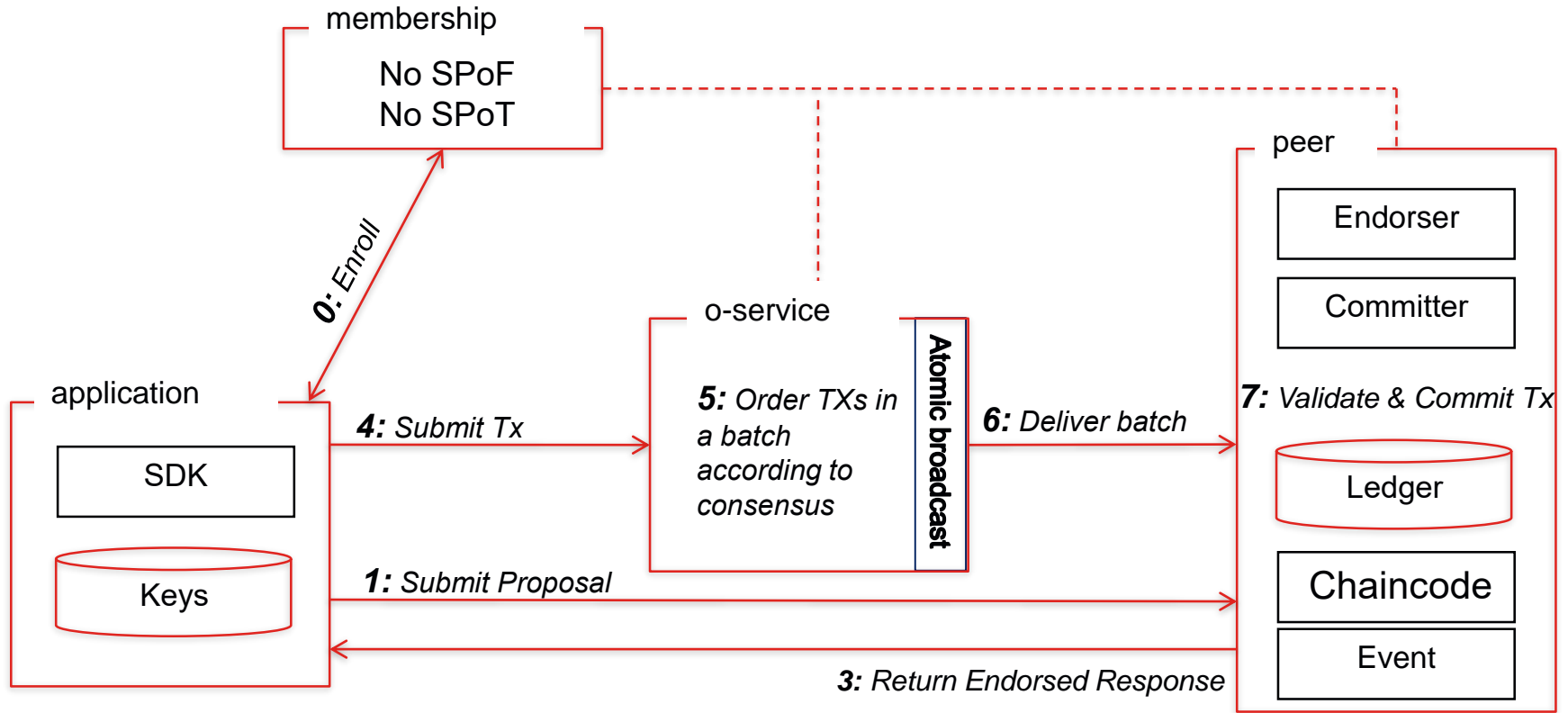
Hyperledger Fabric V1 アーキテクチャ

PoCから学んだビジネス要件を満たすため、V1はアーキテクチャを改善しました

Hyperledger Fabric v0.6



Hyperledger Fabric V1



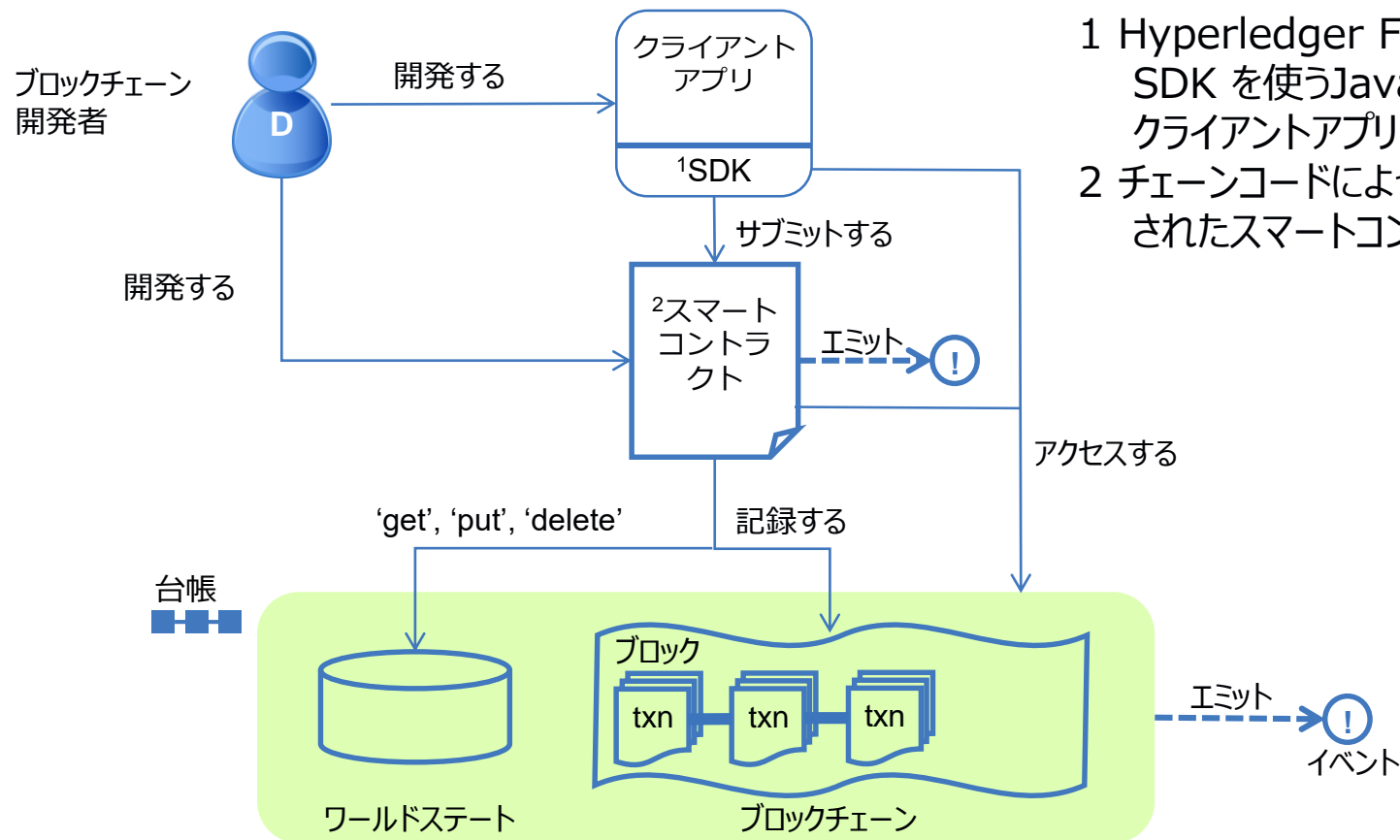
Based on Source : <https://jira.hyperledger.org/browse/FAB-37>

2: Execute CC

アプリケーションから台帳を利用する

アプリケーションからブロックチェーンサービスを利用するには REST APIでなく gRPC API を SDK から利用するようになりました

ブロックチェーンアプリケーションと台帳の概要



- 1 Hyperledger Fabric Client SDK を使うJavaScript クライアントアプリ
- 2 チェーンコードによって実装されたスマートコントラクト




ブロックチェーン・アプリケーション

- アプリケーション
 - ブロックチェーンのビジネスニーズとユーザー・エクスペリエンスに対応
 - 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
- スマートコントラクト
 - 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。
 - コントラクトの開発者はインターフェースを定義する。（例：queryOwner, updateOwner ...）
 - 色々なインターフェースが台帳をアクセスする。Read/Write の一貫性が保たれる。
 - 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。
- 台帳
 - 台帳（ワールドステート）はスマートコントラクトに関する現在値を保持する。
 - 例：vehicleOwner=Daisy
 - ブロックチェーンは全ての台帳の更新履歴を保持する。
 - 例：updateOwner(from=John, to=Anthony); updateOwner (from=Anthony, to=Daisy);etc

ネットワーク・コンセンサス

Validating Peer が Committing Peer, Endorsing Peer, Ordering Node に役割分担されるようになりました

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）を持っていたてもよい。
	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) Propose Transaction








アプリがトランザクションをプロポーズ

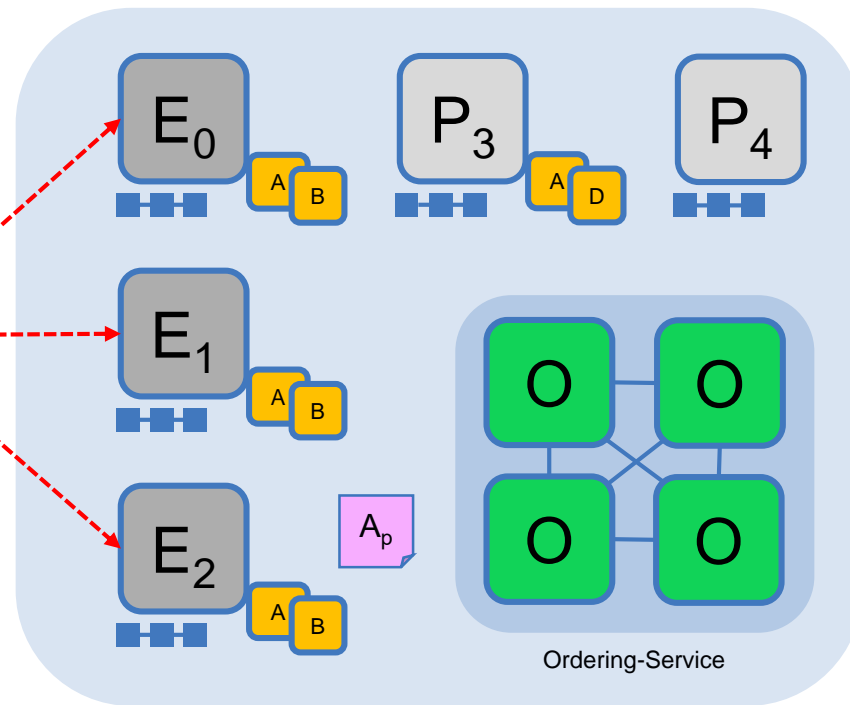
Endorsement policy:

- "E₀, E₁ and E₂ must sign"
- (P₃, P₄ are not part of the policy)

Client application submits a transaction proposal for Smart Contract A. It must target the required peers {E₀, E₁, E₂}

Key:

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



Fabric

トランザクションの例 (2/7) Execute Proposal

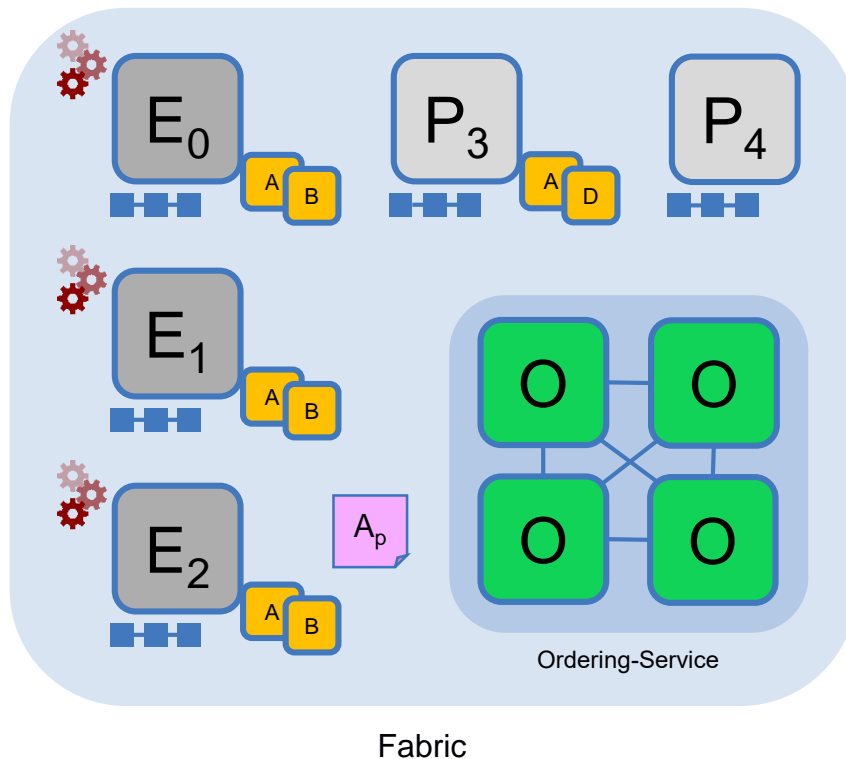
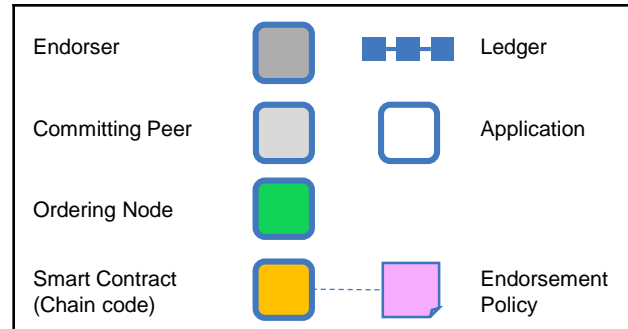
エンドーサーがプロポーザルを実行

E_0 , E_1 & E_2 will each execute the *proposed* transaction. None of these executions will update the ledger

Each execution will capture the set of Read and Written data, called RW sets, which will now flow in the fabric.

Transactions can be signed & encrypted

Key:



トランザクションの例 (3/7) Proposal Response








アプリケーションは応答を受診

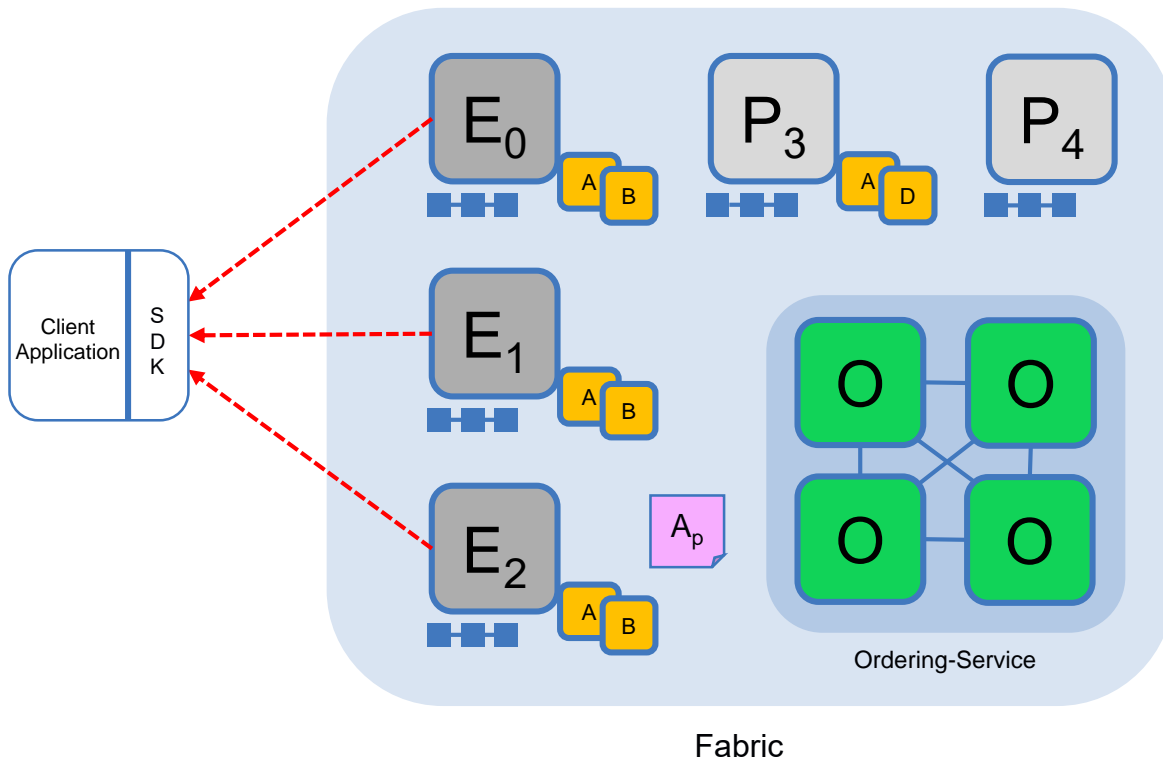
RW sets are asynchronously returned to application

The RW sets are signed by each endorser, and also includes each record version number

(This information will be checked much later in the consensus process)

Key:

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

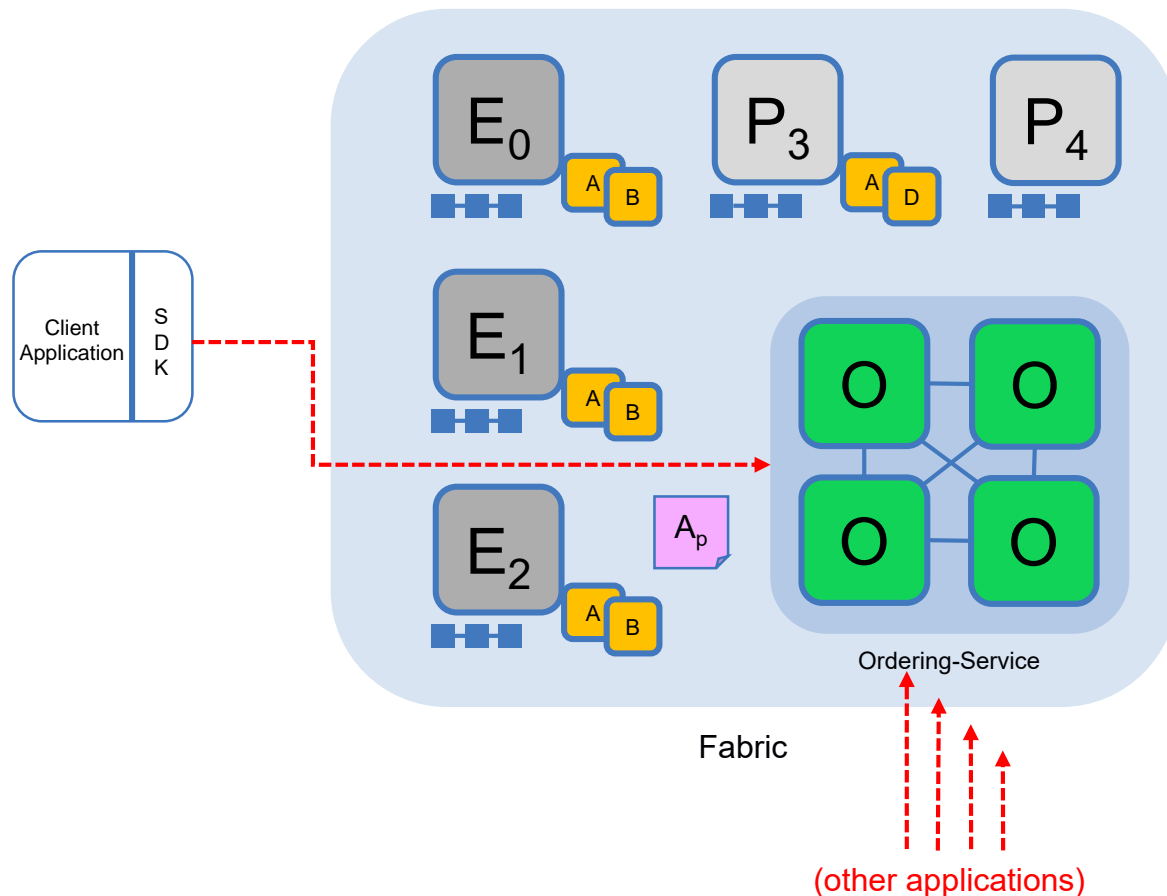


トランザクションの例 (4/7) Order Transaction

アプリは応答をオーダーに送信

Application submits responses as a transaction to be ordered.

Ordering happens across the fabric in parallel with transactions submitted by other applications



Key:

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

トランザクションの例 (5/7) Deliver Transaction








オーダーから全Committing peersへ通知

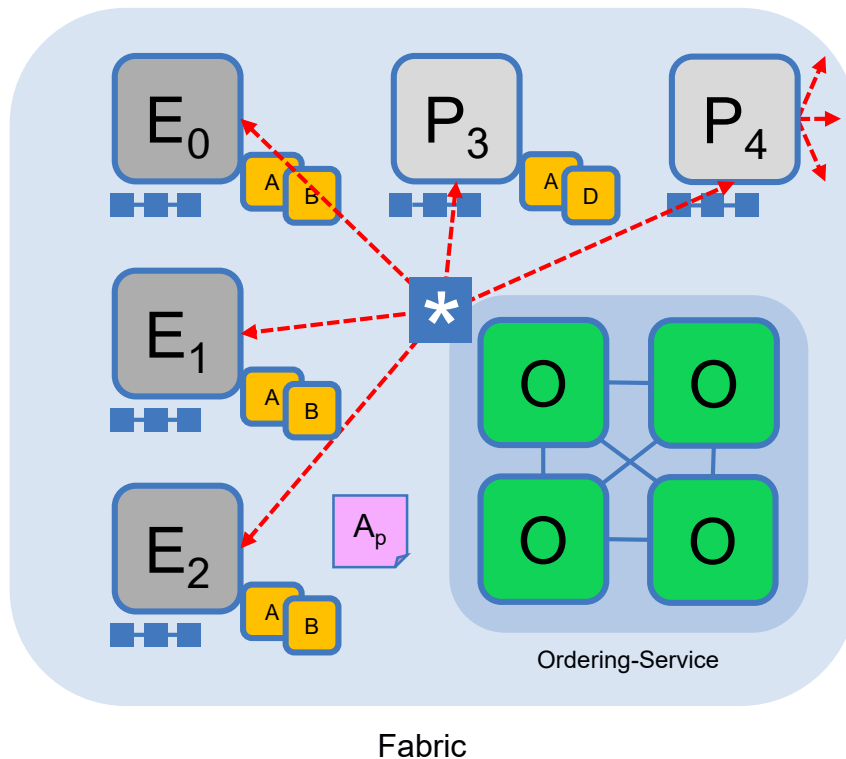
Ordering service collects transactions into proposed blocks for distribution to committing peers. Peers can deliver to other peers in a hierarchy (not shown)

Different ordering algorithms available:

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

Key:

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



トランザクションの例 (6/7) Validate Transaction

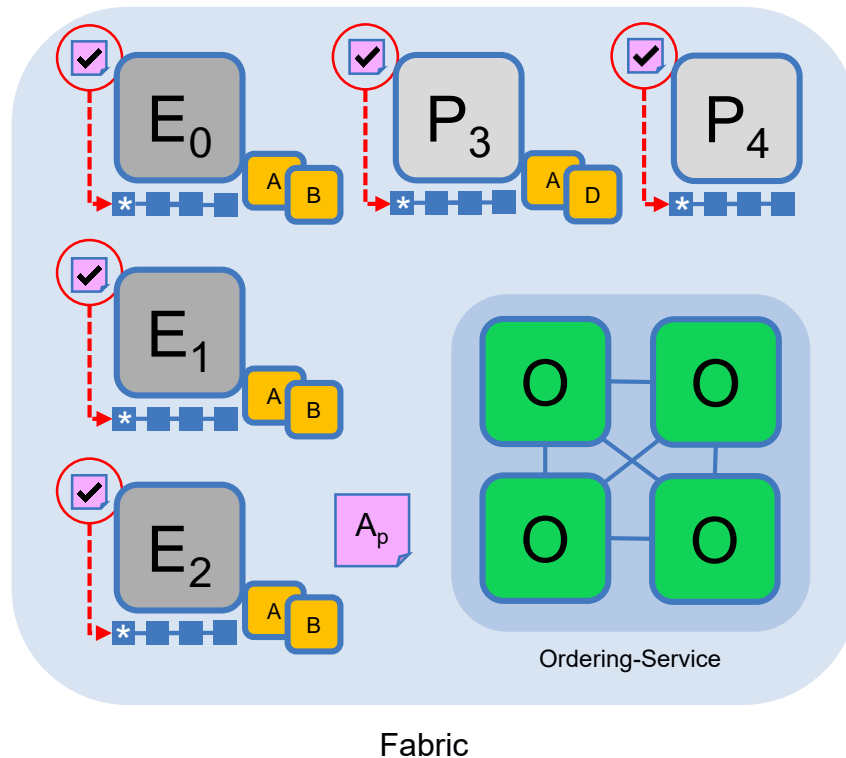
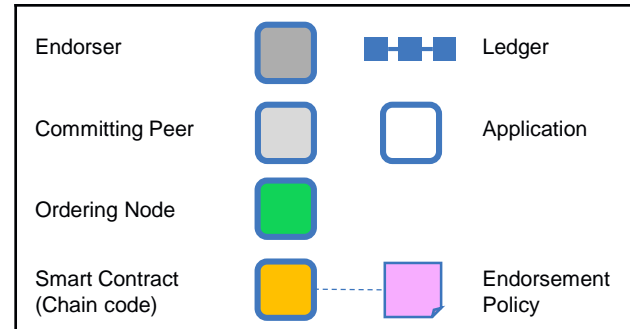
Committing peersはトランザクションを検証

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:

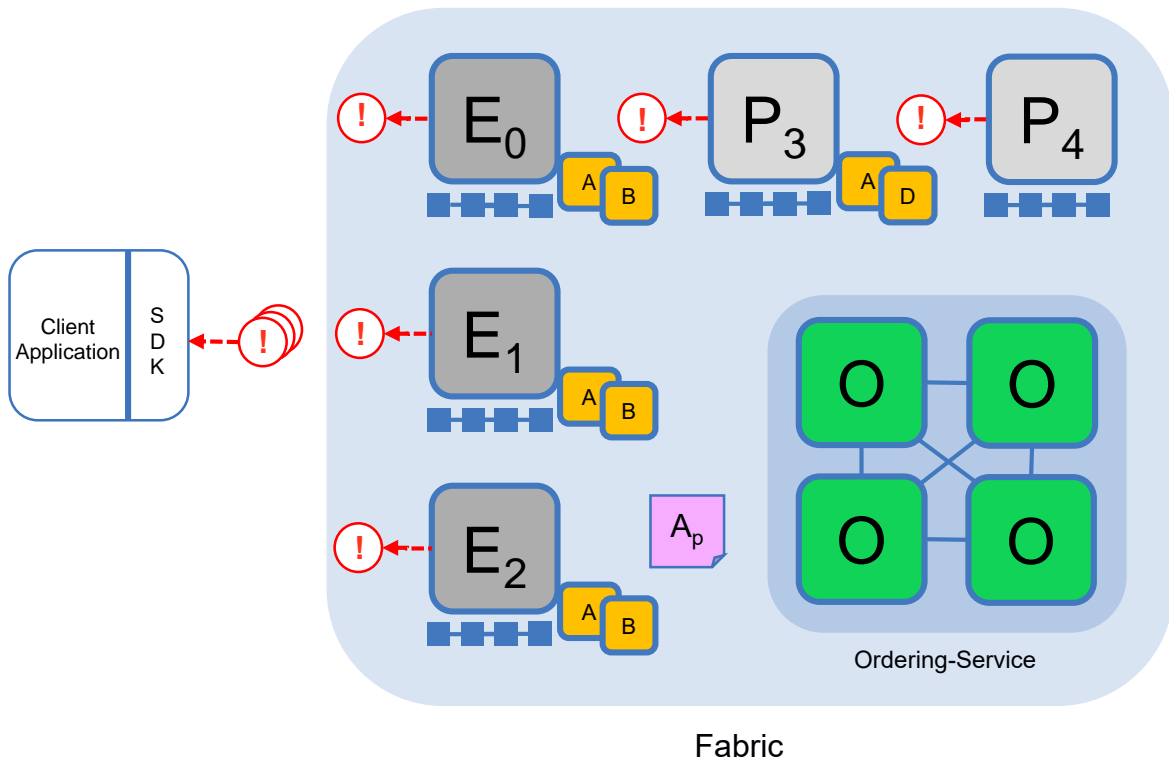


トランザクションの例 (7/7) Notify Transaction

Committing peersはアプリへ通知

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 (Chain code)			Endorsement Policy

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 Work 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

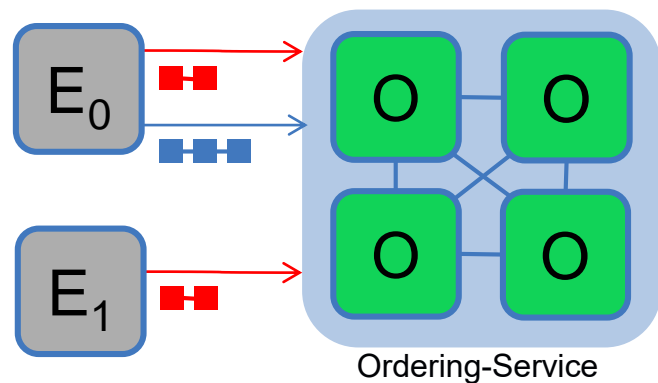
チャンネルとオーダリングサービス

ブロックチェーンネットワークをチャン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ンネルごとに存在します。

チャンネル

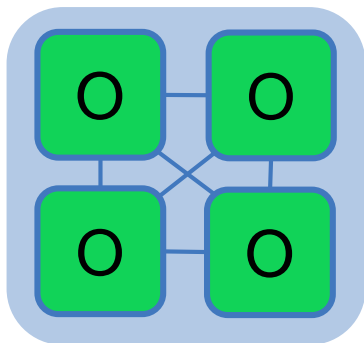
チャンネル毎にトランザクションを異なる台帳で管理



- 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用にチャンネルごとにインスタンス化される
- 台帳はチャン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
- Peerは複数のチャンネルに参加可能
- 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

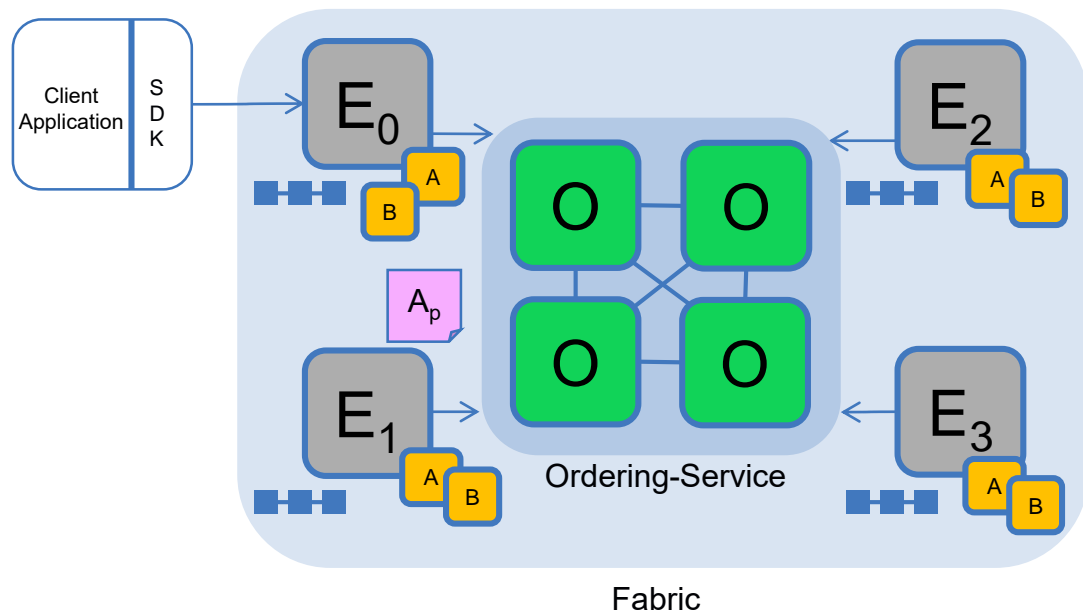
– **Kafka**：クラッシュ・フォルト・トレラント・コンセンサス

- 最小ノード構成 3～n
- 奇数ノードを推奨

– **SBFT**：ビザンチン・フォルト・トレラント・コンセンサス

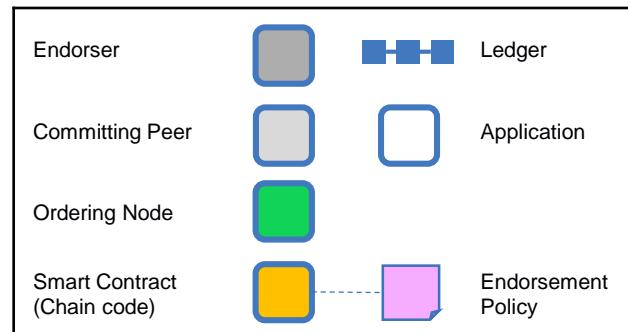
- 最小ノード構成 4～n

一つのチャネルによるエンドースメント

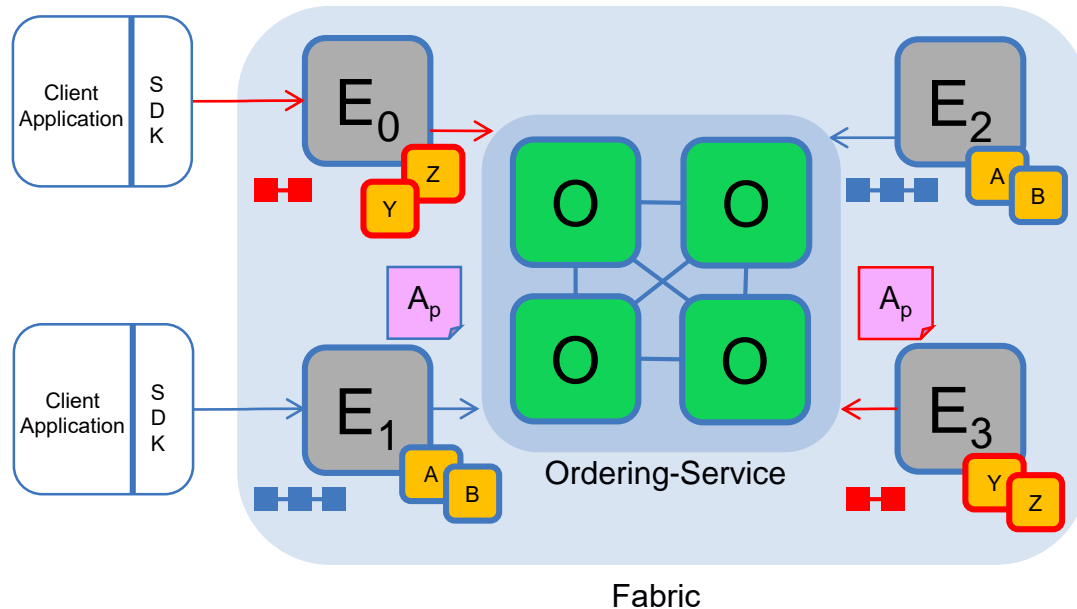


- 0.6 PBFTモデルと同様
- 全てのPeerが同じシステムチャネル（青）にコネクトする
- 全てのPeerは同じチェーンコードと台帳を保持する
- Peers E_0, E_1, E_2, E_3 によるエンドースメント。

Key:



複数チャネルによるエンドースメント



- Peers E_0 , E_3 は赤のチャンネルに接続してチェーンコードY、Zを利用
- Peers E_1 , E_2 は青のチャンネルに接続してチェーンコードA、Bを利用

Key:

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

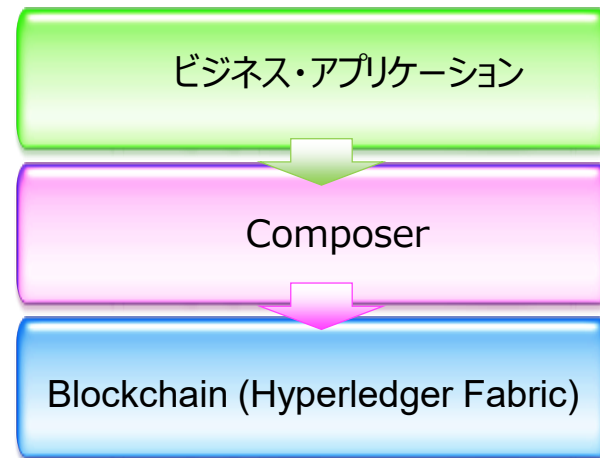
Hyperledger Composer

ブロックチェーン・アプリケーション開発を簡単に、チェーンコードはプログラマーでなくても理解できるように

Composer: Time to Value を加速

IBMからHyperledger Project に寄贈されたオープンソースツールです。ブロックチェーンアプリケーション開発を効率化します。

- ビジネスネットワークのモデリング
- 迅速なアプリケーション開発
- 既存システムとの統合



早い



低リスク

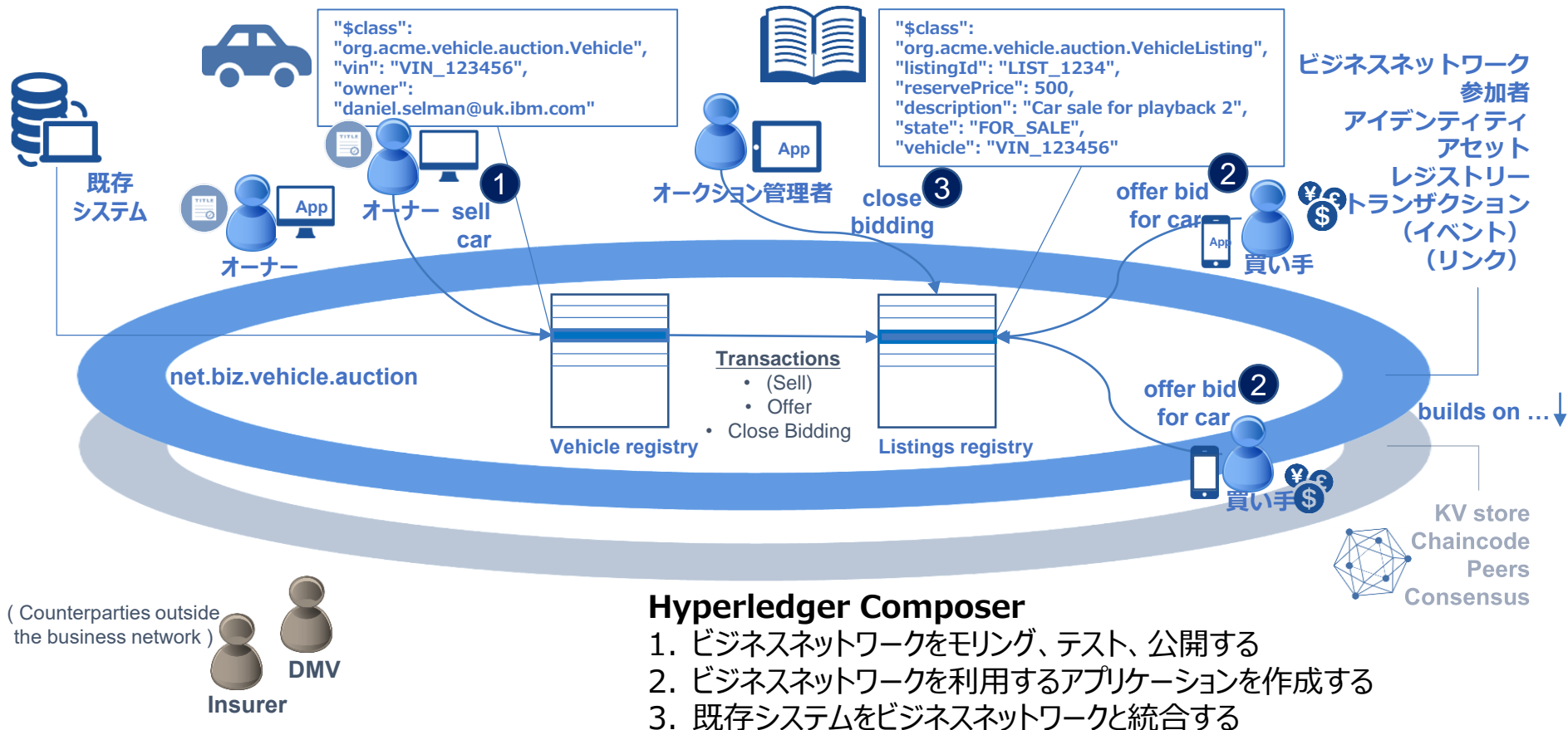


柔軟



理解が容易

ビジネスネットワーク例



ビジネスネットワーク

- ビジネスモデルは**参加者**、**アセット**、**トランザクション**を相互の関係性を含め定義する
 - 表現力あるシンタックス： Arrays, Enumerations, References
 - **アクセス・コントロール・リスト**： 共有とプライバシーのルールを定義する
- **トランザクション・プロセッサー**： 追加のビジネス要件を実装する
 - 標準Javascriptで記述するため開発が容易でポータビリティも高い
- ビジネスネットワークを定義するComposerの構成要素
 - モデル定義ファイル
 - Javascriptプロセッサー
 - Business Network Archivesとしてパッケージ (ease of deployment)

参加者 (Participants)

1 2 4

```
participant Person identified by personId {  
  o String personId  
  o String firstName  
  o String lastName  
}
```

3

Blockchain Composer Editor Assets Participants Transactions

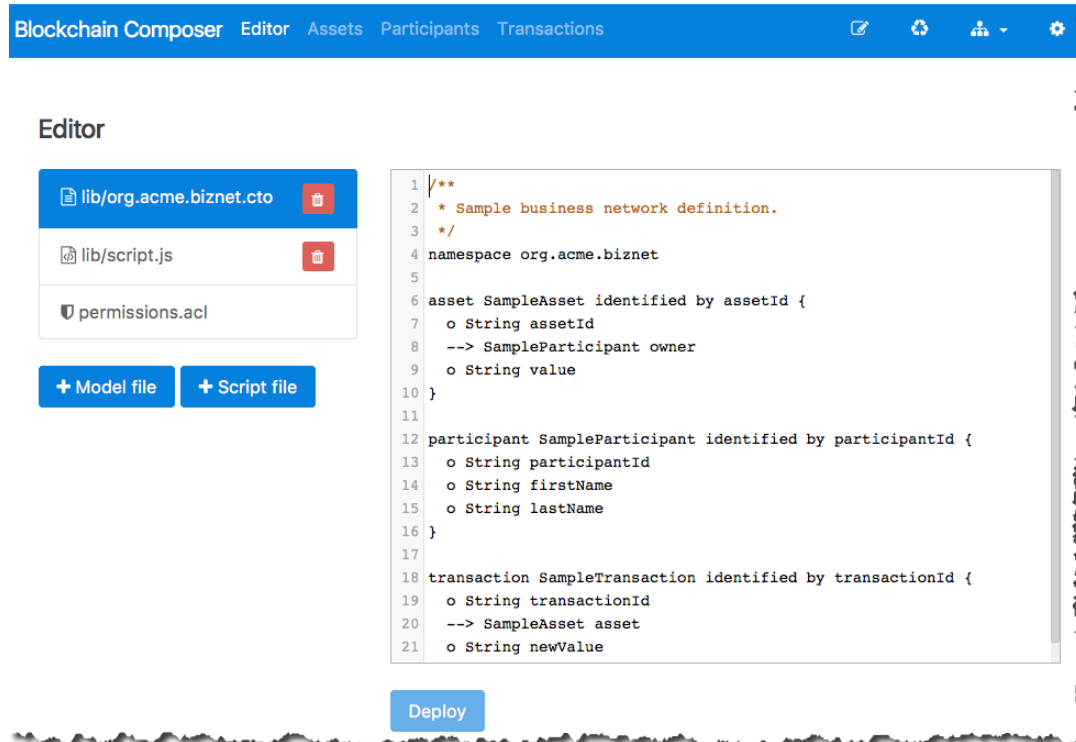
Participant registries

ID	Name
org.acme.biznet.SampleParticipant	Participant registry for org.acme.biznet.SampleParticipant

- 1 参加者を定義
- 2 Participant クラス名
- 3 クラスを定義するデータ構造
'o' で has-a 関係を表現
- 4 キー・フィールド
- 5 レジストリー

開発プレイグラウンド

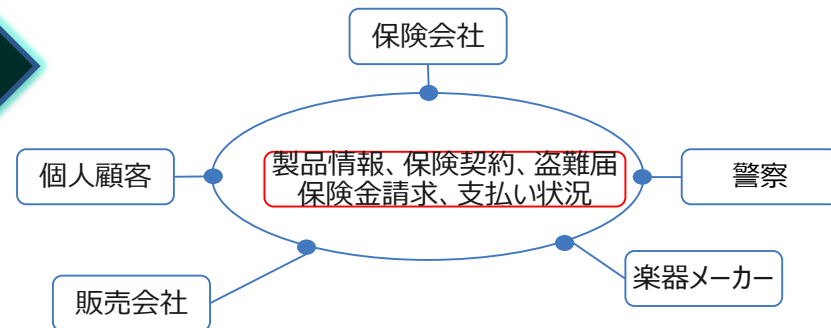
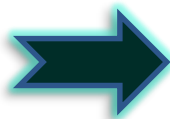
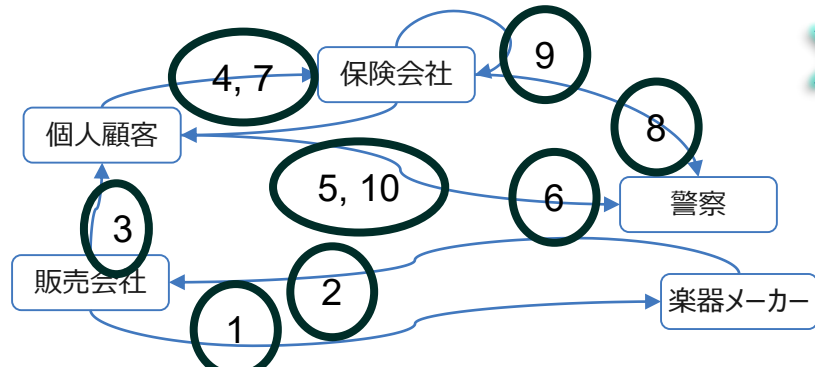
- Hyperledger Composer はアセットを作り、デプロイし、アップデートするためのスタンドアロンのWebベースの開発環境を提供します



サマリーと次のステップ

ユースケースを作り、ブロックチェーンクラウドでアジャイルに開発

ユースケースの作り方



1 販売会社はカスタム・ギターの仕様を提供

2 楽器メーカーから販売会社へ出荷

3 個人顧客がオンラインでギターを購入

4 個人顧客は保険会社から見積もりを取得

5 保険会社は顧客の保険ポリシーにギターを追加

6 3年後、ギターが盗難にあったので警察に被害届を提出

7 保険金を請求

8 保険会社は被害届を確認

9 保険会社は保険金額を決定

10 保険金支払いを通知

1 楽器メーカーは製造時にシリアル番号、製品の特徴を B C に登録

2 販売会社への出荷情報を B C に追加

3 販売会社は入荷情報を B C に追加

4 ギター購入時に支払い情報を B C に追加

5 スマートコントラクトは保険の見積もりを開始

6 個人顧客が保険契約に同意するとその内容を B C に記録

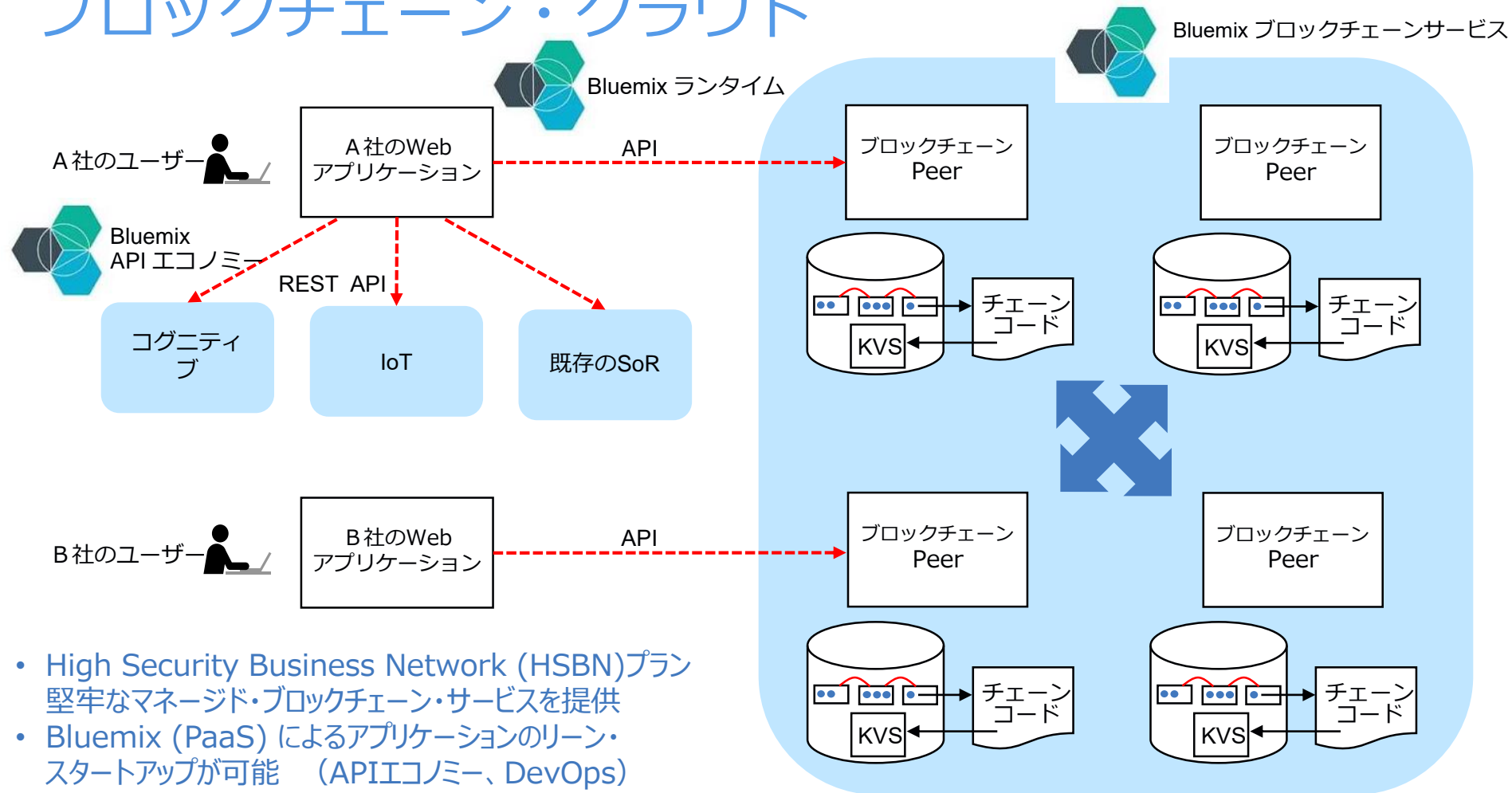
7 毎年、ギターの現在価値を B C に記録

8 個人顧客は盗難届を B C に追加

9 個人顧客は保険金を B C 上で請求、スマートコントラクトは被害届を参照して検証

10 保険金を決定し、B C 上のギターのトラッキング情報を更新

ブロックチェーン・クラウド



- High Security Business Network (HSBN)プラン
堅牢なマネージド・ブロックチェーン・サービスを提供
- Bluemix (PaaS) によるアプリケーションのリーン・
スタートアップが可能 (APIエコノミー、DevOps)

ブロックチェーン関連IBM製品・サービス

- IBM Bluemix (クラウドサービス)
 - ブロックチェーン・サービス - Starter Developerプラン:
Bluemixから Hyperledger Fabric をご提供するマルチテナント・マネージドサービス
 - ブロックチェーン・サービス - High Security Business Network (HSBN)プラン:
日本IBMの国内データセンター等の LinuxONEからお客様専用のHyperledger Fabric をご提供するシングルテナント・マネージドサービス
- IBM Bluemix Garageサービス (支援サービス)
 - Bluemix Garage Method によるアジャイル DevOps 支援サービス
- Hyperledger Docker イメージ (カスタマーサポート・サービス)
 - IBMがテスト、検証済みの Hyperledger Fabric Docker イメージのカスタマーサポート・サービス

次のステップ



- ブロックチェーンはまだ始まったばかりです！
- 共有台帳とスマートコントラクトを皆さんのビジネスネットワークに応用してみましよう
- 参加者 (Participants)、アセット(Assets)、ビジネスプロセスについて考えてみましよう
- 現実的なビジネス・ユースケースを時間をかけて描いてみましよう
- テクノロジーに手を触れて体験してみましよう
- 概念実証 (PoC) の最初のプロジェクトを始めませんか
- IBMはお客様のブロックチェーン・プロジェクトをご支援します

クラウド&コグニティブの世界へ

みなさまのアイデアをかたちにするプラットフォーム
Bluemix 30日間無料トライアルでご体験ください

➡ **bluemix.net** (Web ブラウザーからご利用いただけます)

✓ **今すぐ始められる**

クレジット・カード情報不要 - アカウント作成後、すぐご利用頂けます

✓ **十分なリソースを提供**

30 日間無料で 2GB のランタイムおよびコンテナ・メモリーが利用可能
最大 10 個のサービスのプロビジョンにアクセスできます

✓ **相談窓口**

お困りの際は、無料のヘルプ・デスク・サポートをご利用ください

ワークショップ、セッション、および資料は、IBMまたはセッション発表者によって準備され、それぞれ独自の見解を反映したものです。それらは情報提供の目的のみで提供されており、いかなる参加者に対しても法律的またはその他の指導や助言を意図したのではなく、またそのような結果を生むものでもありません。本講演資料に含まれている情報については、完全性と正確性を期するよう努力しましたが、「現状のまま」提供され、明示または暗示にかかわらずいかなる保証も伴わないものとします。本講演資料またはその他の資料の使用によって、あるいはその他の関連によって、いかなる損害が生じた場合も、IBMは責任を負わないものとします。本講演資料に含まれている内容は、IBMまたはそのサプライヤーやライセンス交付者からいかなる保証または表明を引きだすことを意図したもので、IBMソフトウェアの使用を規定する適用ライセンス契約の条項を変更することを意図したものでなく、またそのような結果を生むものでもありません。

本講演資料でIBM製品、プログラム、またはサービスに言及していても、IBMが営業活動を行っているすべての国でそれらが使用可能であることを暗示するものではありません。本講演資料で言及している製品リリース日付や製品機能は、市場機会またはその他の要因に基づいてIBM独自の決定権をもっていつでも変更できるものとし、いかなる方法においても将来の製品または機能が使用可能になると確約することを意図したものではありません。本講演資料に含まれている内容は、参加者が開始する活動によって特定の販売、売上高の向上、またはその他の結果が生じると述べる、または暗示することを意図したもので、またそのような結果を生むものでもありません。パフォーマンスは、管理された環境において標準的なIBMベンチマークを使用した測定と予測に基づいています。ユーザーが経験する実際のスループットやパフォーマンスは、ユーザーのジョブ・ストリームにおけるマルチプログラミングの量、入出力構成、ストレージ構成、および処理されるワークロードなどの考慮事項を含む、数多くの要因に応じて変化します。したがって、個々のユーザーがここで述べられているものと同様の結果を得られると確約するものではありません。

記述されているすべてのお客様事例は、それらのお客様がどのようにIBM製品を使用したか、またそれらのお客様が達成した結果の実例として示されたものです。実際の環境コストおよびパフォーマンス特性は、お客様ごとに異なる場合があります。

IBM、IBM ロゴ、ibm.com、Bluemix は、世界の多くの国で登録されたInternational Business Machines Corporationの商標です。他の製品名およびサービス名等は、それぞれIBMまたは各社の商標である場合があります。現時点での IBM の商標リストについては、www.ibm.com/legal/copytrade.shtmlをご覧ください。

Linuxは、Linus Torvaldsの米国およびその他の国における登録商標です。

JavaおよびすべてのJava関連の商標およびロゴは Oracleやその関連会社の商標または登録商標です。