Fabric Gateway – current (alpha) Endorse logic

- The Endorse() API method takes a ProposalRequest from the client, gathers an adequate number of endorsements, builds a transaction envelope which it returns to client for signing and ordering.

- Currently, the logic for selecting endorsing peers depends on whether WithEndorsingOrganizations() is specified:
  - If specified, the discovery PeersOfChannel() method is used to select one peer from each of the specified Orgs (that has the chaincode installed). No other peers will be selected. If the client hasn't specified the correct Orgs, this will probably do the wrong thing.

  - If not specified, the discovery PeersForEndorsement() method is used to select an endorsement plan. It currently has no knowledge of collections or SBE keys. If the chaincode deals with those, this will probably do the wrong thing.

- Bottom line – the client app developer must know exactly what the chaincode does and specify the endorsing orgs manually if necessary.

IBM

Collections – using the legacy SDKs

- The discovery service can only derive the correct endorsement layouts if the client request contains 'hints' as to which collections are accessed by each chaincode invocation. It can then look up the policies for each collection.
- This hint is in the form of a ChaincodeInterest structure and can be assigned by the app developer to a contract object.
- But this is not a trivial structure to understand and can get unwieldy for transactions that touch multiple collections.
- It can also be different for each function within a chaincode, something that is not modelled well in the existing SDKs.
- Bottom line – use of collections might narrow the number of allowable Orgs that can endorse a transaction proposal (INTERSECTION / SUBTRACTIVE)

State-based endorsement – using the legacy SDKs

- With SBE, signature policies can be applied to individual keys (states) in during the execution of the chaincode.
- It overrides the chaincode policy for those states and gets checked at validation stage (all must pass).
- It is written into the read-write set and not the configuration, so can't be accessed by the discovery service at all.
- The client app is responsible for specifying the endorsing Orgs to satisfy the chaincode policy and ALL the SBE states that get written to.
- Bottom line – use of SBE might expand the number of endorsing Orgs to cover all possible state-based endorsement policies (UNION / ADDITIVE).

Combining collections and SBE

- If a chaincode invocation touches both collections and SBE states, then the set of endorsing orgs required must be the intersection of:
  - The set of endorsement layouts from the discovery service using the ChaincodeInterest
  - The minimum set of orgs (MSP Principals) that will satisfy all of the SBE policies.
- Any more than that might fail in the chaincode simulation (unauthorized access to a collection).
- Any fewer than that will fail in the validation phase.
- The existing SDKs don't help with this, the client app developer must specify the set of endorsing Orgs manually.

Private data and the Transient field

- The Transient field of a transaction proposal is designed to carry sensitive data, since it doesn't get written to the ledger.
- It is an established pattern that private data gets sent in the transient field, then written to a private data collection in the chaincode.
- We *could* use the endorsement policy derived from the collections hints (ChaincodeInterest) to decide which Orgs to send the transient data.
- However, that could have unintended consequences, especially for collections that allow 'blind writes'.
- I think there always needs to be a manual mechanism where the app developer can guarantee which orgs will receive the request
  - Better an endorsement failure than to leak sensitive data.

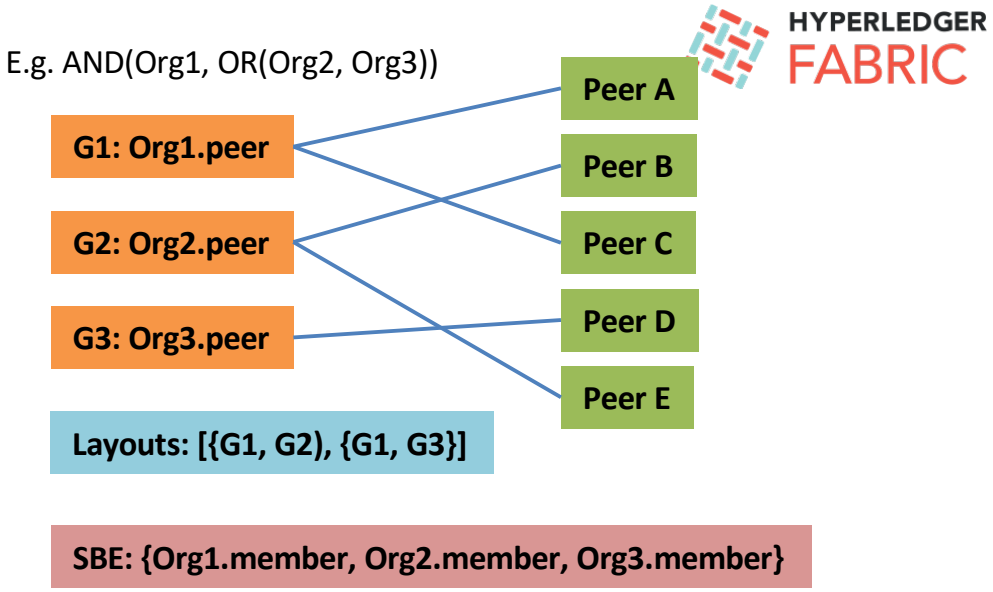
IBM

Proposal for Gateway logic

- *Where possible, automatically derive the ChaincodeInterest structure, SBE policies, and combine them in the Gateway Endorse() method without requiring hints from the client application.*

1. Endorse the proposal on a peer in the gateway's org (could be the host peer).
   - The tx_simulator / query_executor is enhanced to collect information to populate ChaincodeInterest and SBE writes (details later).
   - The ProposalResponse proto is augmented with the ChaincodeInterest and SBE policies for this TX simulation.
2. The discovery PeersForEndorsement() method is invoked passing this ChaincodeInterest.
3. The set of SBE policies are combined into a single merged policy.
4. The set of layouts from step 2 is augmented within the constraints of the collection policies to satisfy the merged SBE policy (future enhancement – move this logic into the discovery service?).
   - If this is an empty set, return error – endorsement can't be satisfied.
5. A layout that includes the endorsement already gathered in step 1 is selected and the remaining peers are invoked.
6. The resultant endorsements are combined into a EndorserResponse and returned to the client.

Endorser selection logic



Select peer from gateway org
- Using endorsersByOrg() / PeersOfChannel()

Invoke proposal
- Response contains ChaincodeInterest & SBEs

Get endorsement plan from discovery
- Based on derived ChaincodeInterest

	No SBEs	SBEs
No Collections	Select a layout from discovery PeersForEndorsement().	Select a layout from discovery PeersForEndorsement(). Add extra orgs' peers as required by SBE policies.
Collections	Select a layout from discovery PeersForEndorsement().	Select a layout from discovery PeersForEndorsement(). Add extra orgs' peers as required by SBE policies *only if those orgs are represented by groups in the endorsement plan*.

Enhancement of endorser logic

- The transaction simulation logic is performed in fabric/core/ledger/kvledger/txmgmt/txmgr/
  - tx_simulator.go & query_executor.go
  - This adds to the read/write set (public & private collections) for each invocation of stub functions

- Currently, the private read set for collections is not captured.
  - There is no way to distinguish GetPrivateData() and GetPrivateDataHash() in the current RWset.
  - We need this to derive the 'NoPrivateReads' boolean in the ChaincodeInterest.
  - Proposal to add the Private read set to the rwset_builder object.

- In the tx_simulator structure, collect the set of SBE signature policies.
  - Whenever SetState()/SetStateMetadata() is invoked, check if a signature policy exists in the metadata for that state. If it does, add the policy to the set.
  - GetTxSimulationResults() can add these to the simResults to pass back to Endorser.SimulateProposal()

- Endorser.SimulateProposal() converts these to a ChaincodeInterest structure and adds to the ProposalResponse
  - The private read set is discarded

- The set of SBE policies is also added to the ProposalResponse

Private/Transient data

HYPERLEDGER FABRIC

- This proposal removes the need for the client app to worry about which collections and SBE states are touched by the chaincode.

- But it doesn't remove the need to worry about sensitive data.

- Current proposal is
  - **If** Endorse() is invoked with Transient data, but without explicitly specifying the endorsing orgs...
  - **then** the proposed logic is followed down to step 4 (on slide 6)...
  - **but** if the single local endorsement is not enough to satisfy an endorsement layout...
  - **then** an error is returned to the client saying that they will need to explicitly set the endorsing orgs.

IBM

ProposalResponse protobuf

```
message ProposalResponse {
  // Version indicates message protocol version
  int32 version = 1;

  // Timestamp is the time that the message was created as  defined by the sender
  google.protobuf.Timestamp timestamp = 2;

  // A response message indicating whether the endorsement of the action was successful
  Response response = 4;

  // The payload of response.
  // It is the bytes of ProposalResponsePayload
  bytes payload = 5;

  // The endorsement of the proposal, basically the endorser's signature over the payload
  Endorsement endorsement = 6;

  // The chaincode interest derived from simulating the proposal.
  ChaincodeInterest interest = 7;
}
```

Only the payload gets signed (endorsed) and written to the ledger. No other fields.

ChaincodeCall protobuf

```
// ChaincodeInterest defines an interest about an endorsement
// for a specific single chaincode invocation.
// Multiple chaincodes indicate chaincode to chaincode invocations.
message ChaincodeInterest {
   repeated ChaincodeCall chaincodes = 1;
}

// ChaincodeCall defines a call to a chaincode.
// It may have collections that are related to the chaincode
message ChaincodeCall {
   string name = 1;
   repeated string collection_names = 2;
   bool no_private_reads = 3; // Indicates we do not need to read from private data
   bool no_public_writes = 4; // Indicates we do not need to write to the chaincode namespace

   // The set of signature policies associated with states in the write-set that have state-based endorsement policies.
   repeated common.SignaturePolicyEnvelope key_policies = 5;
}
```

Questions