



# Welcome !

---



## Fabric Application Developer Community Call

Thurs September 17<sup>th</sup> 2020



# Agenda

---



## Today's Agenda :

- Fabric Gateway: prototype demo / preview - Andy Coleman (IBM)

RFC: <https://github.com/andrew-coleman/fabric-rfcs/blob/master/text/0000-fabric-gateway.md>

- Fabric-Samples: State based endorsement(SBE): presentation/demo – Paul O'Mahony (IBM)
- AOB

Note: Any resource links usually shared on:

<https://wiki.hyperledger.org/display/fabric/Resources%3A+Fabric+App+Developer+Community>

State Based Endorsement (SBE):

Fabric Samples

## RECAP: Chaincode Based vs State Based Endorsement (SBE)

- Endorsement *policies* specifies the set of organizations whose peers need to endorse a transaction before it can be added to the ledger.
- Each chaincode that is deployed to a channel has an endorsement policy that governs the assets managed by the chaincode smart contract(s)
- You can *override* the chaincode level endorsement policy to create an endorsement policy for a specific *key*, either on the public channel ledger or - in a private collection. *State* overrides *Collection level* (if using PvtData) overrides *chaincode level* generally speaking
- *State-based* endorsement policies, also known as *key-level* endorsement policies, allow channel members to use different endorsement policies for assets that are managed by the same smart contract
- Fabric Samples has a state-based endorsement (SBE) asset transfer sample
  - demonstrates how to use key-level endorsement policies
  - ensure that an asset only is endorsed by an asset owner (scenario follows what happens when Org1 is owner, then Org 2 becomes owner).
  - Sample uses discovery for chaincode-level endorsement

Chaincode Endorsement Policy:



Org1, Org2, Org3

State-Based Endorsement



World State

<u>Key</u> "CAR001"
<u>Value</u> "Model" = "Nova" "Colour" = "Blue" "Owner" = "Tom"

<u>Key</u> "CAR002"
<u>Value</u> "Model" = "Thanos" "Colour" = "Yellow" "Owner" = "Jill"

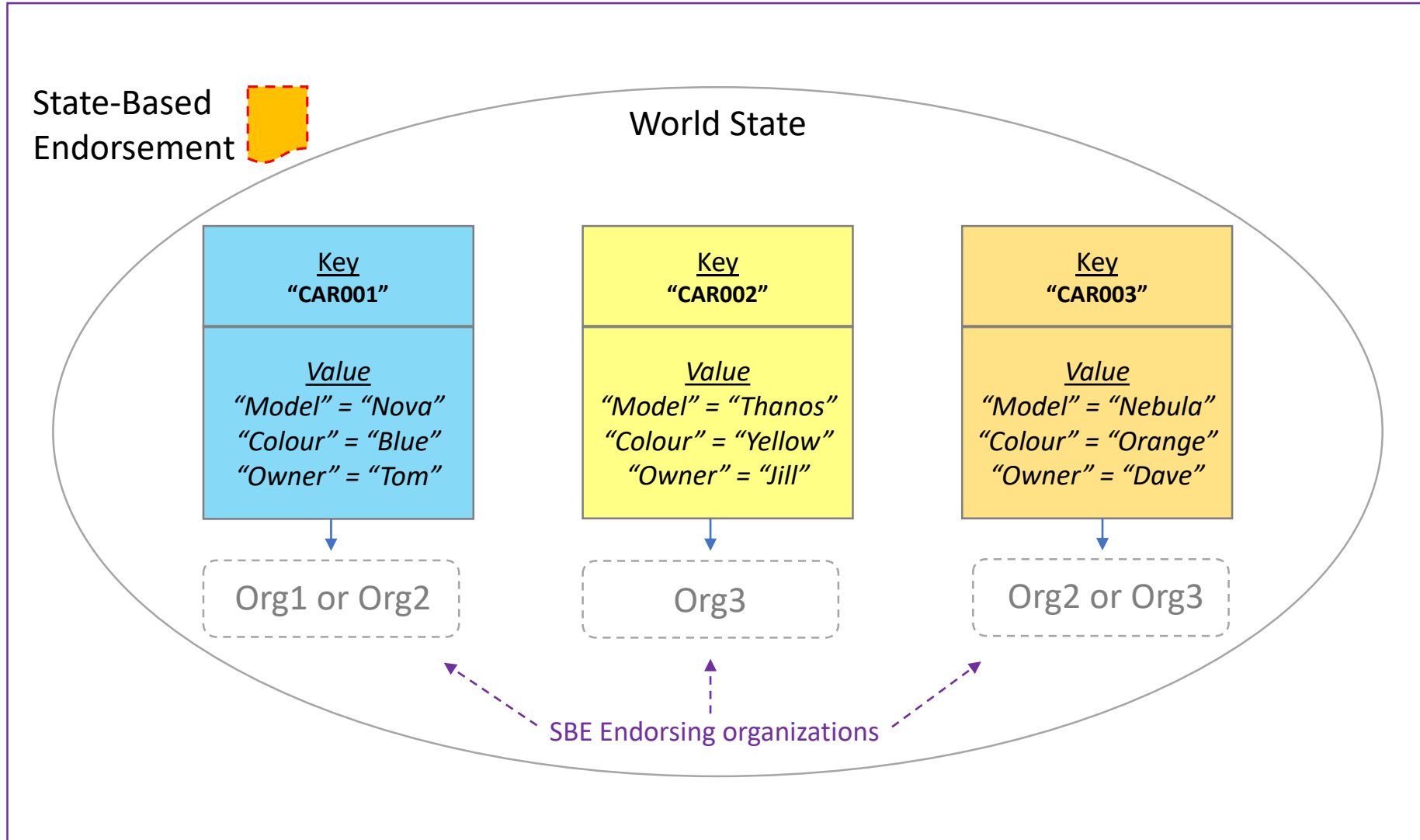
<u>Key</u> "CAR003"
<u>Value</u> "Model" = "Nebula" "Colour" = "Orange" "Owner" = "Dave"

Org1 or Org2

Org3

Org2 or Org3

SBE Endorsing organizations



# SBE Scenario

Chaincode Endorsement Policy:



Org1, Org2

State-Based Endorsement



World State

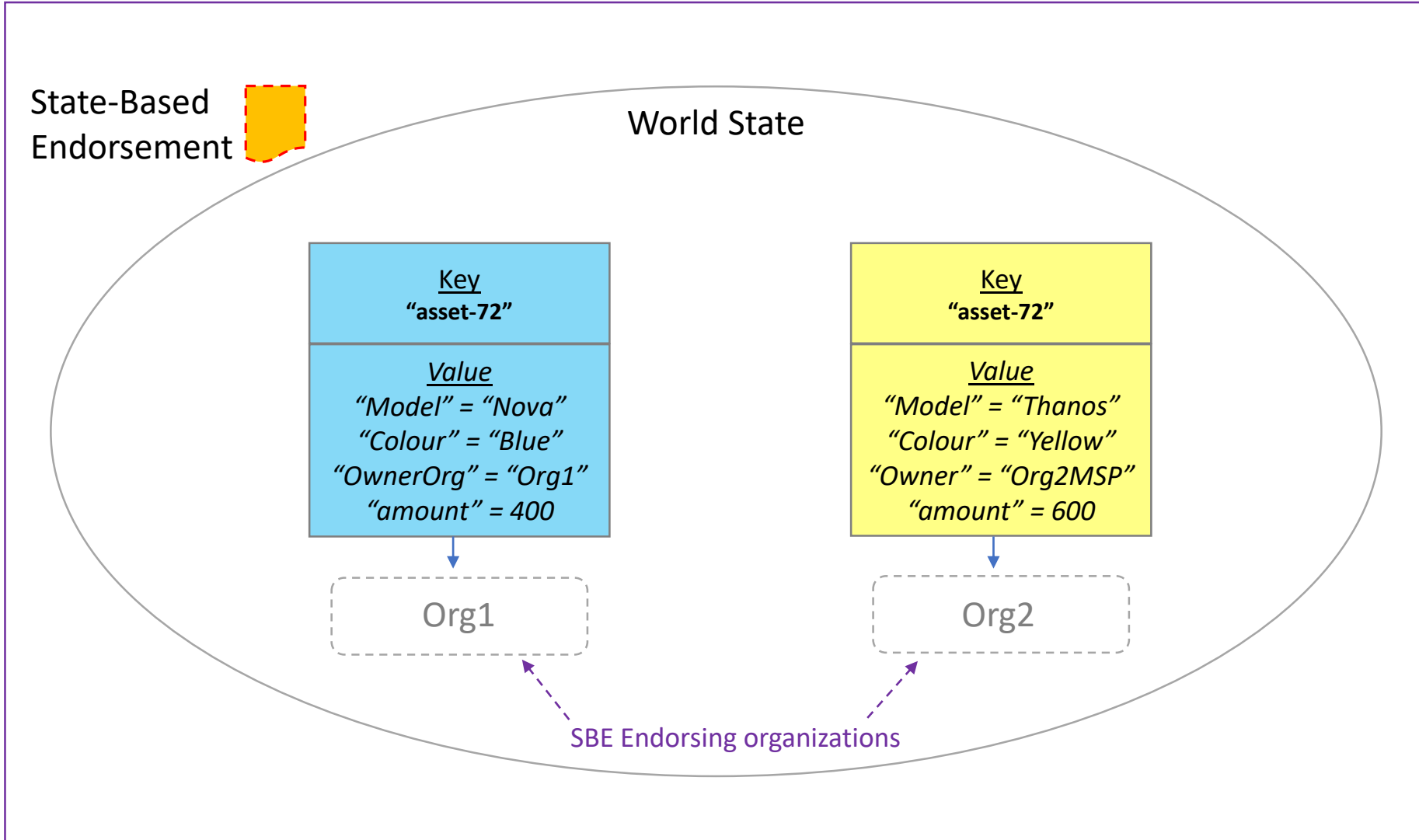
Key
"asset-72"
Value
"Model" = "Nova"
"Colour" = "Blue"
"OwnerOrg" = "Org1"
"amount" = 400

Key
"asset-72"
Value
"Model" = "Thanos"
"Colour" = "Yellow"
"Owner" = "Org2MSP"
"amount" = 600

Org1

Org2

SBE Endorsing organizations



## SBE: Sequence of events

Org transacting	Transaction	Endorsement Type (CC or SBE)	Requested Endorsing Orgs sent to	Results	Current Owner / Asset Value
Org1	Create_asset	CC	All Orgs	Succeeds	Org1 / 100
Org1	Update Asset	SBE	Org1 only	Succeeds	Org1 / 200
Org1	Update Asset	SBE	All Orgs	Succeeds, when Org1 endorses (Org2 endorses, but not required)	Org1 / 300
Org2	Update Asset	SBE	All Orgs	Succeeds, when Org 1 endorses	Org1 / 400
Org2	Update Asset	SBE	Org2 only	<b>FAILS</b> – not owning org – remains at 400 (not 500)	Org1 / 400
Org1	Transfer Asset	SBE	Org1 only	Succeeds – change owner to Org2	Org2 / 400
Org2	Update Asset	SBE	Org2 only	Succeeds	Org2 / 600
Org1	Update Asset	SBE	Org1 only	<b>FAILS</b> – not owning org – remains at 600 (not 700)	Org2 / 600
Org1	Delete Asset	SBE	Org1 only	<b>FAILS</b> – not owning org	as above
Org2	Delete Asset	SBE	Org2 only	Succeeds	n/a

# Running the State Based Endorsement (SBE) sample

Option 1: manually – perform steps per the README or

Option 2: run the *app.js* node application per step 4 below.

1. Bring up 2-org network:

```
./network.sh up createChannel -ca
```

2. Deploy SBE contract (Typescript version in this case):

```
./network.sh deployCC -ccn sbe -ccl typescript
```

3. Change to *fabric-samples/asset-transfer-sbe/application-javascript* and run:

```
npm install
```

4. Run the javascript client *app.js* – this acts as an app from each of Org1, Org2 to complete the scenario sequence described.

To get detailed level DEBUG of SBE at SDK logging level “under the covers” – then re-run *app.js* to see endorsement, commit activity:

```
export HFC_LOGGING='{ "debug": "console" }'
```



# Community Chat

---



- **Developer Chat (RocketChat) – channel names for posting:**

<https://chat.hyperledger.org/channel/>

**#fabric-chaincode-dev**

**#fabric-java-chaincode**

**#fabric-sdk-node**

**#fabric-sdk-java**

**#fabric-sdk-go**

# Community Support

---



- **Questions & Open Community Support (Stack Overflow, Mailing List, Twitter)**

**Contracts:** <http://stackoverflow.com/questions/tagged/hyperledger-fabric>  
<http://stackoverflow.com/questions/tagged/hyperledger-chaincode>

**SDKs:** <http://stackoverflow.com/questions/tagged/hyperledger-fabric-sdk-js>  
*(etc ie add 'language suffix')*

**Twitter:** <https://twitter.com/hyperledger>

**Mailing List:** details next page

# Application Developer Community Support (Mailing List)

---



- **Hyperledger Mailing List**

To subscribe or unsubscribe, visit  
<https://lists.hyperledger.org/g/fabric>

or, via email, send a message with subject or body 'help' to  
[fabric@lists.hyperledger.org](mailto:fabric@lists.hyperledger.org)

Lists (Subgroups): <https://lists.hyperledger.org/g/main>

Help: [fabric+help@lists.hyperledger.org](mailto:fabric+help@lists.hyperledger.org)

You can change your settings once you log in at  
<https://lists.hyperledger.org/g/main>

# Further Links

---



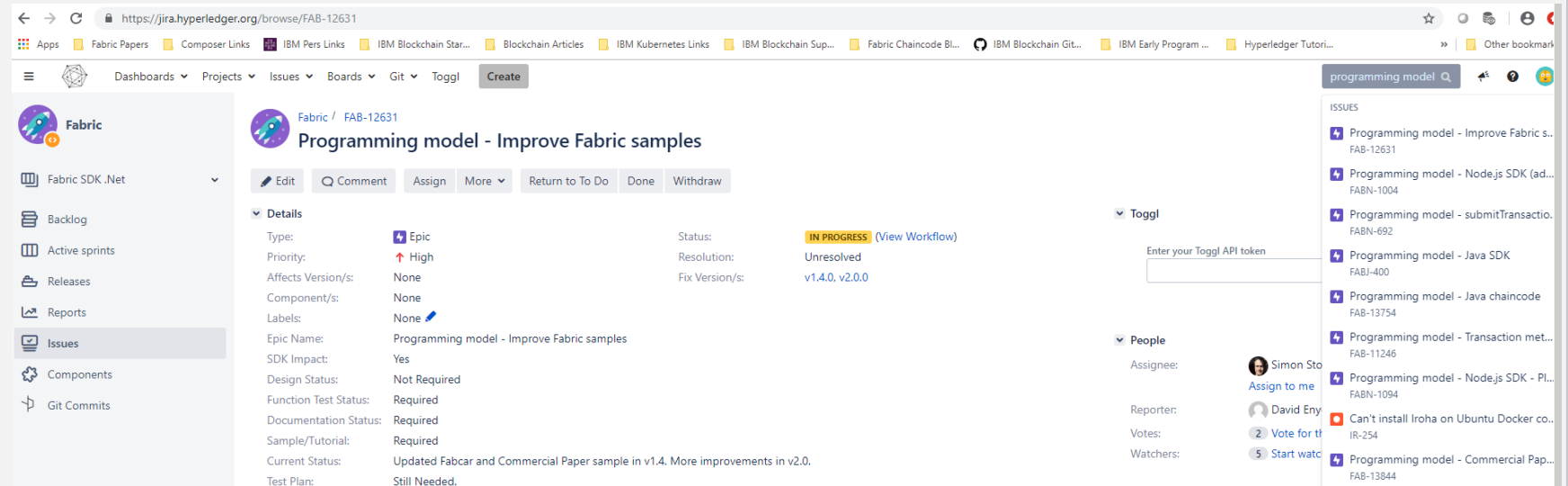
- **Fabric Developer - Community Home Page**  
<https://wiki.hyperledger.org/display/fabric/Fabric+Application+Developer+Community+Calls>
- **Fabric Developer Community – resources**  
<https://wiki.hyperledger.org/display/fabric/Resources%3A+Fabric+App+Developer+Community>

- Developing Applications using Hyperledger Fabric** (using the new programming model)  
[https://hyperledger-fabric.readthedocs.io/en/latest/developapps/developing\\_applications.html](https://hyperledger-fabric.readthedocs.io/en/latest/developapps/developing_applications.html)
- Fabric Application Developer samples (Contracts, SDK, Tutorials):**
  - **Fabric samples Repo (eg Commercial Paper, Fabcar):** : <https://github.com/hyperledger/fabric-samples/tree/release-1.4/>
  - **'Animal Tracking' TypeScript sample Contract/App Client (1.4)** <https://github.com/mahoney1/animaltracking> **with**
  - **Tutorial:** <https://github.com/mahoney1/docs/blob/master/animaltracking-tutorial.md>
- Go developer API (prototype) example** - (see README): <https://github.com/awjh-ibm/fabric-go-developer-api>
- Resources, tips and best practices** to share with the community  
<https://github.com/ampretia/fabric-application-examples>

# Fabric Team encourages you to review/comment on current Fabric JIRAs (ie stories, epics, requests relating to work-in-progress)

Eg. JIRA search - 'programming model' – (results below)

JIRAs – feel free to comment, give your input



The screenshot shows a JIRA issue page for 'Programming model - Improve Fabric samples' (ID FAB-12631). The issue is categorized as an Epic with a High priority. The status is 'IN PROGRESS' and it is currently 'Unresolved'. The fix versions are listed as v1.4.0 and v2.0.0. The issue description includes a current status update: 'Updated Fabcar and Commercial Paper sample in v1.4. More improvements in v2.0.' and a test plan status of 'Still Needed.'.

The right sidebar shows search results for 'programming model', listing several related issues such as 'Programming model - Improve Fabric s...' (FAB-12631), 'Programming model - Node.js SDK (ad...' (FABN-1004), and 'Programming model - submitTransactio...' (FABN-692).

# Reference

---



## New Programming Model (1.4.x) :

- JIRA references - Fabric Programming Model Info:
  - <https://jira.hyperledger.org/projects/FABN/issues/FABN-692>
  - <https://jira.hyperledger.org/browse/FAB-11246>

THANKS!



CIAO FOR NOW!

