# Introduction

Contents

[Introduction 1](#_Toc535912661)

[Content 1](#_Toc535912662)

# The packages.

## fabric-client

I have seen this in low level java script code.

Command for installation:- npm install fabric-client

SDK for writing node.js applications to interact with [Hyperledger Fabric](http://hyperledger-fabric.readthedocs.io/en/latest/).

This package encapsulates the APIs to interact with Peers and Orderers of the Fabric network to install and instantiate chaincodes, send transaction invocations and perform chaincode queries.

Reference:- <https://www.npmjs.com/package/fabric-client>

## fabric-ca-client

I have seen this in low level java script code.

Npm install fabric-ca-client

This package encapsulates the APIs to interact with the Fabric CA to manage user certificates lifecycle such as register, enroll, renew and revoke.

Reference:- <https://www.npmjs.com/package/fabric-ca-client>

## fabric-network

I have seen this mostly in javascript code.

This package encapsulates the APIs to connect to a Fabric network, submit transactions and perform queries against the ledger.

Reference: - <https://www.npmjs.com/package/fabric-network>

# Application developer documentations

Reference link: - <https://fabric-sdk-node.github.io/>

Reference link(preferable) :- <https://fabric-sdk-node.github.io/release-1.4/module-api.html>

# Hyperledger Fabric Client objects

fabric-client and fabric-ca-client are written in CommonJS modules and take advantage of ECMAScript 2015 class syntax.

* The main top-level class is **Client**. The client's view of a fabric [channel] is the class **Channel**. The SDK allows you to interact with multiple channels. A channel object can be configured with a different ordering service or share a common ordering service, depending on how the target blockchain network is set up. A client object has a *KeyValueStore* to store private keys and certificates for authenticated users. Through the client object the application can perform
* The **KeyValueStore** is a very simple interface which SDK uses to store and retrieve all persistent data. This data includes private keys, so it is very important to keep this storage secure. The default implementation is a simple file-based version found in the *FileKeyValueStore* class. The SDK also provides an implementation based on CouchDB which can be configured to use a local CouchDB database or a remote deployment including a Cloudant database.
* The **User** class represents an end user who transacts on the channel. The user object must have a valid enrollment configured in order to properly sign transaction requests. The enrollment materials can either be obtained from enrolling with fabric-ca or an external Certificate Authority.
* The **FabricCAClientImpl** class provides security and identity related features such as user registration and enrollment, transaction certificate issuance. The Hyperledger Fabric has a built-in implementation that issues *ECerts* (enrollment certificates) and *TCerts* (transaction certificates). ECerts are for enrollment identity and TCerts are for transactions.

# Fabcar

## startFabric.sh

Language is selected.

### Start.sh

Start.sh is called. (network creation For channel creation and joining of peer to channel)

You can read about start.sh [here](#_start.sh).

### Launching the cli container

Now launch the cli container.in order to install instantiate chaincode and to prime the ledger with 10 cars

docker-compose -f ./docker-compose.yml up -d cli

-f => specify an alternate file

docker-compose up -d starts the containers in the background and leaves them running.

### Chain Code Install

docker exec -e "CORE\_PEER\_LOCALMSPID=Org1MSP" -e "CORE\_PEER\_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp" cli peer chaincode install -n fabcar -v 1.0 -p "$CC\_SRC\_PATH" -l "$CC\_RUNTIME\_LANGUAGE"

-e +> setting environment variables

Peer chaincode install => installing chaincode

**-**n, **--**name string Name of the chaincode

**-**v, **--**version string Version of the chaincode specified **in** install**/**instantiate**/**upgrade commands

**-**p, **--**path string Path to chaincode

**-**l, **--**lang string Language the chaincode **is** written **in** (default "golang")

### Chain Code Instantiate

|  |
| --- |
| docker exec -e "CORE\_PEER\_LOCALMSPID=Org1MSP" -e "CORE\_PEER\_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp" cli peer chaincode instantiate -o orderer.example.com:7050 -C mychannel -n fabcar -l "$CC\_RUNTIME\_LANGUAGE" -v 1.0 -c '{"Args":[]}' -P "OR ('Org1MSP.member','Org2MSP.member')"  peer chaincode instantiate  -o => orderer end point  **-**C, **--**channelID string The channel on which this command should be executed  -n => name of the chaincode  -l => language (default go)  -c => Constructor message **for** the chaincode **in** JSON format (default "{}")  **-**P, **--**policy string The endorsement policy associated to this chaincode |

Sleep for 10 Seconds

### Chain Code Invoke

docker exec -e "CORE\_PEER\_LOCALMSPID=Org1MSP" -e "CORE\_PEER\_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp" cli peer chaincode invoke -o orderer.example.com:7050 -C mychannel -n fabcar -c '{"function":"initLedger","Args":[]}'

-e => environment variables

Peer chaincode invoke

-o => orderer end point

-C => channel on which to execute

-n => name of the chaincode

-c => constructor msg in JSON format

## start.sh

### Container creation

it reads the docker compose file and starts the ca authority, orderer and peer.

docker-compose -f docker-compose.yml up -d ca.example.com orderer.example.com peer0.org1.example.com couchdb

up=> Define and run multi-container applications with Docker.

-f => specify an alternate file

-d, --detach => Detached mode: Run containers in the background,

print new container names. Incompatible with

--abort-on-container-exit.

docker-compose up -d starts the containers in the background and leaves them running.

We wait for 10 seconds (sleep) in order for hyperledger to start.

### Channel creation

Now we create the channel

docker exec -e "CORE\_PEER\_LOCALMSPID=Org1MSP" -e "CORE\_PEER\_MSPCONFIGPATH=/etc/hyperledger/msp/users/Admin@org1.example.com/msp" peer0.org1.example.com peer channel create -o orderer.example.com:7050 -c mychannel -f /etc/hyperledger/configtx/channel.tx

Usage: docker exec [OPTIONS] CONTAINER COMMAND [ARG...]

Run a command in a running container

Options

-e => set environment variables.

usage: docker create [OPTIONS] IMAGE [COMMAND] [ARG...]

peer channel create => Create a channel

peer channel create ( –o) => orderer end point

**-**c, **--**channelID string => In case of a newChain command, the channel ID to create**.** It must be all lower case, less than 250 characters long **and** match the regular expression: [a**-**z][a**-**z0**-**9.**-**]**\***

**-**f, **--**file string Configuration transaction file generated by a tool such **as** configtxgen **for** submitting to orderer

Operate a channel: create**|**fetch**|**join**|**list**|**update**|**signconfigtx**|**getinfo**.**

Usage:

peer channel [command]

Available Commands:

create Create a channel

fetch Fetch a block

getinfo get blockchain information of a specified channel**.**

join Joins the peer to a channel**.**

list List of channels peer has joined**.**

signconfigtx Signs a configtx update**.**

update Send a configtx update**.**

Flags:

**--**cafile string Path to file containing PEM**-**encoded trusted certificate(s) **for** the ordering endpoint

**--**certfile string Path to file containing PEM**-**encoded X509 public key to use **for** mutual TLS communication **with** the orderer endpoint

**--**clientauth Use mutual TLS when communicating **with** the orderer endpoint

**--**connTimeout duration Timeout **for** client to connect (default 3s)

**-**h, **--**help help **for** channel

**--**keyfile string Path to file containing PEM**-**encoded private key to use **for** mutual TLS communication **with** the orderer endpoint

**-**o, **--**orderer string Ordering service endpoint

**--**ordererTLSHostnameOverride string The hostname override to use when validating the TLS connection to the orderer**.**

**--**tls Use TLS when communicating **with** the orderer endpoint

Global Flags:

**--**logging**-**level string Default logging level **and** overrides, see core**.**yaml **for** full syntax

Use "peer channel [command] --help" **for** more information about a command**.**

### Joining of peer to channel

# Join peer0.org1.example.com to the channel.

docker exec -e "CORE\_PEER\_LOCALMSPID=Org1MSP" -e "CORE\_PEER\_MSPCONFIGPATH=/etc/hyperledger/msp/users/Admin@org1.example.com/msp" peer0.org1.example.com peer channel join -b mychannel.block

join => Joins the peer to a channel**.**

**-**b, **--**blockpath => string Path to file containing genesis block

### References:-

<https://hyperledger-fabric.readthedocs.io/en/release-1.3/commands/peerchannel.html>

<https://hyperledger-fabric.readthedocs.io/en/release-1.3/commands/peerchannel.html>

<https://docs.docker.com/engine/reference/commandline/exec/>

<https://hyperledger-fabric.readthedocs.io/en/release-1.3/command_ref.html>

## Order of execution of javascript files

### Enrolling Admin:-

node enrollAdmin

### Register User:-

node registerUser

a new user user1 will be createdwhich will be used by the other applications to interact with the deployed FabCar contract.

### Invoke

node invoke

By default the invoke application will create a new car, but you can update the application to submit new transactions

### Query

node query

By default the query application will return all cars but you can modify to suit your usage