gemalto
security to be free

HSMoD Service

HYPERLEDGER FABRIC (BLOCKCHAIN) INTEGRATION GUIDE

Document Information

Product Version	1.7
Document Part Number	007-013897-001
Release Date	21 November 2018

Revision History

Revision	Date	Reason
Rev. A	17 August 2017	For initial release 1.1.0
Rev. B	19 September 2017	For release 1.1.1
Rev. C	14 November 2017	For release 1.2
Rev. D	05 February 2018	For release 1.3
Rev. E	02 March 2018	For HSM on Demand release 1.3
Rev. F	05 April 2018	For release 1.4
Rev. G	07 May 2018	For HSM on Demand release 1.4
Rev. H	10 June 2018	For release 1.5
	12 September 2018	For release 1.6
	21 November 2018	For release 1.7

SafeNet Data Protection on Demand1.7

HYPERLEDGER FABRIC (BLOCKCHAIN) INTEGRATION GUIDE

Contents

Overview

This guide demonstrates using an HSM on Demand Service's PKCS#11 API to securely store a Hyperledger Admin Certificate Authority (CA), Peer, and Orderer private keys. The guide then provides examples of the e2e_cli end-to-end execution for creating channels and querying the chaincode.

This integration guide features sample material to demonstrate the process of integrating your HSM on Demand Service with Blockchain HyperLedger Fabric.

Using an HSM on Demand Service to generate the ECDSA signing keys for Blockchain Identities provides the following benefits:

> Secure generation, storage, and protection of Identity signing keys on a FIPS 140-2 level 3 validated HSM*.

> Full life cycle management of the keys.

> Performance improvements resulting from off-loading cryptographic operations from application servers to the HSM on Demand Service.

This document contains the following sections:

> "Preparing for the Integration" on page 6

> "Integrating Hyperledger Fabric (Blockchain) with an HSM on Demand Service" on page 9

*Validation in progress

Third Party Application Details

This integration guide uses the following third party applications:

> Hyperledger Fabric

> Hyperledger Fabric CA

Supported Platforms

The following platforms are tested with HSMoD Service:

Platforms Tested	Golang	Docker	Docker Compose
RHEL 64-bit	1.8.3	17.06.0-ce	1.8.0

Preparing for the Integration

Before you proceed with the integration, ensure you have completed the following:

> "Provision HSM on Demand Services" below

> "Install Unix Components" on the next page

> "Set up Golang" on the next page

> "Set up Docker " on the next page

Provision HSM on Demand Services

Follow the instructions in the *HSM on Demand Application Owner Quick Start Guide* and complete the following:

1. Select the **Hyperledger** tile and create the following three **HSM on Demand Services**.
   - org1.example.com
   - org2.example.com
   - orderer.example.com
2. For each service create a client with the same name, and download the zip to the host machine.
3. Execute the following commands to create the service directories on the host machine.

   # mkdir -p /etc/hyperledger/fabric/dpod/org1.example.com

   # mkdir -p /etc/hyperledger/fabric/dpod/org2.example.com

   # mkdir -p /etc/hyperledger/fabric/dpod/orderer.example.com
4. Unzip the three clients in their respective directories.
5. Initialize the service, Crypto Officer, and Crypto User roles on the services. For the purpose of this integration guide, the passwords have been set to userpin. Follow the instructions in the *Application Owner Quick Start Guide* and set the **ChrystokiConfigurationPath** environment variable to point to the **Chrystoki.conf** file.

   NOTE In a production configuration the passwords should be set to coincide with your organization's security policy.

   NOTE We recommend using separate HSMoD Service's for the Peer, Orderer, and Users roles.

Constraints on HSMoD Services

Please take the following limitations into consideration when integrating your application software with an HSM on Demand Service.

HSM on Demand Service in FIPS mode

HSMoD services operate in a FIPS and non-FIPS mode. If your organization requires non-FIPS algorithms for your operations, ensure you enable the **Allow non-FIPS approved algorithms** check box when configuring your HSM on Demand service. The FIPS mode is enabled by default.

Refer to the *Mechanism List* in the SDK Reference Guide for more information about available FIPS and non-FIPS algorithms.

Verify HSM on Demand <slot> value

LunaCM commands work on the current slot. If there is only one slot, then it is always the current slot. If you are completing an integration using HSMoD services, you need to verify which slot on the HSMoD service you send commands to. If there is more than one slot, then use the **slot set** command to direct a command to a specified slot. You can use **slot list** to determine which slot numbers are in use by which HSMoD service.

Install Unix Components

The following components must be installed on the host system where you will generate the keys for the User, Peer, and Orderer:

> Git

> Curl

> Alien

> Python

Execute the following command as a user with privileges to install the components:

sudo apt-get install git curl alien python-pip libtool libltdl-dev

Set up Golang

Hyperledger Frabric uses the Go programming language. Download the golang binaries from https://golang.org/dl/. Follow the instructions at https://golang.org/doc/install and install the golang binaries.

Ensure that the GO program is in the PATH variable.

# export PATH=/usr/local/go/bin:$PATH

If the GOPATH is not set, set it. The value will be a directory of the development workspace.

# export GOPATH=/opt/gopath

# mkdir –p $GOPATH/src/github.com/hyperledger

# cd $GOPATH/src/github.com/hyperledger

Set up Docker

Docker and Docker-compose need to be installed on the host system. Follow the instructions at https://docs.docker.com/engine/installation/linux/docker-ce/ubuntu/ to install the Docker-CE.

Execute **sudo pip install docker-compose==1.8.0** to install the docker-compose.

You can configure Docker so that it does not require **sudo**. Execute:

# sudo gpasswd –a $USER docker

# newgrp docker

Update the Dockerfiles

You need to add a command to the peer and orderer **Dockerfile.in** to install the libtool.

To update the Dockerfiles

1. Open the **images/orderer/Dockerfile.in** in a text editor.

   Add

   ```
   RUN apt-get update && apt-get install -y libtool
   ```

   After

   ```
   RUN mkdir -p /var/hyperledger/production $FABRIC_CFG_PATH
   ```

2. Open the **images/peer/Dockerfile.in** in a text editor.

   Add

   ```
   RUN apt-get update && apt-get install -y libtool
   ```

   After

   ```
   RUN mkdir -p /var/hyperledger/production $FABRIC_CFG_PATH
   ```

Integrating Hyperledger Fabric (Blockchain) with an HSM on Demand Service

This guide will detail how to generate keys for the User, Peer, and Orderer roles on the HSM on Demand Servicein an example configuration. It will then demonstrate the e2e_cli end-to-end execution for creating channels and querying the chaincode.

This integration contains the following topics:

Configuring HyperLedger Fabric to use PKCS11 Blockchain Cryptographic Service Provider (BCCSP)

You must make modifications to the Hyperledger Fabric configuration files before you can begin generating keys using the PKCS11 Blockchain Cryptographic Service Provider (BCCSP).

To configure HyperLedger Fabric to use PKCS11 BCCSP

1. Create the Hyperledger fabric repository:

   # git clone https://gerrit.hyperledger.org/r/fabric

   # cd fabric

2. Open the **docker-env.mk** file in a text editor. Remove the **–static** linking option from the **DOCKER_GO_ LDFLAGS** line.

   Original: DOCKER_GO_LDFLAGS += -linkmode external -extldflags '-static -lpthread'

   Updated: DOCKER_GO_LDFLAGS += -linkmode external -extldflags '-lpthread'

3. Compile the docker images and executables

```
# make docker
# make release
```

4. Clone the fabric-ca project and build the fabric-ca client binary.

```
# cd $GOPATH/src/github.com/hyperledger
# git clone https://gerrit.hyperledger.org/r/fabric-ca
# cd fabric-ca
# git checkout -b v1.1.0 v1.1.0
```

NOTE This integration guide was developed using the v1.1.0 tag. We recommend you use this version, as the instructions may not be compatible with the latest versions available in the master branch of Hyperledger Fabric project.

```
# make fabric-ca-client
```

Generating a Certificate Signing Request using fabric-ca-client and PKCS11 BCCSP

The fabric-ca-client utility can be used to generate certificate signing requests (CSR) for Peers, Orderers, and Users in their respective MSP directories.

The fabric-ca-client utility uses the BCCSP to generate cryptographic material. If you configure the BCCSP to use a PKCS11 implementation, you can generate and store the keys using the HSM on Demand Service.

You must update the fabric-ca-client-config.yaml to use the PKCS11 cryptographic provider, configure the Orderer and Peer nodes, and initialize communication between the objects.

Configuring the User nodes

You must configure the User nodes by updating the **fabric-ca-client-config.yaml** file to use the PKCS11 cryptographic provider.

You can do this by updating the file in a text editor or exporting the environment variables. See:

> "To update the fabric-ca-client-config.yaml in a text editor" below

> "To update the fabric-ca-client-config.yaml using environment variables" on the next page

To update the fabric-ca-client-config.yaml in a text editor

1. Open the **~/.fabric-ca-client/fabric-ca-client-config.yaml** file in a text editor and update the file for your configuration. The following is an example of a BCCSP configuration using an HSM on Demand Service.

```
bccsp:
    default: PKCS11
    sw:
        hash: SHA2
        security: 256
        filekeystore:
            # The directory used for the software file-based keystore
            keystore: msp/keystore
    pkcs11:
        library: /etc/hyperledger/fabric/dpod/org1.example.com/libs/64/libCryptoki2.so
        pin: userpin
        label:
        hash: SHA2
        security: 256
        filekeystore:
            # The directory used for the software file-based keystore
               keystore: msp/keystore
```

2. Add a **keyrequest** setting to the **csr** section to specify the key size.

```
Add:
       KeyRequest:
```

```
A: ecdsa
S: 256
```

To update the fabric-ca-client-config.yaml using environment variables

1. Export the following environment variables to configure the **fabric-ca-client-config.yaml** file. Execute the following to export the values:

```
# export FABRIC_CA_CLIENT_BCCSP_DEFAULT=PKCS11
# export FABRIC_CA_CLIENT_BCCSP_PKCS11_LABEL=<HSMoD_service_label>
# export FABRIC_CA_CLIENT_BCCSP_PKCS11_PIN=<HSMoD_SO_password>
# export FABRIC_CA_CLIENT_BCCSP_PKCS11_LIBRARY=<PKCS11_library>
```

Configuring the Peer nodes

To configure the Peer nodes to access the HSM on Demand Service keys you must update the **core.yaml** file or set the environment variables for the following:

> PKCS11 BCCSP cryptographic library

> the HSM on Demand Service

> HSM on emand Servicepassword

> location of the configuration files

You can do this by updating the file in a text editor or exporting the environment variables. See:

> "To update the core.yaml file in a text editor" below

> "To update the core.yaml file using environment variables" below

Additionally, you must mount the **core.yaml** file in the volumes section . The **core.yaml** file provides basic configuration options for Peer modules.

To update the core.yaml file in a text editor

1. Open the **core.yaml** file in a text editor.
2. Specify PKCS11 as the default cryptographic provider in the BCCSP section. The file should appear as follows:

```
BCCSP:
Default: PKCS11
PKCS11:
# TODO: The default Hash and Security level needs refactoring to be
# fully configurable. Changing these defaults requires coordination
# SHA2 is hardcoded in several places, not only BCCSP
Hash: SHA2
Security: 384
Library: /etc/hyperledger/fabric/dpod/org1.example.com/libs/64/libCryptoki2.so
Label: <organization_label>
Pin: <password>
SoftwareVerify: true
SensitiveKeys: true
#FileKeyStore:
# KeyStore:
```

To update the core.yaml file using environment variables

1. Export the following environment variables to change the configuration settings of the **core.yaml** file.

```
# export CORE_PEER_BCCSP_DEFAULT=PKCS11
# export CORE_PEER_BCCSP_PKCS11_LABEL=<HSMoD_service_label>
# export CORE_PEER_BCCSP_PKCS11_PIN=<HSMoD_SO_password>
# export CORE_PEER_BCCSP_PKCS11_LIBRARY=<PKCS11_library>
```

2. Ensure that the HSM on Demand Service client directory is available to the peer, and is using the correct HSM on Demand Service for the peer. The **ChrystokiConfigurationPath** must point to the HSM on Demand Service client directory, where the **Crystoki.conf** is stored.

```
# export ChrystokiConfigurationPath=<path_to_Chrystoki.conf>
```

Configuring the Orderer nodes

To configure the Orderer nodes to access the HSM on Demand Service keys you must update the **orderer.yaml** file or set the environment variables for the following:

> PKCS11 BCCSP cryptographic library

> the HSM on Demand Service

> HSM on Demand Service password

> location of the configuration files

You can do this by updating the file in a text editor or exporting the environment variables. See:

> "To update the orderer.yaml file in a text editor" below

> "To update the orderer.yaml file using environment variables" below

Additionally, you must mount the **orderer.yaml** file in the volume section. The **orderer.yaml** file provides basic configuration options for Orderer modules.

To update the orderer.yaml file in a text editor

1. Open the **orderer.yaml** file in a text editor.
2. Specify PKCS11 as the default cryptographic provider in the BCCSP section. The file should appear as follows:

```
BCCSP:
Default: PKCS11
PKCS11:
# TODO: The default Hash and Security level needs refactoring to be
# fully configurable. Changing these defaults requires coordination
# SHA2 is hardcoded in several places, not only BCCSP
Hash: SHA2
Security: 384
Library: /etc/hyperledger/fabric/dpod/orderer.example.com/libs/64/libCryptoki2.so
Label: <organization_label>
Pin: <password>
SoftwareVerify: true
SensitiveKeys: true
#FileKeyStore:
# KeyStore:
```

To update the orderer.yaml file using environment variables

1. Export the following environment variables to change the configuration settings of the **orderer.yaml** file.

```
# export ORDERER_GENERAL_BCCSP_PKCS11_LABEL=<HSMoD_service_label>
# export ORDERER_GENERAL_BCCSP_PKCS11_PIN=<HSMoD_SO_password>
# export ORDERER_GENERAL_BCCSP_PKCS11_LIBRARY=<PKCS11_library>
```

2. Ensure that the HSM on Demand Service client directory is available to the peer, and is using the correct HSM on Demand Service for the peer. The **ChrystokiConfigurationPath** must point to the HSM on Demand Service client directory, where the **Crystoki.conf** is stored.

```
# export ChrystokiConfigurationPath=<path_to_Chrystoki.conf>
```

Generating a CSR

You must generate a CSR for each role node in the configuration. You need to adjust the options and variables for the requirements of the particular CSR.

The command to generate a CSR utilizes the following syntax:

```
# ./fabric-ca-client gencsr --csr.cn <value> --mspdir <value> --csr.names <value>
```

Argument	Description
--csr.cn <value>	The common name field of the certificate signing request.
--mspdir <value>	Path to the Membership Service Provider directory.
--csr.names <value>	A list of comma-separated CSR labels of the form <name>=<value>. Values include: C=<country>, ST=<state>, OU=<user_role>

NOTE When generating CSR requests ensure that you specify the correct CN, MSP directory, and CSR names. The OU value should equate to peer, orderer or client, depending on the CSR.

To generate the User CSR

1. Execute the following to generate the User CSR for **org1.example.com**.

```
# ./fabric-ca-client gencsr --csr.cn <common_name_for_CSR> --mspdir <MSP_directory> --csr.names
"C=US,ST=California,L=San Francisco,OU=<OU>"
```

2. Submit the CSR to your CA to obtain the signed certificate for the role, and place the signed certificate in the respective **msp/signcerts** directory.

Example Configuration: Running e2e_cli end-2-end execution

The following procedural set provides an example e2e_cli end-2-end execution configuration. Before beginning this procedural set, it is assumed you have completed the prerequisite sections of the documentation set, and the preceding tasks.

1. Copy the compiled **fabric-ca-client** to the **examples/e2e_cli** directory.

   # cd $GOPATH/src/github.com/hyperledger/fabric-ca/

   # cp bin/fabric-ca-client ../fabric/examples/e2e_cli

2. Copy the following script and save the file as **gencerts.sh**.

NOTE The script works in conjunction with the **cryptogen** tool. The script generates all of the Peer, Orderer, and Admin user MSPs using the **fabric-ca-client gencsr** command. Certificate are generated using **openssl**.

```
-----------------------------------------------------------------------------------------------
-----------------------------------------------
#!/bin/bash

#Copyright (C) 2017 SafeNet. All rights reserved.

###########################################################################
# This script generates certificates and keys to work with the cryptogen util
# to be used with the e2e_cli hyperledger fabric example.
# This allows the keys for the certificate to be generated with the
# PKCS11 BCCSP which in turn allows keys to be generated in an HSM.
##########################################################################

CA_CLIENT=./fabric-ca-client
CRYPTO_CONFIG=$PWD/crypto-config
ROOT=$PWD

BCCSP_DEFAULT=PKCS11
PIN=userpin

check_error() {
  if [ $? -ne 0 ]; then
    echo "ERROR:  Something went wrong!"
    exit 1
  fi
}

signcsr() {
  MSP=$1
  CN=$2
  CA_DIR=$3
  CA_NAME=$4
  CA_CERT=$(find $CA_DIR -name "*.pem")
  CA_KEY=$(find $CA_DIR -name "*_sk")
  CSR=$MSP/signcerts/$CN.csr
  CERT=$MSP/signcerts/$CN-cert.pem

openssl x509 -req -sha256 -days 3650 -in $CSR -CA $CA_CERT -CAkey $CA_KEY -CAcreateserial -out
$CERT
  check_error
}

genmsp() {
  ORG_DIR=$1
  ORG_NAME=$2
  NODE_DIR=$3
  NODE_NAME=$4
  NODE_OU=$6
  CN=${NODE_NAME}${ORG_NAME}
  CA_PATH=$CRYPTO_CONFIG/$ORG_DIR/$ORG_NAME
  NODE_PATH=$CA_PATH/$NODE_DIR/$CN
  MSP=$NODE_PATH/msp
  TLS=$NODE_PATH/tls

  LABEL=$5
  rm -rf $MSP/keystore/*
  rm -rf $MSP/signcerts/*
  echo $LABEL
  export FABRIC_CA_CLIENT_BCCSP_DEFAULT=$BCCSP_DEFAULT
  export FABRIC_CA_CLIENT_BCCSP_PKCS11_LABEL=$LABEL
  export FABRIC_CA_CLIENT_BCCSP_PKCS11_PIN=$PIN
  export ChrystokiConfigurationPath=/etc/hyperledger/fabric/dpod/$LABEL
  export FABRIC_CA_CLIENT_BCCSP_PKCS11_
LIBRARY=/etc/hyperledger/fabric/dpod/$LABEL/libs/64/libCryptoki2.so
  $CA_CLIENT gencsr --csr.cn $CN --mspdir $MSP --csr.names "C=US,ST=California,L=San
Francisco,OU=$NODE_OU"
  check_error

  signcsr $MSP $CN $CA_PATH/ca $ORG_NAME
}

copy_admin_cert_node() {
  ORG_DIR=$1
  ORG_NAME=$2
  NODE_DIR=$3
  NODE_NAME=$4
  CN=$NODE_NAME.$ORG_NAME
  CA_PATH=$CRYPTO_CONFIG/$ORG_DIR/$ORG_NAME
  NODE_PATH=$CA_PATH/$NODE_DIR/$CN
  MSP=$NODE_PATH/msp
  ADMIN_CN=Admin@$ORG_NAME
  ADMIN_CERT=$CA_PATH/users/$ADMIN_CN/msp/signcerts/$ADMIN_CN-cert.pem
  cp $ADMIN_CERT $NODE_PATH/msp/admincerts
  check_error
}

copy_admin_cert_ca() {
  ORG_DIR=$1
  ORG_NAME=$2
  CA_PATH=$CRYPTO_CONFIG/$ORG_DIR/$ORG_NAME
  ADMIN_CN=Admin@$ORG_NAME
  ADMIN_CERT=$CA_PATH/users/$ADMIN_CN/msp/signcerts/$ADMIN_CN-cert.pem
  cp $ADMIN_CERT $CA_PATH/msp/admincerts
  check_error
  cp $ADMIN_CERT $CA_PATH/users/$ADMIN_CN/msp/admincerts
  check_error
}

for org in 1 2; do
  for peer in 0 1; do
    genmsp peerOrganizations org${org}.example.com peers peer${peer}. org${org}.example.com
peer
  done
  genmsp peerOrganizations org${org}.example.com users Admin@ org${org}.example.com client
  for peer in 0 1; do
    copy_admin_cert_node peerOrganizations org${org}.example.com peers peer${peer}
  done

  copy_admin_cert_ca peerOrganizations org${org}.example.com

done

genmsp ordererOrganizations example.com orderers orderer. orderer.example.com orderer
genmsp ordererOrganizations example.com users Admin@ orderer.example.com client

copy_admin_cert_node ordererOrganizations example.com orderers orderer orderer.example.com
copy_admin_cert_ca ordererOrganizations example.com
```

```
--------------------------------------------------------------------------------
```

3. Copy the **core.yaml** and **orderer.yaml** files to the **examples/e2e_cli** directory.

   # cp ../../sampleconfig/core.yaml ../../sampleconfig/orderer.yaml .

4. Open both the **core.yaml** and **orderer.yaml** files in a text editor. Edit the BCCSP section in both files to use the following values.

```
BCCSP:
  Default: PKCS11
  PKCS11:
  # TODO: The default Hash and Security level needs refactoring to be
  # fully configurable. Changing these defaults requires coordination
  # SHA2 is hardcoded in several places, not only BCCSP
  Hash: SHA2
  Security: 256
  Library:
  Label:
  Pin:
  SoftwareVerify: true
  SensitiveKeys: true
  #FileKeyStore:
  #    KeyStore:
```

5. Open the **base/peer-base.yaml** file in a text editor.
   a. Add the following line to the Environment section:

   ```
   CORE_PEER_BCCSP_PKCS11_PIN=userpin
   ```

   b. Add a Volumes section with the following entries to the base of the Services section:

   ```
   volumes:
   - ../core.yaml:/etc/hyperledger/fabric/core.yaml
   ```

6. Open the **base/docker-compose-base.yaml** file in a text editor.
   a. Add the following line to the Environment section of orderer.example.com:

   ```
   - ORDERER_GENERAL_BCCSP_PKCS11_PIN=userpin
   ```

   b. Add the following line to the Volumes section of orderer.example.com:

   ```
   - ../orderer.yaml:/etc/hyperledger/fabric/orderer.yaml
   ```

7. Open the **docker-compose-cli.yaml** file in a text editor.
   a. Add the following lines to the end of the **orderer.example.com** section:

   ```
   environment:
   - ORDERER_GENERAL_BCCSP_PKCS11_LABEL=orderer.example.com
   - ORDERER_GENERAL_BCCSP_PKCS11_LIBRARY=
   /etc/hyperledger/fabric/dpod/orderer.example.com/libs/64/libCryptoki2.so
   - ChrystokiConfigurationPath=/etc/hyperledger/fabric/dpod/orderer.example.com
   volumes:
   - etc/hyperledger/fabric/dpod/orderer.example.com:
   /etc/hyperledger/fabric/dpod/orderer.example.com
   ```

   b. Add the following lines to the end of the **peer.0.org1.example.com** and **peer1.org1.example.com** sections:

   ```
   environment:
   - CORE_PEER_BCCSP_PKCS11_LABEL=org1.example.com
   - CORE_PEER_BCCSP_PKCS11_LIBRARY=
   /etc/hyperledger/fabric/dpod/org1.example.com/libs/64/libCryptoki2.so
   ```

```
- ChrystokiConfigurationPath=/etc/hyperledger/fabric/dpod/org1.example.com
volumes:
- /etc/hyperledger/fabric/dpod/org1.example.com:
/etc/hyperledger/fabric/dpod/org1.example.com
```

c. Add the following lines to the end of the **peer0.org2.example.com** and **peer1.org2.example.com** sections:

```
environment:
- CORE_PEER_BCCSP_PKCS11_LABEL=org2.example.com
- CORE_PEER_BCCSP_PKCS11_LIBRARY=
/etc/hyperledger/fabric/dpod/org2.example.com/libs/64/libCryptoki2.so
- ChrystokiConfigurationPath=/etc/hyperledger/fabric/dpod/org2.example.com
volumes:
- /etc/hyperledger/fabric/dpod/org2.example.com:
/etc/hyperledger/fabric/dpod/org2.example.com
```

d. Add the following lines to the cli volumes section:

```
- /etc/hyperledger/fabric/dpod:/etc/hyperledger/fabric/dpod
- ./core.yaml:/etc/hyperledger/fabric/core.yaml
```

8. Open the **scripts/scripts.sh** file in a text editor.

   a. Add the following lines to the setGlobals function after “if [$1 –eq 0 –o $1 –eq 1] ; then”:

```
export CORE_PEER_BCCSP_PKCS11_
LIBRARY=/etc/hyperledger/fabric/dpod/org1.example.com/libs/64/libCryptoki2.so
export CORE_PEER_BCCSP_PKCS11_PIN=userpin
export CORE_PEER_BCCSP_PKCS11_LABEL=org1.example.com
export ChrystokiConfigurationPath=/etc/hyperledger/fabric/dpod/org1.example.com
```

   b. Add the following lines to the setGlobals function after “else”:

```
export CORE_PEER_BCCSP_PKCS11_
LIBRARY=/etc/hyperledger/fabric/dpod/org2.example.com/libs/64/libCryptoki2.so
export CORE_PEER_BCCSP_PKCS11_PIN=userpin
export CORE_PEER_BCCSP_PKCS11_LABEL=org2.example.com
export ChrystokiConfigurationPath=/etc/hyperledger/fabric/dpod/org2.example.com
```

   c. Add the following lines to the beginning of the checkOSNAvailability function:

```
export CORE_PEER_BCCSP_PKCS11_
LIBRARY=/etc/hyperledger/fabric/dpod/orderer.example.com/libs/64/libCryptoki2.so
export CORE_PEER_BCCSP_PKCS11_PIN=userpin
export CORE_PEER_BCCSP_PKCS11_LABEL=orderer.example.com
export ChrystokiConfigurationPath=/etc/hyperledger/fabric/dpod/orderer.example.com
```

9. Open the **generateArtifacts.sh** file in a text editor. Edit the bottom section of the file to use **gencerts.sh** to create key material. Modify the file so it appears as the following:

```
generateCerts
replacePrivateKey
./gencerts.sh
generateChannelArtifacts
```

10. Open the **network_setup.sh** file in a text editor. Comment the networkDown function so that artifacts are not deleted. Change:

```
rm -rf channel-artifacts/*.block channel-artifacts/*.tx crypto-config
```

to

```
# rm -rf channel-artifacts/*.block channel-artifacts/*.tx crypto-config
```

11. Enter the following command to run e2e_cli.

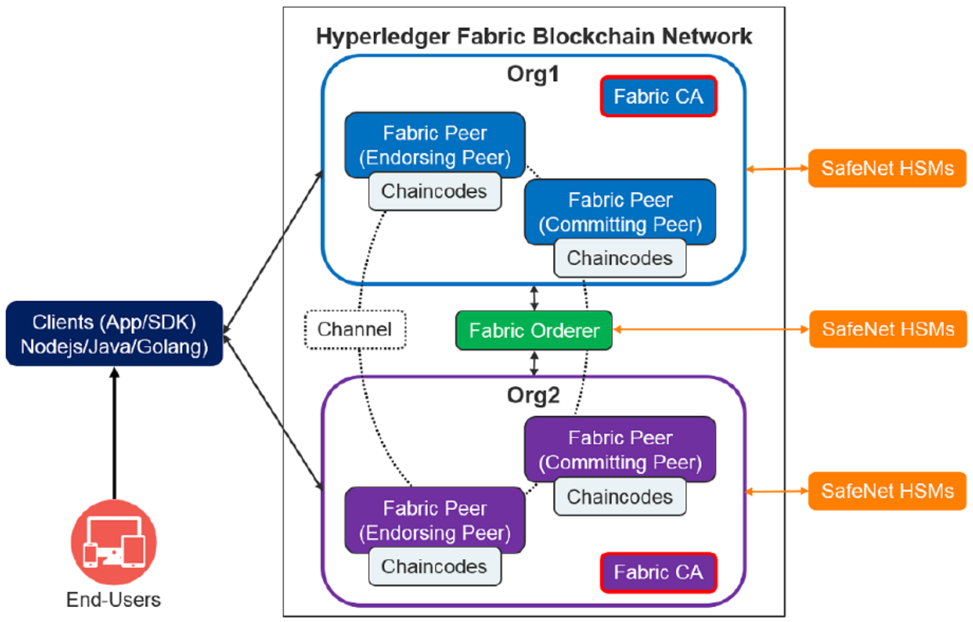
```
./generateArtifacts.sh
./network_setup.sh up
```

About the Hyperledger Fabric CA Server

Hyperledger Fabric is a variant of the Hyperledger Blockchain project. Hyperledger Fabric allows for private and permissioned access, the members of the network must enroll through a Membership Service Provider (MSP).

Hyperledger Fabric has a ledger, uses smart contracts, and is a system that allows participants to manage their transactions. Ledger data can be stored in multiple formats, consensus mechanisms may be switched in and out, and multiple MSPs are supported. Hyperledger Fabric also offers the ability to create channels, allowing you to create a separate ledger of transactions.

The following diagram illustrates how the Hyperledger Fabric CA server fits into the overall Hyperledger Fabric architecture:


Hyperledger Fabric Blockchain Network
Org1
Fabric CA
Fabric Peer (Endorsing Peer)
Chaincodes
Fabric Peer (Committing Peer)
Chaincodes
SafeNet HSMs
Clients (App/SDK) Nodejs/Java/Golang)
Channel
Fabric Orderer
SafeNet HSMs
Org2
Fabric Peer (Committing Peer)
Chaincodes
SafeNet HSMs
Fabric Peer (Endorsing Peer)
Chaincodes
Fabric CA
End-Users

There are two methods of interacting with a Hyperledger Fabric CA server:

> The Hyperledger Fabric CA client

> The Hyperledger Fabric SDK

The Hyperledger Fabric CA client or SDK may connect to a server in a cluster of Hyperledger Fabric CA servers (see the Cluster of Fabric-CA Servers in the diagram). In this configuration the client routes to an HA proxy endpoint which load balances traffic to one of the fabric-ca-server cluster members.

A server may contain multiple CAs. Each CA is either a root CA or an intermediate CA. Each intermediate CA has a parent CA which is either a root CA or another intermediate CA.

All Hyperledger Fabric CA servers in a cluster share the same database for keeping track of identities and certificates. If an LDAP is configured the identity information is stored by the LDAP, rather than the database.