Commercial Paper on Blockchain for NSD v2

Decentralized application manages instructions to transfer securities between members of NSD. See Functional Specification Google Doc.

Deployment with dockers run on separate hosts

Install prerequisites

- Clone Nsd Commercial Paper delivery packages from github:
```
git clone -b 2018_03-PRE_RELEASE_02 https://github.com/Altoros/nsd-commercial-paper
cd nsd-commercial-paper
./prerequisites-deployment.sh
```

On other Linux distros make sure these versions or higher are installed:

Docker version 17.12.1
docker-compose version 1.8.0
jq

To install them on Ubuntu 16.04 you nay use the following commands:

```
cd fabric-starter
./init-docker.sh
```

Now re-login to have user applied into docker group.

Next execute in console:
```
cd fabric-starter
./init-fabric.sh
```

##Configuration

For initial deployment the following organizations are used: - ORG1 – nsd - ORG2 – sberbank - ORG3 – mts

and the corresponded IP addresses: - IP1=91.208.232.164 - NSD node's IP - IP2=193.232.123.109 - Sberbank node's IP - IP3=213.87.44.178 - MTS node's IP

In Commercial Paper v2 installation NSD serves as MAIN_NODE which is configured as environment variable exported in files *env-common*. Other memebers are defined as THIS_ORG variable set correspondingly in *env-org-* files.

Check initial configuration or reconfigure organization names, and IP-addresses in configuration files:

Folder **nsd-commercial-paper**: - *env-common* - *env-org-sberbank* - *env-org-mts*

In these files the common variables and the current organization configuration are defined. The configuration specific variables are - `THIS_ORG`, `MAIN_ORG`, `IP_ORDERER`.

as well as initialization arguments for blockhains : - *instruction_init.json* - *book_init.json* - *security_init.json*

##Deployment:

At first each member has to generate their crypto material; it then will be exposed by http interface on port 8080 to be accessible by the other organizations:

1. Sberbank:
```
cd nsd-commercial-paper
source ./env-org-sberbank
./org-generate-crypto.sh
```

2. Mts: `cd nsd-commercial-paper`
```
source ./env-org-mts
./org-generate-crypto.sh
```

Here the first command `source env-org-<name>` loads the environment variables into the current (terminal) session in order to subsequent scripts use the necessary environment. The second command `./org-generate-crypto.sh` uses the loaded environment and generates crypto material by using Hyperledger Fabric utilities for current organization (which is defined by `THIS_ORG` environment variable). Finally this script starts a web-server (container) mapped to port 8080 which provisions the node's TLS and MSP sertificates thus allowing nodes securely communicate with each other.
The blockchain components are not started by this script.

Note: It's not completely necessary to generate crypto-material for all organizations to start. You may add organizations one by one. See section ***Adding new organization*** *then.*

After that the main org (NSD) starts the blockchain network, adds the members one by one and creates *common*, *depository* and bilateral and trilateral channels:

3. Nsd:
```
cd nsd-commercial-paper
source ./env-org-nsd
./main-start-org.sh
./main-register-new-org.sh $ORG2 $IP2
./main-register-new-org.sh $ORG3 $IP3
```

Note, when new organization is registered it's added to the list of existing organizations `env-external-orgs-list`. This list is used to automatically create tri-lateral channels with the new organization which is being added.
This list may be adjusted manually in the file to control trilateral channels creation.

If you are going to modifythe list of existing organizations make sure you created a backup copy of the file while all organizations are still in list. This file is also used during the smart-contract version upgrade process and this process should preferably be perfromed for all organizations, so smart-contract for all bi-lateral and tri-lateral channels are upgraded. In the other case the rest smart-contracts will have to be upgraded manually, which might be time-consuming process.

The first script `./main-start-org.sh` does several things. It's intended to be run ob a main node so it generates crypto material for the main organization as well as for the orderer. Then starts orderer and the main organization's blockchain components. It also creates the `common`, and `depository` channels and installsthe `security` and `book` chaincodes correspondingly.

The subsequent two scripts register new organizations in the blockchain, create bilateral channels `nsd-<org>` and installthe `position` chaincode there.

As it was mentioned the list of registered organizations is also adjusted. It's stored in file `env-external-orgs-list` and new organizations are automatically added to the list by this script. If the list contains one or more organizations (except `nsd`) the tri-lateral channels with newly registered organization will be automatically created.

Note: Before register new org adjust the file `instruction_init.json` and add requisites of new org there

On next step the members start the network on their nodes:

4. Sberbank:
   `./org-start-node.sh`

5. Mts (after Sberbank's run is finished):
   `./org-start-node.sh`

The `./org-start-node.sh` script is intended to be ran on member organizations nodes; it starts blockchain components (again based on the environment loaded on the previous steps) and join the organization to the `common` channel as well as bilateral channel with nsd `nsd-<this_org>`

Now newly started members need to join each other:

6. Sberbank:
   `./org-join-org.sh $ORG3 $IP3`

7. Mts (after Sberbank's joining is finished):
   `./ org-join-org.sh $ORG2 $IP2`

This script join an organization to another organization which parameters (name and IP address) are specified. then it joins the org to the tri-lateral channel with the `nsd` and the specified org. Finally it adds the ip address and network configuration information to the connectivity components

Adding new organization

To add new organization into the network the following steps need to performed:

1) Download NSD Commercial source package(s) to new org's server. See prerequisites section.

2) Environment file is created on new org with environment variables adjusted:
   edit `env-org-<neworgname>`: `...  THIS_ORG=neworgname`

3) New organization generates crypto-material:
   New org:
   `source ./env-org-<neworgname>`
   `./org-generate-crypto.sh`

4) On NSD server configure the initialization configuration for *instruction* chaincode:
   edit `instruction_init.json` (add new organization account information)

5) Register new organization in blockchain. Note bi-lateral and tri-lateral channels will be automatically created using the list of existing organizations (in the file `env-external-orgs-list`). This list can be modified accordingly if no all tri-lateral channels need to be created. But make sure to have a backup of the file with the complete list (see notes to the item 3 in the *Deployment* section).

   Nsd:
   `./main-register-new-org.sh neworgname <neworg_ip>`

6) Start blockchain on new org:
   New org:
   `./org-start-node.sh`

7) Mutually join new org to each existing organization (except NSD) and vice-versa:
   - Sberbank:
     New org:
     `./org-join-org.sh sberbank <sberbank_ip>`
     Sberbank: `./org-join-org.sh neworgname <neworg_ip>`

   - Mts:
     New org:
     `./org-join-org.sh mts <mts_ip>`
     Mts: `./org-join-org.sh neworgname <neworg_ip>`

   - Other org (if any):
     New org:
     `./org-join-org.sh <otherorg_name> <otherorg_ip>`
     Other Org: `./org-join-org.sh neworgname <neworg_ip>`

   …
   Repeat for all necessary organizations

##Upgrade smart-contracts to new versions

```
    Note: When you upgrade chaincodes to new versions they have to be re-
instantitated at each channel it's used.
    List if channels is based on the organizations attached to the network. So
before performing the upgrade it's highly
    recommended to restore from a backup the file `env-external-orgs-list` with
the full list of organizations.
```

Developer of blockchain (Altoros) pushes updated smart-contracts code into the repository and puts the git tag of form `2018_03-PRE_RELEASE_XX` where `XX` is a numbering sequence to keep a history of smart-contract which were deployed.

e.g.:

```
git tag --force 2018_03-PRE_RELEASE_02
git push --force origin 2018_03-PRE_RELEASE_02
```

Here XX equals 02.

```
After that the NSD blockchain network may be upgraded with new smart-contracts
without re-deploying the whole network.
To perform the upgrade network administrators select a unique label for the next
version which should be identical on each
organization for current upgrade. It is usually versioning numbers sequence in a
form of "1.0", "2.0", "3.0", but it might be
any label.
```

The following step has to be done on all nodes:

```
cd nsd-commercial-paper
./blockchain-upgrade.sh Y.Z XX
```

Here `Y.Z` defines the version label with which new smart-contracts will be installed. (Initial network deployment installs chaincodes with version 1.0).
`XX` - is the tag suffix

For example to upgrade network with new smart-contracts tagged with `2018_03-PRE_RELEASE_02` in github repository as chaincode version 2.0 the following command have to be executed:

- NSD:
  ```
  cd nsd-commercial-paper
  source env-org-nsd ./blockchain-upgrade.sh 2.0 02
  ```

- Sberbank:
  ```
  cd nsd-commercial-paper
  source env-org-sberbank ./blockchain-upgrade.sh 2.0 02
  ```

- MTS:
  ```
  cd nsd-commercial-paper
  source env-org-mts ./blockchain-upgrade.sh 2.0 02
  ```

##Add new organization to network after smart-contracts were upgraded

The starting organization script by default set the version of chaincodes to 1.0. If the whole network was already upgraded to another version new organization should be upgraded to the the corresponded version either.

So after starting the organization node, and joining organization-partners the same blockchain-upgrade procedure need to be executed.