Informix integration with BlockChain Smart Contracts – Demo instructions

This demo showcase Blockchain smart contract integration with Informix database. Demo application utilizes callback mechanism provided by blockchain database, and Smart trigger callback framework in Informix. This callback framework in Blockchain and Informix databases allow seamless integration between operations in blockchain database, and operations in Informix database.

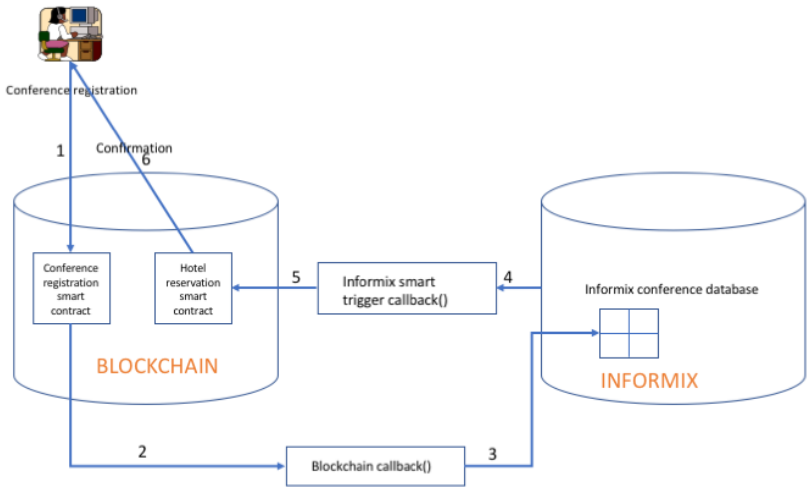
Demo application:

This demo creates two smart contracts within Ethereum blockchain: 1) Conference registration 2) Hotel reservation.
When a user tries to register for a conference, conference registration smart contract gets executed within blockchain network, and once blockchain confirms the transaction, callback function registered on conference registration smart contract gets fired. Within this callback, registration details are saved in Informix database.

Now smart trigger callback registered on Informix conference registration table gets executed which in turn execute hotel reservation smart contract within blockchain.

Smart contracts are written in Solidity smart contract programming language. Solidity is a contract-oriented programming language for writing smart contracts. It is used for implementing smart contracts on various blockchain platforms.
Reference document: https://en.wikipedia.org/wiki/Solidity

Informix connectivity is done through NodeJS, and JDBC driver for Smart triggers.



Blockchain:

A blockchain is a decentralized and distributed digital ledger that is used to record transactions across many computers so that the record cannot be altered retroactively without the alteration of all subsequent blocks and the collusion of the network. This allows the participants to verify and audit transactions inexpensively.
Reference document: https://en.wikipedia.org/wiki/Blockchain

Smart Contracts:

Blockchains can run code. While the first blockchains were designed to perform a small set of simple operations – mainly, transactions of a currency-like token – techniques have been developed to allow blockchains to perform more complex operations, defined in full-fledged programming languages.
Because these programs are run on a blockchain, they have unique characteristics compared to other types of software. First, the program itself is recorded *on* the blockchain, which gives it a blockchain's characteristic permanence and censorship resistance. Second, the program can *itself* control blockchain assets – i.e., it can store and transfer amounts of cryptocurrency. Third, the program is executed *by* the blockchain, meaning it will always execute as written and no one can interfere with its operation.
Reference document: https://www.coindesk.com/making-sense-smart-contracts/

Ethereum:

Ethereum is an open-source, public, blockchain-based distributed computing platform featuring smart contract (scripting) functionality. It provides a decentralized Turing-complete virtual machine, the Ethereum Virtual Machine (EVM), which can execute scripts using an international network of public nodes.

Reference document: https://en.wikipedia.org/wiki/Ethereum

Informix:

IBM® Informix® is a secure embeddable database, optimized for OLTP, IoT and is forging new frontiers with its unique ability to seamlessly integrate SQL, NoSQL/JSON, time series and spatial data. Reliability, flexibility, ease of use, and low total cost of ownership makes Informix the best in class enterprise database available in the market.
Reference document: https://www.ibm.com/analytics/us/en/technology/informix/

Informix Smart triggers:

Smart Triggers are special 'push' notifications from the Informix server to your Java application. This push notification is backed by a selective trigger that you create from your client application (it's not a normal database table trigger) that will push an event to the client when your trigger criteria happens. No more polling for events when you want to know if something has changed.
Reference document: http://www.informixcommunity.com/blogs/introducing-informix-smart-triggers

Truffle framework:

Truffle is a development environment, testing framework and asset pipeline for Ethereum, aiming to make life as an Ethereum developer easier.
Reference document: http://truffleframework.com/docs/

Demo instructions:

Step 1: Add node.js yum repository

```
$ yum install -y gcc-c++ make
$ curl -sL https://rpm.nodesource.com/setup_6.x | sudo -E bash -
```

Step 2: Install node.js and NPM

```
$ yum install nodejs
```

Step 3: Verify versions

```
$ node -v
v6.11.3
$ npm -v
3.10.10
```

Step 4: Install truffle ethereum development framework

Note: Requires NodeJS 5.0+, MacOS, Linux or Windows

$ npm install -g truffle

Step 5: Install EthereumJS TestRPC
Node.js based Ethereum client for testing and development

$ npm install -g ethereumjs-testrpc

Step 6: Install openjdk

$ yum install java-1.8.0-openjdk-devel.x86_64

Step 7: Install nodejs driver for java

$ npm install -g java

Step 8: Install Informix NodeJS Driver

$ npm install -g ifxnjs

Step 9: Create smart contract project using truffle:

$ mkdir informix-blockchain-demo
$ cd informix-blockchain-demo

$ truffle init
Downloading project...
Project initialized.

Step 10: Copy 12.10xC10 or later version JDBC driver ifxjdbc.jar to informix-blockchain-demo

Step 11: Copy Conference.sol to informix-blockchain-demo/**contracts**

Step 12: Copy Hotel.sol to informix-blockchain-demo/**contracts**

Step 13: Copy conference.js to informix-blockchain-demo/**test**

Step 14: Copy hotel.js to informix-blockchain-demo/**test**

Step 15: In new terminal, start ethereum clinet testrpc:

$ testrpc

EthereumJS TestRPC v4.1.3 (ganache-core: 1.1.3)

Available Accounts
==========
(0) 0x77c964c76963e559e4d1212cea270c5fd747f51c
(1) 0x41ec38e4c274254cd56821f201b4da8b5ef1625b
(2) 0x4f399ba5d5e492ad2211708022dcfffdf4c0fd03
…. …..
…. …..

Step 16: Connect to informix server, and create the following schema:

create database conference with log;
create table registration (id bigserial, address char(1024)) lock mode row; -- registration details
create table contract_addr (owner char(1024), conaddr char(1024)) lock mode row; -- smart contract address

Step 17: Update Informix connectivity credentials in test/conference.js
Update server name, hostname, user name and password:

global.ConStr =
"SERVER=myserv;DATABASE=conference;HOST=172.20.0.10;SERVICE=60000;PROTOCOL=onsoctcp;UID=informix;PWD=changeme;";

Step 18: Update Informix connectivity details and JDBC jar file location in test/hotel.js
Update hostname, port number, user name and password:

global.jdbcurl = "jdbc:informix-sqli://172.20.0.10:60000/sysadmin:USER=informix;PASSWORD=changeme";

global.jdbcjarfile = "/opt/ibm/data/informix-blockchain-demo/ifxjdbc.jar"

Step 19: Compile Smart contrats:
cd informix-blockchain-demo
truffle compile
Compiling ./contracts/Conference.sol...
Compiling ./contracts/Hotel.sol...

Step 20: Start two new terminals and set these environment variables:
export NODE_PATH=`npm root -g`:$NODE_PATH
export
LD_LIBRARY_PATH=${INFORMIXDIR}/lib:${INFORMIXDIR}/lib/esql:${INFORMIXDIR}/lib/cli

export PATH=$INFORMIXDIR/bin:$PATH

Step 21: In terminal 1, execute hotel reservation smart contract.
Note: Contract execution blocks till conference registration smart contract was executed

$ truffle test test/hotel.js

Using network 'development'.

Contract: Hotel
Created Hotel Reservation SmartContract.
Register Informix smart trigger on conference registration table and wait for event data
✓ ==>Check and create Hotel Reservation smart-contract! (74ms)

Event Document:
{"operation":"insert","table":"registration","owner":"informix","database":"conference","label":"smart-trigger","txnid":974977205456,"operation_owner_id":200,"operation_session_id":5940,"commit_time":1507220021,"op_num":1,"rowdata":{"id":22,"address":"0xe207654d70a33c320f643ca739ebb71e4476a5f4" }}

Add customer account 0xe207654d70a33c320f643ca739ebb71e4476a5f4 to Hotel Reservation Smart Contract account
✓ ==>Wait and reserve hotel for conference attendee! (9141ms)

2 passing (9s)

Step 22: From terminal 2, execute conference registration smart contract:

$ truffle test test/conference.js

Using network 'development'.

Contract: Conference
✓ ==>Get Conference contract address from Informix!
Conference contract already deployed at address: 0x22d9ed8b971a0c70001de8928a5487c6008a6578
✓ ==>Check and create conference contract!
BlockChain callback: New smart contract registration event received! {"_from":"0xe207654d70a33c320f643ca739ebb71e4476a5f4","_amount":"50000000000000000"}
Informix: Added new registration record to Informix DB!!