CRYPTO-CURRENCY EXCHANGE eKYC

Electronic know your customer

Business Requirements Document (BRD)

V.0.4

February 14, 2022

**Document Information**

**Document History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 14/02/22 | 0.1 | Initial Draft | Christopher/Bening |
| 19/02/22 | 0.2 | Working Draft | Christopher |
| 21/02/22 | .03 | Working draft | Christopher/Beining |
| 23/02/22 | 0.4 | Working Draft | Christopher/Bening |
|  |  |  |  |

**Definitions, Acronyms and Abbreviations**

| **Term** | **Explanation** |
| --- | --- |
| eKYC | Electronic Know Your Customer |
| DLT | Distributed Ledger Technology |
| BRD | Business Requirements Document (this document) |
| FINTRAC | Financial Transaction and Reports Analysis Centre of Canada |
| VERIFF | Refers to the company that e-verify digital identity |
| AML | Anti-Money Laundering |
| CFT | Combating Finances of Terrorism Rule |
| WAD | Wallet Address |
| MSP | Membership Service Provider |
|  |  |
|  |  |
|  |  |

**References**

| **Name** | **Description/Path** |
| --- | --- |
| Shared Extranet Site | Shared repository for key documents, deliverables, and action items:  <https://www.veriff.com/case-studies/veriff-plutus-partnership> |
| Non Functional Specifications | Document outlining technical requirements and quality attribute |
| Platform Recommendations |  |
|  |  |

**Introduction**

**What This Is**

The purpose of the Business Requirements Document (BRD) is to express the requirements of the customers and stakeholders to be served by the deliverables of the project—the perceived customer wants and needs for a product, system, or service.

The primary audience for the document is the project team, and specifically, the functional groups that must determine how to implement the application, system, service, etc. to meet the customers’ requirements. The team is responsible for determining how to develop what the Business has requested.

The BRD distinguishes between the business solution and the technical solution. When examining the business solution, the BRD should answer the question, “What does the business want to do?” For example, the business wants to have the ability to know who is transacting business on their portal so that they can forward when requested to regional authorities, when or if the information is requested.

On the part of business, it is part of their compliance signed when applying for a license of operation in a certain jurisdiction.

Some jurisdictions will ask only for Place of residence, Identification of place of residence, and transaction history.

Other jurisdictions will ask for a selfie with an IID card held close to your chest with a white paper indicating the date, when the photo was taken.

The business requirement documents most state specifically what type of process has been ordered, evaluate, and get a final sign-off on the decision of the customer.

**Requirements**

**Executive Summary**

The first transactional exchange of Bitcoin happened in 2010 when two pizzas were bought for 10,000 BTC, which now equates to over $450 million. However, despite the nature of that very first purchase, cryptocurrencies are still more commonly used as an investment rather than for daily spending. This is partly because it is impossible to spend cryptocurrencies at your local stores. Secondly, there are no known pronouncements by governments around the world on their stands on Cryptocurrency Currently, every day retailers simply don’t accept cryptocurrencies as a form of payment, and this is something that could take years to overcome. All intelligent reports point to the fact that sooner than later government will make it’s stand known. For now, the early adopters, investors and stakeholders alike are busy figuring out how to solve some immediate problems of cryptocurrency exchange, transactional details, and event logs. To this end, we are proposing a Hyperledger fabric to create a robust solution for various cryptocurrency exchanges who are ready to go into the mainstream of satisfying one on the most critical part of exchange without a third party,

eKYC using the verify solutions will allow the crypto exchange to be able to verify users’ identities while keeping in line with the government requirements of know your customers.

**The Problems**

* Various Governments around the globe are struggling with what to do with the new innovation. A wait-and-see approach has been taken by most of them with the exemption of India, and of late Canada that has taken a drastic step in banning or confiscating funds made from crypto. In the case of Canada, it took a long time to intervene during a siege in the country’s capital simply because they do not know how the money is being laundered or identify does behind it.
* . As highlighted in the paragraph above, it is easy and convenient pseudo to send money anywhere across the globe using cryptocurrency this made the government un-easy since it could be used for money laundering and any other crimes.
* On the user’s side, it is very difficult to get loans, or file returns and, for example, HST/GST when your government does not recognize it as a form of business.

**KYC Challenges**

The KYC project are often a headache for cryptocurrency exchange firms and the complexity is driven by multiple factors the following are some of the lists of constraints that affect KYC.

* eKYC project requires huge investment
* The absence of a common standard in the KYC documentation process
* No end in sight for eKYC since after eKYC the exchange is obligated to monitor and update changes to any changes in KYC data..
* Elaborate documentation
* There are geographical challenges involved
* Dropping out of compliance process
* Delay in onboarding

**eKYC Solution**

In order to solve the above-mentioned challenges, there is a need to covert the analog KYC process to digital, this will streamline the process and make it more user-friendly, re-usable, recallable, and in fact cost-effective.

This can be achieved by creating an ecosystem of digital data and system providers that will ensure easy and direct proceeding for onboarding and compliance.

**eKYC List of Features**

* Upload records: Records can be uploaded in any format(doc, pdf, jpg, etc.) up to a maximum of 10 MB per second. These records are automatically encrypted using an AES symmetric encryption algorithm where decryption keys are automatically stored in the exclusive web application of the uploading entity.
* Searching for records based on index peculiar to companies and viewing relevant metadata
* Whitelisting entities so that they can download all records uploaded by white-lister.
* Ability of clients to be able to approve or remove information from registered institution to access data

**Hyperledger Fabric**

Hyperledger Fabric started in December 2015 with IBM. The objective is to build a permission blockchain focused on data privacy, needs-based information sharing, the immutability of data and security

The DLT that will be used for this project is the chaincore, chain.com which is used by financial institutions, gift cards, loyalty points issuers

Most stable product release v 2.2 from Hyperledger projects –for developing applications and solutions

Based on IBM’s open blockchain and Digital Asset Holdings and Chaincode development effort

Using Hub and Spoke network model

Key elements are

* Channels
* Chaincode
* Ledger
* CouchDB
* Network
* Ordering Service
* Ordering style
* Membership Service Provider (MSP)
* Certifying authority is pluggable thru Fabric-CA-API
* Define Signing algorithm and Verify algorithm
* Consensus and identity management algorithms are plug and play

**Web App Interface and Roles**

The application that we are building will have permission system-based roles with the use of hyper ledger fabric to control transactions, approve transactions and also used to establish a new rule.

The functionality will allow :

* Stroring eKYC information of user
* Putting out a request for the eKYC user
* Responding to request
* Responding by providing only required information

**eKYC User Diagram**

Graphical user interface

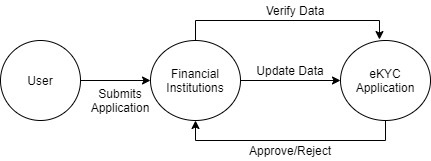
Description automatically generated

**eKYC State Diagram**

Diagram

Description automatically generated

**Scenarios**

[](https://user-images.githubusercontent.com/45354395/113464817-89a17180-93fd-11eb-88ea-1ef9146b94c1.png)

**Organizations**

Org 1 – Crypto Exchange/Bank 1 Org 2 – Fintrac 2

**Roles**

* User – Any user can create account which will be save in CouchDB
* Org 1 Admin – Users from Org 1 who verify data and approve or reject an application
* Org 2 Amin – User from Org 2 who can verify data and approve or reject an application

**State Data**

* State: Register; Not registered, Approved or Rejected
* Name: First Name; Middle Name; Last Name
* Date of Birth; Most be over 16 years of age
* Address: Within the Jurisdiction where ID is issued
* Identification Permitted: Citizenship Passport, Drivers License, Permanent resident card

**Transition**

* RequestInfo by uniqueID(Name, Date of birth, Identification number)
* Registration(Name, Date of birth, Address, Identification number)
* Incomplete(Name)
* RequestDenied(Name)
* Approved(Name)

**State Data**

* User – Any user can create account which will be save in CouchDB
* Org 1 Admin – Users from Org 1 who verify data and approve or reject an application
* Org 2 Amin – User from Org 2 who can verify data and approve or reject an application

**Flow Chart**

Diagram

Description automatically generated

**Contract Definition**

//Initialize Contract User – userId, age, gender, Permanent house address Output msg.sender is made as creatorAdmin msg.sender is made as superAdmin msg.sender is made as Approved

//Request a new admin requestNewAdmin - address, userId, age, gender, Permanent house address msg.sender should be admin address and userId should not be already registered Output – marks userId, state as pending, level as admin, age, gender, Permanent house address, createdBy as msg.sender, createdAt as current time.

//Request a new user requestNewUser - address, userId, age, gender, Permanent house address msg.sender should be admin address and userId should not be already registered Output – marks userId, state as pending, level as user, age, gender, Permanent house address, createdBy as msg.sender, createdAt as current time.

//Approve a pending request ApproveUser – address msg.sender should be Superadmin Output – state as Approved

//Get a user detail getUserDetails – address Output – userId, state, level, age, gender, Permanent house addres

**Data Base Data**

* User Schema
* Login
* Password
* User type
* Client
* Ledger id
* Id generated ledger
* Registered person
* Clients only field
* orgNum and ledger user fields encrypt and saved
* Org Credentials
* Crypto exchange entry field
* OrgNum and ledger field encrypt and saved

**Data Base Data**

* /client/create
* method: POST
* params: login, password, name, date of birth, address, id number
* description: register new client
* chaincode: calls 'createClient' function
* /client/login
* method: POST
* params: login, password, user type
* description: client login
* /client/getClientData
* method: GET
* params: ord number, ledger user, ledge ID
* description: return the data from the client chaincode: calls 'getClientData' function
* /client/approve
* method: POST
* params: org number, ledger user, ledger ID, crypto exchange ID
* description: approve a crypto exchangeto read client data
* chaincode: calls 'approve' function
* /client/remove
* method: POST
* params: org number, ledger user, ledger ID, crypto exchange ID
* description: remove a crypto exchangeto read client data
* chaincode: calls 'remove' function
* /client/getApprovedFis
* method: GET
* params: org number, ledger user, ledger ID
* description: return a list of approved crypto exchanges by the client
* chaincode: calls 'getRelationByClient'
* /fi/create
* method: POST
* prams: login, password, name, id number
* description: register new crypto exchange
* chaincode: calls ‘createCrypto exchange’
* /fi/createClient
* method: POST
* params: login, password, name, date of birth, address id number, org number,
* ledger user
* description: register new client
* chaincode: call ‘createClient’
* /fi/login
* method: POST
* params: login, password, user type
* description: crypto exchangelogin
* /fi/getFiData
* method: GET
* params: org number, ledger user
* description: return the data from the crypto exchange
* chaincode: calls 'getCryptoExchangeData' function
* /fi/getClientData
* method: GET
* params: org number, ledger user, client ID, fields wanted
* description: return the data from the client
* chaincode: calls 'getClientData' function
* /fi/getApprovedClients
* method: GET
* params: org number, ledger user
* description: return a list of clients that approved this crypto exchange
* chaincode: calls 'getRelationByFi'

**Ledger Data**

* Client
* name: Name of the client
* ID front
* dateOfBirth: Date of birth of the client
* address: Address of the client
* idNumber: Some identification number of the client
* ID back
* whoRegistered: { orgNum: number of the organization that registered the client,
* ledgerUser: name of the ledger user that registered the client}
* Crypto exchange
* name: Name of the crypto exchange
* idNumber: Some identification number of the crypto exchange
* orgCredentials: { orgNum: number of the organization, ledgerUser: name of the
* ledger user of that organization}

**Chaincode functions**

* initLedger(ctx)
* Initiate ledger storing crypto exchangeand client data
* getCallerId(ctx)
* Internal function - return the ledger user that called the method
* isWhoRegistered(ctx, clientId)
* Internal function - return the ledger user the registered that client
* createClient(ctx, clientData)
* Create a new client. Who registers the client is who called the method
* getClientData(ctx, clientId, fields)
* Return client data. A list of fields wanted is passed as a parameter
* getFinancialInstitutionData(ctx)
* Return crypto exchangedata
* approve(ctx, clientId, fiId)
* Approve a crypto exchangeto access client data
* remove(ctx, clientId, fiId)
* Remove crypto exchangeaccess data approval
* getRelationsArray(ctx, relationResultsIterator)
* Internal function - iterate a composite key iterator
* getRelationByClient(ctx, clientId)
* Return a list of approved crypto exchangeby that client
* getRelationByFi(ctx)
* Return a list of clients who approved the caller FI
* queryAllData(ctx)
* Return a list of all data stored in the ledger

**Approvals**

* Composite key clientId~fiId
* Maps all FIs approved by the client
* Composite key fiId~clientId
* Maps all clients that approved that FI

**Diagram

Description automatically generated**

**Stake Holders**

**TBD**

**Precedence and Priority of Features**

| **ID** | **Requirement Summary** | **High** | **Med** | **Low** |
| --- | --- | --- | --- | --- |
|  | *Users need to be able to search and email legislators in their aread* | ***x*** |  |  |
|  | *Action and donate pages need to load quickly on mobile devices* | ***x*** |  |  |
|  | *Training* |  |  | ***x*** |
|  | *Research depository* | ***x*** |  |  |
|  | *Login function (3 access levels)* |  | ***x*** |  |
|  | *Guided Search* | ***x*** |  |  |
|  | *Beta demo* |  | ***x*** |  |
|  | *Chapter sites* |  | ***X*** |  |
|  | *User Acceptance Testing* | ***x*** |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Sign Off**

EKYC Communications Director, Dave Date

Definition 6 Account Director, Deepaak Date