**Stellar Lumens Integration into Exchange´s Trading Platform**

**TECHNICAL ARCHITECTURE**

Draft Version ● DECEMBER 23, 2018

**Table of Content**

1. Overview
2. How to Integrate Stellar to the exchange
3. Hardware Requirements
4. Public Horizon Servers REST Integration
5. Accounts Setup
6. Integration Points
7. Seed, Public and Private Keys
8. Multisignature
9. References

**1. Overview**

This document is an architectural overview for Stellar Integration to an Exchange’s trading platform and wallet.

**2. How to Integrate Stellar**

Stellar can be Integrated by using instances of Stellar Core and Horizon, or using Public REST API Horizon.

**Setup**

Stellar can be integrated by using:

* Own Instances of Stelar Core and Horizon
* Public Horizon Servers

Is recommended to run own instances of Stellar Core and Stellar Horizon.

**3. Hardware Requirements**

Hardware requirements *will grow along with usage* of the Stellar network. These are recommended hardware requirements listed for each service.

# **Nodes**

## Stellar-Core

Instances of Stellar-Core are part of the network as a node and therefore need to be large enough to support the volume on the network.

### Recommended

**CPU**: 8-Core (16-Thread) Intel i7/Xeon or equivalent (c5.2xlarge on AWS)  
**RAM**: 16GB DDR4  
**SSD**: 120GB

## **Horizon**

Instances of Horizon need to be large enough to support ingesting all of the latest transactions on the network.

**CPU**: 16-Core (32-Thread) Intel i7/Xeon or equivalent (c5.4xlarge on AWS)  
**RAM**: 32GB DDR4  
**SSD**: 120GB

**DB Usage**

*There is a significant amount of computation that is done on the DB side of Horizon, these requirements are only for the application side of horizon.*

*You will need to account for using a larger machine if using the same machine for the DB, or a separate machine for the DB altogether.*

**4. Public Horizon Servers REST Integration**

Another way to integrate is using REST APIs from public Horizon servers at Stelllar.org:

test net: {hostname:'horizon-testnet.stellar.org', secure:true, port:443};

live: {hostname:'horizon.stellar.org', secure:true, port:443};

**REST API** **:** Horizon RESTful API allows client applications to interact with the Stellar network. There are several Stellar SDKs avaliable (mainly Javascript, Java and Go) that can be used to generate keys , accounts , transactions and interact with the Stellar Ecosystem.

More info at:

https://www.stellar.org/developers/horizon/reference/index.html

### **5. Accounts Setup**

There are two types of account for handling Hot Wallet and Cold Walllet cases:

* **Issuing Account**
* **Base Account.**

These account types simplifies handling Hot Wallet and Cold Walllet cases:

### **Issuing account**

Account used to keep the majority of customer funds secure

Is an account whose secret keys are not on any device that touches the Internet.

Transactions are manually initiated by a human and signed locally on the offline machine.

This design makes the issuing account secret key much harder to compromise.

### **Base account**

A base account contains a more limited amount of funds than an issuing account. A base account is a Stellar account used on a machine that is connected to the Internet. It handles the day-to-day sending and receiving of lumens. The limited amount of funds in a base account restricts loss in the event of a security breach.

### **CustomerID – Incoming Deposits match**

A CustomerID is used to match incoming deposits with exchange internal user account.

### **Database Setup**

* Need to create a table for pending withdrawals.
* Need to create a table to hold the latest cursor position of the deposit stream
* Need to add a row to your users table that creates a unique customerID for each user.
* Need to populate the customerID row.

### **6. Integration Points**

Main integration points are:

* Listening for deposit transactions from the Stellar network
* Submitting withdrawal transactions to the Stellar network

### **7. Seeds, Public and Private Keys**

Stellar accounts uses a seed to generate public and private keys.

The seed is used instead of the private key for convenience.

To have full access to an account, you only need to provide a seed instead of both a public key and a private key.

*Stellar uses ED25519 keypairs. There is only one public key that matches the given secret key.*

### **Accounts Creation**

To create an account you start by creating your own seed and key .

Once you have a seed and public key, you can create an account.

each account must have a minimum balance of 1 lumen ( for preventing unnecessary accounts).

When the account is created, you’ll send only the public key to a Stellar server.

### **8. Multisignature**

Stellar uses **signatures** as authorization. Transactions always need authorization from at least one public key.

Transactions may need more than one signature. If the transaction has operations *that affect more than one account*, it will need authorization from every account in question.

## **Thresholds**

Operations fall under a specific threshold category: low, medium, or high.

This threshold is the amount of signature weight required to authorize an operation at that level.

Once the signature threshold is met (combination of both signer’s signatures weight) the transaction is authorized.

## **Signers**

refers to the master key or to any signing keys added later. A signer is defined as the pair: public key, weight.

Multisig Code sample at:

https://github.com/iveltondequeiroz/StellarChallenge

### **References**

* https://www.stellar.org/developers/