StartEngine LDGR™ WHITEPAPER

NO MONEY OR OTHER CONSIDERATION IS BEING SOLICITED, AND IF SENT IN RESPONSE, WILL NOT BE ACCEPTED.
NO OFFER TO BUY THE SECURITIES CAN BE ACCEPTED AND NO PART OF THE PURCHASE PRICE CAN BE RECEIVED UNTIL THE OFFERING STATEMENT FILED BY THE COMPANY WITH THE SEC HAS BEEN QUALIFIED BY THE SEC. ANY SUCH OFFER MAY BE WITHDRAWN OR REVOKED, WITHOUT OBLIGATION OR COMMITMENT OF ANY KIND, AT ANY TIME BEFORE NOTICE OF ACCEPTANCE GIVEN AFTER THE DATE OF QUALIFICATION.
AN INDICATION OF INTEREST INVOLVES NO OBLIGATION OR COMMITMENT OF ANY KIND

Summary

LDGR™ is a Regulation Crowdfunding, Regulation D, and Regulation A compatible security token for issuing and trading securities. LDGRToken is an ERC-20 compatible token that complies with the new Securities Act Regulations: Regulation Crowdfunding, Regulation D, and Regulation A.

Table of Contents

Introduction

With the launch of Regulation Crowdfunding and amendments to Regulation A, investors who have purchased securities have the right to sell them to other investors under the conditions specified in the purchase agreement, without needing to register the transaction with the Securities Exchange Commission (SEC). Liquidity is a critical issue for ordinary investors who purchase securities in privately held companies. They are not necessarily able to wait years before having the opportunity to sell their securities, and in some cases their financial situation requires them to sell them immediately. Solving this issue is critical to allowing these new regulations to continue to provide corporations the capital they need to grow their businesses.

Peer to Peer trading with the Blockchain (patent pending)

The ability to store documents digitally has become the norm in our new digital economy. However, the securities industry has been slow to adopt new digital technologies that enhance its service and reduce costs to its customers. The ability to store securities digitally with a central database has raised a new set of concerns about security of the securities and--more importantly--personal data. There have been many well-documented hacks of personal information from both government and banking databases. The security risks in the financial industry are higher and the consequences very serious. Consumers are worried about their data being stolen and then sold to other fraudsters, while financial institutions are worried about their reputation and high costs to restitute investor losses.

The advent of the new cryptocurrency and digital ledger technology called the Blockchain offers a very secure and cost effective solution for providing security to the consumer and lower costs for the industry. The securities can be held securely using end-to-end encryption, yet openly authenticated, referenced and documented so that ownership of the securities can be trusted as reliable. In the past, this type of technology would have been extremely expensive to build and manage, but today it it can be implemented and deployed relatively easily at little cost. The winners are investors, who can 1) rely on the security's ownership authentication; 2) reduce their costs for maintaining their accounts, and 3) sell securities when they would like to.

StartEngine is building a system to facilitate the recording of ownership and transfer of securities sold under the new Securities Act Regulation Crowdfunding, Regulation D, Regulation S and Regulation A, and the trading of those securities to other investors without the fear of the loss of data or the high costs that are ordinarily imposed on trading of those securities. StartEngine intends to create a

new cryptocurrency for each company that raises capital using these above regulations. Each individual security sold is represented as one coin in the Blockchain. For example if an investor purchases 100 shares in a company then those shares are represented as 100 tokens. If an investor lends a company $1,000 in a loan then the loan is represented as 1,000 tokens. StartEngine hopes this system can reduce the overall cost of trading securities, and by enhancing security and greatly improve the experience for investors. The buyer can use fiat, Bitcoin or ether to purchase the securities and the seller can either keep the currency tendered or convert them into other currency.

Background

Financial Regulation

Since the creation of the Securities Exchange Commission and the enactment of the Securities Act of 1933, companies generally needed to register their offerings and securities with the SEC if they wanted to sell securities to the general public. Without registration, companies could only sell their securities if the transaction complied with an exemption from registration, such as that available under Regulation D, which provides a relatively easy to comply with exemption from registration if securities are only sold to accredited investors: those with incomes of over $200,000 or net worth (excluding homes) of over $1 million.

This changed in April 2012, when president Barack Obama signed the JOBS ACT, a bipartisan new financial law designed to help small businesses raise the capital and hire the people they need to grow and build a successful business. The JOBS ACT changed the system by requiring the SEC to implement or amend its regulations to permit greater ability for companies to sell their securities directly to the general public. Three regulations have emerged from the JOBS ACT that are being used by companies to raise capital online:

1. New Regulation Crowdfunding, which permits companies to raise capital from the general public up to $1,070,000 per year upon filing a Form C with the EDGAR SEC public database. This regulation is gaining strong popularity since its launch in May 2016 with more than XX companies having currently filed and raised close to $YY.
2. Amendments to Regulation A, popularly referred to as “Regulation A+,” which permits companies to also raise capital directly from the general public up to $50M per year and requires the company to file the Form 1A for review and commenting by the SEC. In order to proceed, the offering must be qualified by the SEC. This process is more expensive and it can take several months to get to qualification. Since 2015, hundreds of companies have filed a form 1A with the SEC.
3. New Regulation D 506(c), which permits companies to raise capital directly from accredited investors with no limit. The company files a form D with the SEC and States after the close of the raise. This process is not expensive but it is limited to accredited investors and the securities are “restricted securities,” meaning that for at least a year they cannot be easily traded.

The Capital Market for Small Businesses

Small businesses with under 50 employees are the backbone of the US economy, generating more jobs than medium and large businesses. There are more than five million businesses operating in the United States, and many of them seek capital for equipment, purchasing inventory, improving a property, leases and hiring employees. However, because they are small, getting access to capital is very hard. Banks typically do not lend to small businesses because they do not have tangible assets or sufficient track record of profits, Venture Capitalists provide around $70B in capital every year, but they only invest in a few thousands of companies a year, and are biased towards highly educated and experienced management teams in narrow fields of the industry. Finally, angel investors provide somewhere around $40B year in capital to tens of thousands of companies who are able to find these wealthy accredited investors. Unfortunately, only a small portion of the population is accredited and many of those do not invest in high risk companies as they are generally reaching the age of retirement and seeking more income-generating investments such as real estate or municipal bonds.

With the JOBS ACT, the general consumer has access to startups and small businesses who can take advantage of these new regulations. However, one issue those regulations do not address is liquidity. Even successful startups do not typically provide any liquidity until 5 to 7 years after it is founded. Other small businesses have longer business cycles where investors have to be very patient. While accredited investors are used to having their capital locked for a long period of time with the hopes of a strong return, ordinary investors are probably not able to wait this long.

If a company decides to go public they typically hire an investment bank and file a registration statement on Form S-1 with SEC. Once that registration statement is effective, they sell securities to a set of investors during the Initial Public Offering and then list their securities on a national market such as the NASDAQ or the New York Stock Exchange. After listing, any investor can purchase securities from other investors through any registered broker-dealer. The share price fluctuates based on the supply and demand for these securities, and there is an entire industry catered to analyzing and reporting on these listed companies. The cost to trade these securities used to be very expensive until the discount brokers appeared on the market, followed by online trading websites. Today, it costs just a few dollars to execute a trade.

There are half the number of listed companies today compared to 20 years ago. To go public in the traditional way, a company needs to be very large and raise hundreds of millions of dollars to interest any of the large investment banks. A small business has no chance to raise capital this way. With the changes implemented by the JOBS ACT companies can raise money from the general

public, but there is no way to provide liquidity to the investors by listing shares on a national exchange. The smaller trading forums such as the OTCQX market have large requirements such as a minimum of $2M in assets, and a company must have a financial sponsor or investment bank willing to be a market maker for the company's securities - another barrier for the small businesses.

Investors who purchase securities in a company who is using Regulation Crowdfunding need to wait one year before they can easily sell their securities to other investors. With Regulation A+ there is no waiting time. However, only a few companies have listed their securities with any kind of exchange or trading forum. The investor has no ability to sell their securities unless they find themselves an investor who wants to purchase them. This lack of liquidity is a hurdle for these ordinary investors and until an effective solution exists, it will slow down these investors and small businesses will have harder time raising capital which defeats the purpose of having these new regulations.

Blockchain Technology

Blockchain technology is the backbone of cryptocurrencies such as Bitcoin and Ether. However, as a 'distributed ledger,' blockchain also has a vast number of additional applications. Smart contracts, or code executed on a blockchain, bring significant advantages over existing database-centric applications. Open blockchains such as the one used by Bitcoin offer these benefits:

- Low-cost: There is no middleman to impose fees. Transfers require only small transaction fees paid to the blockchain miners.
- Immutability: The ledger is policed by every member in the network and its integrity is checked and agreed by the network as a whole on an ongoing basis. Any changes that a minority party attempts to make to the blockchain are recognized and rejected by the majority.
- Transparency: Everything that takes place on the ledger is visible to everyone. It is possible to see everything that has been recorded since the ledger's beginning on the blockchain.
- Irreversibility: Because the ledger is immutable, a transfer that has been accepted into the blockchain cannot be reversed.
- Security: Because the blockchain is maintained by a large network of participants, no hacker can get enough influence to submit a fraudulent transaction. The more valuable the tokenized security or cryptocurrency, the larger the network and therefore the more resources it would take for a hacker to be able to enter into fraudulent behaviors.

The first application of the blockchain was Bitcoin, first introduced in January 2009 as an innovative and secure currency. Without the blockchain, owners of Bitcoin could not be guaranteed that someone else could not copy their Bitcoin and 'double spend.' Normally that occurs via an intermediary such as a bank, and that is where the costs and delays occur. The intermediary may prove to be

untrustworthy and can reverse a transaction without the user's permission. What's more, there is a single point of failure and the bank can be hacked or the transaction intercepted by hackers. Bitcoin needed a solution to reducing the costs of transferring Bitcoin to other parties, that eliminated security risk and virtually all cost. The Bitcoin system that emerged is a peer-to-peer shared ledger and therefore is secure without the need for an intermediary.

The initial innovation of the blockchain is that it is very difficult to add a Bitcoin transaction to the blockchain but extremely easy for anyone to check if a transaction is valid. Fraudulent transactions are quickly identified and prevented from entering the blockchain.

The second innovation of the blockchain is smart contracts. The transfer of funds does not involve moving Bitcoin from one database to another; rather it updates the ledger to reflect how much Bitcoin is owned by each address. Using the same technology of transparency, security and immutably by consensus of the entire network, this can be extended to the trading of securities issued under Regulation Crowdfunding and Regulation A. Smart contracts are code executed on the blockchain that enable the additional complexity required for managing tokenized securities (versus managing a currency).. With smart contracts, the securities industry can create decentralized apps--known as "dapps"--for a number of applications which require the security and low cost that the blockchain can offer.

LDGR Trading Use Case

An Alternative Trading System (ATS) is a broker dealer who uses the blockchain to efficiently and securely manage the trading of securities for Regulation Crowdfunding, Regulation D, Regulation S and Regulation A. A Registered Transfer Agent (RTA) manages the ownership and records the transactions of the investors who purchased securities under Regulation Crowdfunding, Regulation D, Regulation S and Regulation A. The RTA is a trusted source for the ATS. The ATS is to use the LDGR blockchain technology to efficiently and securely manage the trading of securities for Regulation Crowdfunding, Regulation D, Regulation S and Regulation A. The trading of securities is subject to SEC and State blue sky laws.

An investor who has purchased securities under these three rules can choose to sell these securities on an ATS by posting their proposed offer using a set of possible pricing models. Buyers can set a buying limit order setting a ceiling to the price of a purchase of securities. Sellers can set a selling limit order setting a minimum price to sell the securities.

Once posted, the ATS will verify with the RTA if the sellers owns free and clear the securities and they can be sold according to any applicable transfer restrictions. The ATS places a marker to block any further transactions for these securities. This function requires

the RTA to permit a portion or all of the securities to be restricted and not sold by other trading platforms or broker dealers for a determined amount of time. If securities are locked by the ATS, they can only be unlocked at anytime by the same ATS; or if 7 days passes they are automatically released. If an investor tries to sell any securities which are locked then the RTA will not accept the transaction until they are released by the ATS who requested a lock of those securities.

The buyer can now purchase the securities using fiat, Bitcoin or Ether. This is done peer-to-peer without intermediaries. Once completed the the ATS updates the RTA with the identity of the new owner and the number of securities purchased. The RTA is the trusted source for executing the LDGR smart contract.

This peer to peer model is innovative and secure because it uses the blockchain and the Bitcoin cryptocurrency for payments.

An investor can sell securities and buy securities through the ATS while keeping its money in Bitcoin and incurring low transaction fees. The RTA charges the seller a small fee for each transaction. With this innovative system, an investor can make purchases and sales nearly instantly with low fees. Others, if they want, can convert their money back into dollars or fiat.

The identity of the owners of securities is known to the ATS and the RTA but it is not visible on the blockchain which encrypts this information.

StartEngine LDGR™

StartEngine LDGR™ is a method that allows companies (“Issuer” or “Issuers”) to list in the blockchain each Regulation Crowdfunding, Regulation A+, Regulation D securities or Regulation S sold in an offering. StartEngine LDGR is owned by StartEngine Crowdfunding Inc, a Delaware Corporation (“StartEngine”). StartEngine LDGR requires the Issuer to select and hire the following vendors:

- Registered Transfer Agent (“RTA”) which records the identity of each investor and amount of securities sold in the offering or transferred. The RTA has a contract with one or more ATS to approve the transfers.
- Alternative Trading System (“ATS”) which permits the secondary trading of securities subject to SEC and State blue sky rules.

Each security offering has its own symbol represented by a unique four to eight letter/number combination. Each class of securities is represented by the symbol followed by a period, then by a letter representing the security class, then by a period, and then a cohort

number. For example, preferred StartEngine shares issued as the second series would be “STAR.P.2”. Restricted common shares would be “STAR.C.1” and regular common shares would be “STAR” because they are unrestricted common shares. The RTA would convert the preferred at the request of the investors in to common by burning the “STAR.P.2” tokens and issuing them as “STARP”.

The RTA can issue securities and also cancel them. The RTA issues preferred securities and once the investors request to convert them into common, the RTA will burn the preferred and issue the common for each investor.

The RTA will create for each investor a StartEngine LDGR blockchain address representing their securities.

A RTA designated by the Issuer should offer the following service:

- Be a U.S. based corporation
- Be a SEC Registered Transfer Agent
- Offer a secure application programming interface (“API”)
- Have the ability to lock the securities records during a transaction
- Use the email address as the unique identifier for each investor
- Allow recovery of ownership in case the email address is not longer accessible using a sworn affidavit signed by a Notary
- Have agreements with broker-dealers and ATS

An ATS should offer the following service:

- Be a registered Broker-Dealer with the SEC and a member of FINRA
- Be an Alternative Trading System
- Verify the identity of the investor against a set of federal databases (AML/KYC)
- Verify with the transfer agent that the securities are free and clear to sell and bear no limitations on transfer
- Verify any restrictions in the securities agreement signed with the issuer
- Offer a secure application programming interface (“API”) with the RTA
- Follow the SEC and State blue sky laws for transfer eligibility

The currency used by issuers to use the StartEngine LDGR application is ETH. The Issuer pays a yearly fee and provides an Ethereum address and the selected RTA to create the *StartEngine Issuer Name*. StartEngine pays the miners who host the blockchain ledgers in gas to execute the *StartEngine Issuer Name* smart contract. This Ethereum address can be used by the issuer to change the RTA.

LDGR Issuing Workflow

StartEngine, the RTA and the Issuer pay gas to the third party mining companies to hold the digital securities representation in the blockchain by creating a block in the blockchain for each step in the process.

The RTA provides the gas which is used to execute the smart contract to create each offering.

Every time the issuer makes a disbursement for an offering, the RTA who is now the Trusted Source for the StartEngine Issuer Name, call the smart contract to issue the securities.

Example: Best Company Ever Inc. Issuer picks Best Transfer Agent Inc. to be the RTA and conducts an offering of 100,000 shares in a Regulation Crowdfunding offering to 500 investors. The Issuer requests StartEngine LDGR to create a new code BCEV and pays in ETH. The RTA uses StartEngine LDGR to create individual investor blockchain addresses each containing the number of issued securities totalling 100,000 shares. The offering is named BCEV.C.1 until the trading restrictions are lifted and investors can use an ATS to sell BCEV shares. The ATS will execute the trade by calling a trading function. This function will call the RTA to check if the shares are available and, if true/available, the function will notify the Transfer Agent to register the new owner for the number of securities purchased.

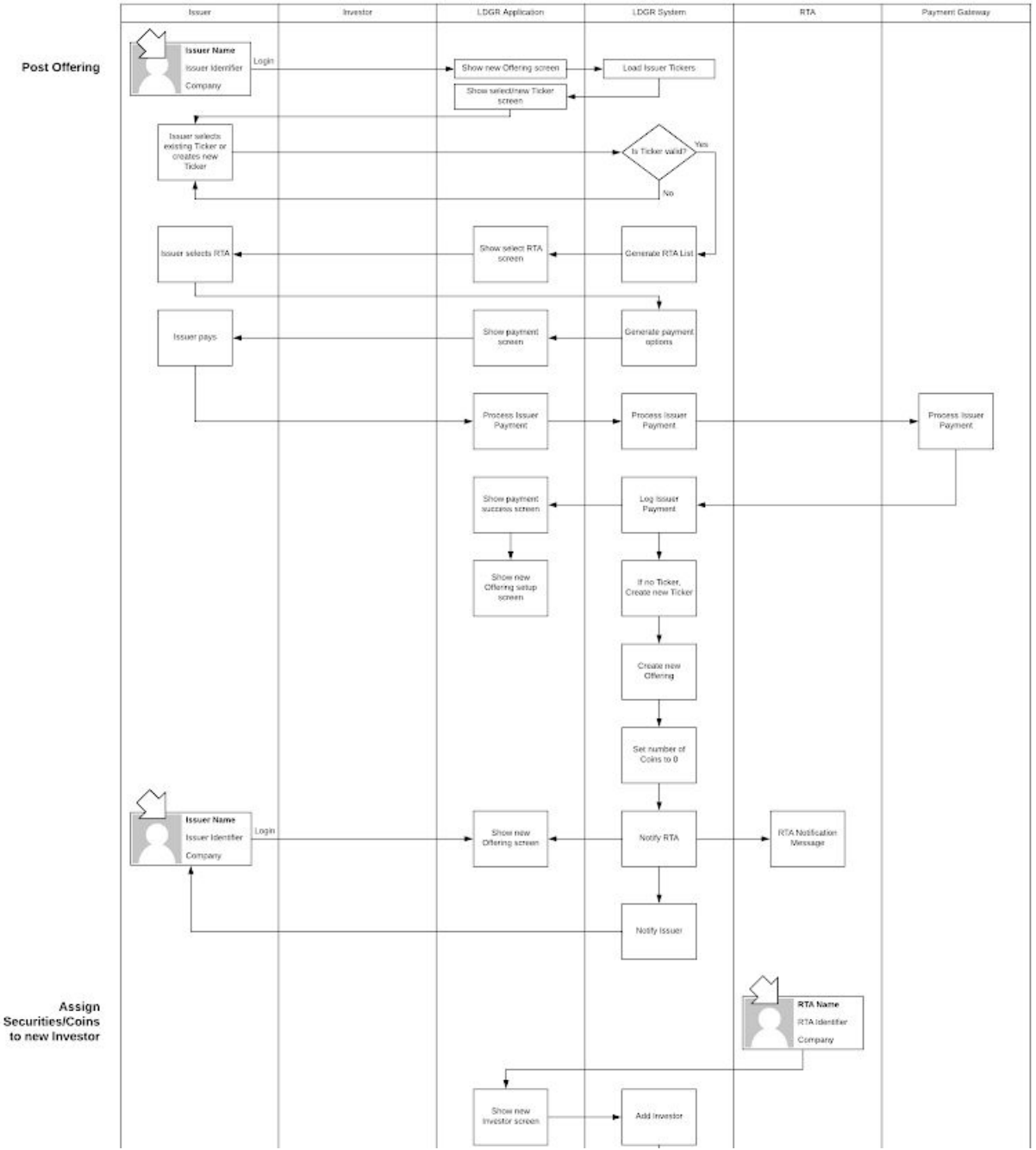
The following diagram illustrates the basic LDGR workflow for issuing securities:

LDGR Technical Diagrams

Create Offering

StartEngine

Technical Specification



LDGR Trading Workflow

To trade, first, the Broker-Dealer checks if the transaction is permitted per SEC and State blue sky laws, verifies the standard purchase agreement terms, gets the seller from the RTA API call using the blockchain address as the key and gets the buyer from the Trading Platform which onboarded the buyer. Then the RTA needs to approve this transfer and creates the new investor and add the investor to the list of investors. The RTA invokes LDGRToken transferFrom function to trade.

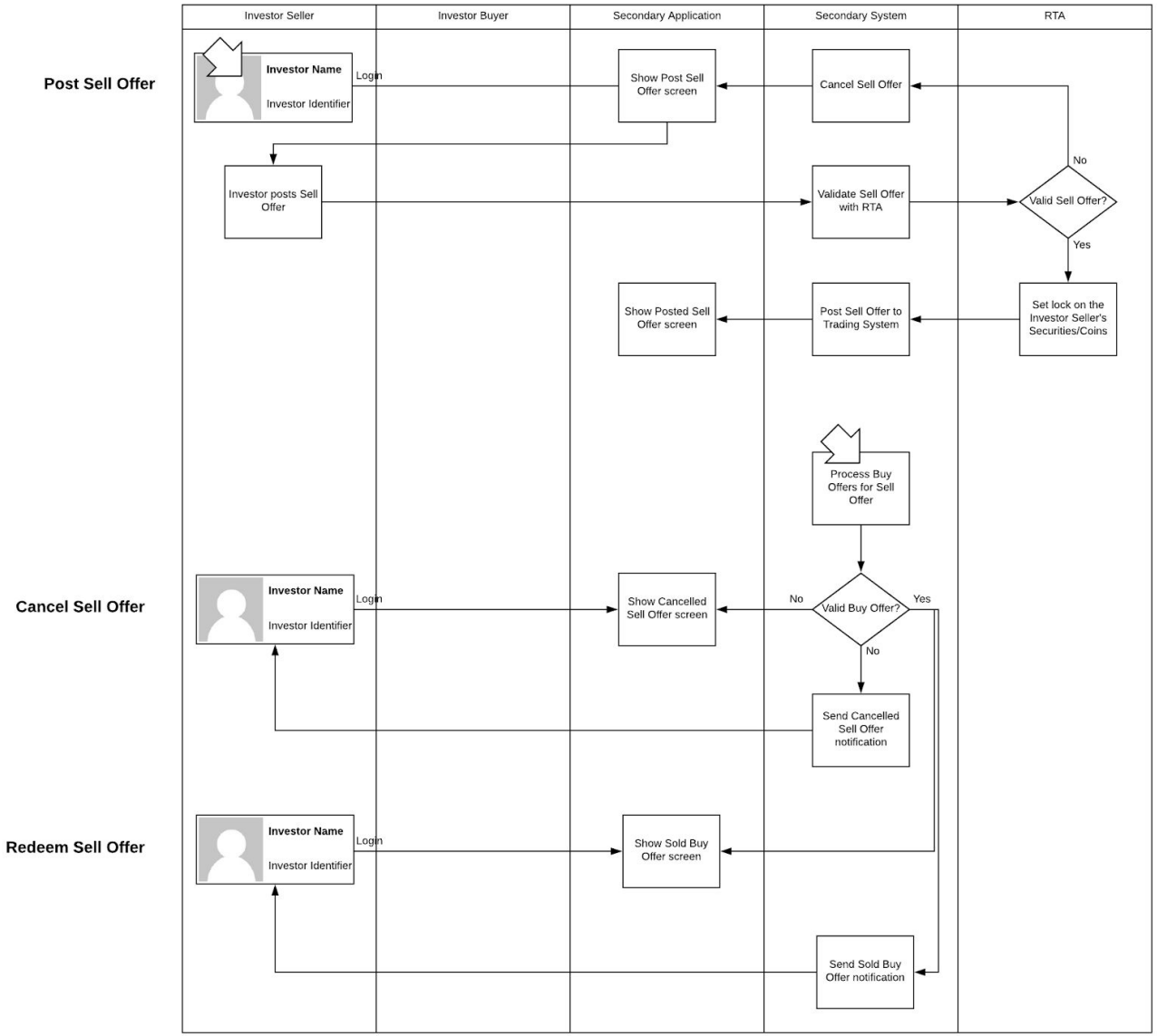
The following diagrams illustrate the basic LDGR workflow for buying and selling securities:

LDGR Technical Diagrams

StartEngine

Sell Offer

Technical Specification

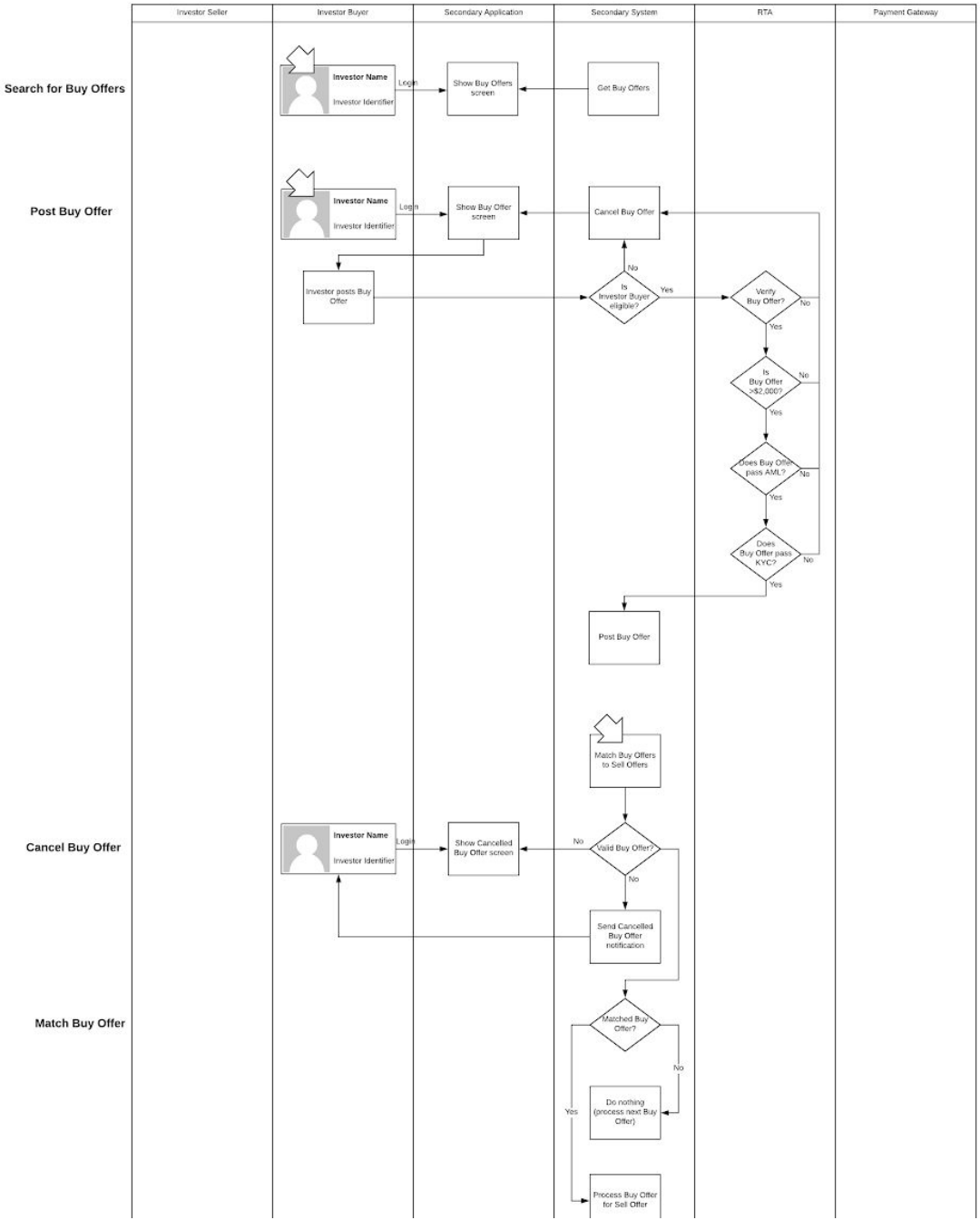


LDGR Technical Diagrams

Buy Offer

StartEngine

Technical Specification



LDGRToken Smart Contract

`LDGRToken` is the underlying technology that facilitates LDGR.

`LDGRToken` facilitates the recording of ownership and transfer of securities sold under the new Securities Act Regulations: Regulation Crowdfunding, Regulation D, and Regulation A. The issuance and trading of securities is subject to the Securities Exchange Commission (SEC) and specific U.S. state blue sky laws and regulations. `LDGRToken` enables issuers to sell tokenized securities to investors through a Funding Portal using one of the SEC registration exemptions. `LDGRToken` also enables Alternative Trading System (ATS) platforms to trade tokenized securities using one of the SEC registration exemptions. `LDGRToken` manages securities ownership during issuance and trading using these required and regulated roles - Issuer, Funding Portal, Broker Dealer, Registered Transfer Agent (RTA), and Investor:

- The Issuer is the company that wants to issue securities to investors. The Issuer specifies the Broker Dealer to manage the offering and the RTA to manage the investor records. The Issuer establishes trust relationships between all parties.
- The Funding Portal manages the Regulation Crowdfunding offerings (or issuances). The Broker Dealer manages the Regulation A or Regulation D offerings and operates the ATS for all three Regulations. The Funding Portal and Broker Dealer verify the identity of the Issuer and its Investors to satisfy Know Your Customer (KYC) requirements. The Funding Portal and the Broker Dealer also verify the source funds to satisfy Anti Money Laundering (AML) requirements (SEC). The Broker Dealer trusts the RTA to record and track security issuances and trades.
- The RTA manages the ownership of these securities by recording investor purchases of new issuances of securities (through a Funding Portal or Broker Dealer). The RTA also manages investor ownership transfers for trades (buys and sells) of securities (through an ATS) by verifying the seller's ownership of the securities and identify of the buyer. The RTA trusts the Broker Dealer to verify investors' identities (KYC) and their fund sources (AML) and comply with the securities laws.
- The Investor purchases securities at their initial issuance directly from the Issuer through the Funding Portal or Broker Dealer. The Investor also sells and buys securities to and from other Investors through the Broker Dealer's ATS.

The chain of ownership and trust for security issuance and transferring is as follows:

- The Issuer specifies who the Funding Portal, Broker Dealer, and the RTA are. The Issuer can change who the Funding Portal, Broker Dealer, and the RTA are.
- The Funding Portal and the Broker Dealer create Issuances. The Broker Dealer instructs the RTA to issue securities (through a Funding Portal or Broker Dealer) and transfer securities (through an ATS).

- The RTA records securities ownership on behalf of the issuer and transfers securities between Investors on behalf of the Funding Portal or Broker Dealer.

The RTA is the only role that is allowed to execute `LDGRToken`'s smart contract `mint`, `burnFrom`, and `transferFrom` functions (i.e. create a new `LDGRToken` for an Issuer to assign to an Investor and transfer `LDGRTokens` from one Investor to another Investor). No role is allowed to execute LDGRToken's smart contract `transfer` function. The Issuer, Funding Portal, and Broker Dealer are not permitted to manipulate `LDGRToken` (aside from the Issuer changing the RTA).

LDGR Specification

The core components of `LDGR` are the smart contracts `LDGRIssuer` (representing the Issuer), `LDGRSecurity` (representing the Security), and `LDGRToken` (representing ownership of the Security). `LDGRToken` extends Ethereum's ERC-20.

LDGRIssuer Smart Contract

`LDGRIssuer` is a non-ERC20 smart contract owned by StartEngine. This contract creates new ticker symbol names. It contains the list of ticker symbols which are unique eight letter Issuer names. This information goes into the blockchain and is visible to the public. The Issuer pays in ETH for registering a new issuer name (this amount can change).

IssuerName Data Table
StartEngine Blockchain Address
List of All Issuer Ticker Symbols

Issuing and transferring securities require that only the Issuer's owner and RTA execute specific functions to maintain regulatory compliance. Instantiating the `LDGRIssuer` contract requires the `Owned` and `TransferAgentControlled` contracts as modifiers.

For compliance reasons, the `LDGRIssuer` constructor must specify the owner, RTA, Issuer name, Issuer state of incorporation, Issuer state filing number, and the physical address of the Issuer's operations.

`LDGRIssuer` defines the `setTransferAgent` function (to change the RTA) and `setPhysicalAddressOfOperation` function (to change the Issuer's address) and must restrict execution to the Issuer's owner with the `onlyOwner` modifier. `setTransferAgent` must emit the `TransferAgentUpdated` event. `setPhysicalAddressOfOperation` must emit the `PhysicalAddressOfOperationUpdated` event. Additionally, `LDGRIssuer` defines the `createSecurity` function (to create securities) which must restrict execution to the RTA with the `onlyTransferAgent` modifier. `createSecurity` must emit the `CreateSecurity` event.

`LDGRIssuer` also defines the `getSecurities` function which is public and enables anyone to get the list of securities owned by the Issuer.

A contract compliantly implements `LDGRIssuer` if the contract is `IssuerControlled` and has a RTA. `LDGRIssuer` must implement the function `createSecurity`. The successful execution of `createSecurity` creates a new `LDGRSecurity` contract and triggers the `CreateSecurity` indexed event.

Owned

The Issuer must own its securities. `LDGRIssuer` must have an `Owned` contract with the `owner` specified in its constructor. `Owned` instantiates the `onlyOwner` modifier for `LDGRIssuer` to enable specific functions to permit only the Issuer's `owner` address to execute them. `Owned` also defines the function `transferOwnership` which transfer ownership of the Issuer to new `owner`'s address and can only be called by the `owner`. `transferOwnership` triggers the `OwnershipTransferred` event.

TransferAgentControlled

The Issuer must also have a RTA to enforce the role and responsibility of the RTA in managing securities transactions. `LDGRIssuer` must have the `TransferAgentControlled` contract with the `transferAgent` specified in its constructor. `TransferAgentControlled` instantiates the `onlyTransferAgent` modifier for LDGRIssuer to enable specific functions to permit only the Issuer's `transferAgent` address to execute them. `TransferAgentControlled` also defines the function `isTransferAgent` to lookup and identify the Issuer's RTA.

LDGRSecurity Smart Contract

`LDGRSecurity` is a smart contract that creates new offerings. The Issuer pays for each offering in ETH. This information goes into the blockchain, visible to the public. It lists the Issuer's blockchain address, unique ticker symbol, the corporate legal company, state file number, state of incorporation, operations business address (address, state and zip code), Registered Transfer Agent blockchain address and list of offerings. This information goes into the blockchain and is visible to the public.

IssueOfferings Data Table
Issuer Blockchain Address
Unique Ticker Symbol
Corporate Legal Name
State File Number
State of Incorporation
Physical Address of Operation
Transfer Agent Blockchain Address
List of Offerings

`LDGRSecurity` requires that the only the Issuer can create a security that only the RTA manages. `LDGRSecurity` also requires that only the Issuer's RTA can create `LDGRTokens` within a security. Instantiating the `LDGRSecurity` contract requires the `IssuerControlled` contract as a modifier. The `LDGRSecurity` constructor must specify the Issuer, security name, and security symbol. `LDGRSecurity` tracks multiple issuances of a security in the `tokens` field. The `LDGRSecurity` constructor can only be executed by the RTA and also creates the first issuance of a `LDGRToken`, setting the `LDGRToken`'s `issuance` to 0, and adds the `LDGRToken` to `tokens`.

`LDGRSecurity` defines the `createToken` function which must restrict execution to the Issuer's RTA with the `onlyIssuerTransferAgent` modifier. `createToken` must emit the `CreateToken` event. `LDGRSecurity` also defines the `getAllTokens` function which is public and enables anyone to get the list of tokens for the specific security.

`LDGRSecurity` upon creation must also deploy a compliant `LDGRToken` contract and save this to `token`.

IssuerControlled

The Issuer must control its securities. isIssuerControlled maintains the Issuer's ownership of their securities by owning the contract and enables the Issuer to set and update the RTA for the Issuer's securities. `LDGRSecurity` must have an `IssuerControlled` contract with the `issuer` specified in its constructor. `IssuerControlled` instantiates the `onlyIssuerTransferAgent` modifier for `LDGSecurity` to enable specific functions to permit only the Issuer's RTA to execute them.

LDGRToken Smart Contract

`LDGRToken` is an ERC20 smarty contract that contains the Issuer blockchain address, the unique ticker symbol, the unique sequential offering number, total number of securities, and list of investor blockchain addresses each with the number of securities. It also contains the total of list of initiated trades, whether completed or not. This information goes into the blockchain and is visible to the public.

Offering Data Table
Issuer Blockchain Address
Unique Ticker Symbol
Unique Sequential Offering Number
Total Number of Securities

Investor Blockchain Address with Number of Securities
Registered Transfer Agent

`LDGRToken` requires that only the Issuer can create a security's token that only the RTA manages. Instantiating the `LDGRToken` requires the `IssuerControlled` contract as a modifier. The `LDGRToken` constructor must specify the Issuer, security name, security symbol, security issuance number, security balances, security totalSupply, and security decimals (always 0). The `LDGRToken` constructor can only be executed by the Issuer.

`LDGRToken` defines the `balanceOf`, `transferFrom`, `mint`, and `burnFrom` functions. `balanceOf` is public and can be executed by anyone. `transferFrom`, `mint`, and `burnFrom` may only be executed by the RTA and are restricted with the `onlyIssuerTransferAgent` modifier.

`LDGRToken` extends the general Ownable and `onlyOwner` modifier to describe a specific subset of owners that automate and decentralize compliance. For a token to be compatible with `LDGRToken`, the token must have been created by a contract that implements `LDGRSecurity`, which in turn must have been created by a contract that implements `LDGRIssuer`. A token is `LDGRToken` compliant if the token implements `transferFrom`, `mint`, and `burnFrom` with modifiers allowing only the Issuer's RTA to execute them.

ERC-20 Extension

ERC-20 tokens provide the following functionality:

```
contract ERC20 {
 function totalSupply() public view returns (uint256);
 function balanceOf(address who) public view returns (uint256);
 function transfer(address to, uint256 value) public returns (bool);
 function allowance(address owner, address spender) public view returns (uint256);
 function transferFrom(address from, address to, uint256 value) public returns (bool);
 function approve(address spender, uint256 value) public returns (bool);
```

```
 event Approval(address indexed owner, address indexed spender, uint256 value);
 event Transfer(address indexed from, address indexed to, uint256 value);
}
```

ERC-20 is extended as follows:

```
/**
 * LDGRToken is an ERC-20 compatible token that complies with the new Securities Act
 * Regulations: Regulation Crowdfunding, Regulation D, and Regulation A.
 *
 * Implementations of the LDGRToken standard must specify the following constructor
 * arguments:
 *
 * '_issuer' the address of the issuer
 * '_name' - the name of the security
 * '_symbol' - the symbol of the security
 * '_issuanceNumber' - the issuance number of the security
 *
 * Implementations of the LDGRToken standard must implement the following modifier:
 *
 * 'onlyIssuerTransferAgent' - Only the address of the issuer's Registered Transfer
 * Agent is permitted to execute the functions transferFrom, mint, and burnFrom.
 *
 * Implementations of the LDGRToken standard must define the following optional ERC-20
 * fields:
 *
 * 'name' - The name of the security
 * 'symbol' - The symbol of the security
 *
 * Implementations of the LDGRToken standard must define the following fields:
 *
```

```
* ‘issuer’ - The address of the issuer of the security
* ‘issuanceNumber’ - The issuance number of the security
* ‘balances’ - The list of security owner addresses with balances
*
* Implementations of the LDGRToken standard must implement the following required ERC-20
* event to always fail:
*
* ‘Approval’ - Should never be called as the functions that emit this event must be
* implemented to always fail.
*
* Implementations of the LDGRToken standard must implement the following required
* ERC-20 functions to always fail:
*
* ‘transfer’ - Not a legal, regulated call for transferring securities because
* the token holder initiates the token transfer. The function must be implemented to
* always fail.
* ‘approve’ - Not a legal, regulated call for transferring securities because
* the token holder may not allow third parties to initiate token transfers. The
* function must be implemented to always fail.
*
* Implementations of the LDGRToken standard must implement the following optional
* ERC-20 function:
* ‘decimals’ - Must return ‘0’ because securities are indivisible entities.
*
* Implementations of the LDGRToken standard must implement the following functions:
*
* ‘mint’ - Only the address of the issuer’s Registered Transfer Agent may create new
* securities.
* ‘burnFrom’ - Only the address of the issuer’s Registered Transfer Agent may burn or
* destroy securities.
*
```

```
 * Implementations of the LDGRToken standard must implement the following safe
mathematical
 * functions:
 *
 * ‘mul’ - Multiplies two numbers, throws an overflow.
 * ‘div’ - Integer division of two numbers, truncating the quotient.
 * ‘sub’ - Subtracts two numbers, throws an overflow (i.e. if subtrahend is greater than
 * minuend).
 * ‘add’ - Adds two numbers, throws on overflow.
 */

Contract LDGRToken {

 /**
 * The constructor must implement a modifier that creates the modifier to only allow
 * the address of the issuer’s Registered Transfer Agent to execute the functions
 * transferFrom, mint, and burnFrom).
 */
 constructor(address _issuer, string _name, string _symbol, uint256 _issuanceNumber)
 public;

 /**
 * Transfer securities.
 *
 * ‘transferFrom’ must implement the onlyIssuerTransferAgent modifier to only allow the
 * address of the issuer’s Registered Transfer Agent to transfer LDGRTokens.
 * ‘transferFrom’ requires the _from address to have _value tokens.
 * ‘transferFrom’ requires that the _to address must not be 0 because securities must
 * not destroyed in this manner.
 */
 function transferFrom(address _from, address _to, uint256 _value) public returns (bool);
```

```
    /**
    * Create new securities.
    *
    * 'mint' must implement the onlyIssuerTransferAgent modifier to only allow the address of
    * the issuer's Registered Transfer Agent to mint LDGRTokens.
    * 'mint' requires that the _to address must not be 0 because securities must
    * not destroyed in this manner.
    * 'mint' must add _value tokens to the _to address and increase the totalSupply by
    * _value.
    * 'mint' must emit the Transfer event.
    */
    function mint(address _to, uint256 _value) public returns (bool);

    /**
    * Burn or destroy securities.
    *
    * 'burnFrom' must implement the onlyIssuerTransferAgent modifier to only allow the
    * address of the issuer's Registered Transfer Agent to burn LDGRTokens.
    * 'burnFrom' requires the _from address to have _value tokens.
    * 'burnFrom' must subtract _value tokens from the _from address and decrease the
    * totalSupply by _value.
    * 'burnFrom' must emit the Transfer event.
    */
    function burnFrom(address _who, uint256 _value) public returns (bool);
}
```

Backwards Compatibility

LDGRToken maintains compatibility with ERC-20 tokens with the following stipulations:

- `allowance(address tokenOwner, address spender) public constant returns (uint remaining);`

    - Must be implemented to always fail because `allowance` is not a legal, regulated call for a security.
- `transfer(address to, uint tokens) public returns (bool success);`
    - As the token holder initiates the transfer, must be implemented to always fail because `transfer` is not a legal, regulated call for a security.
- `approve(address spender, uint tokens) public returns (bool success);`
    - Must be implemented to always fail because `approve` is not a legal, regulated call for a security
- `transferFrom(address from, address to, uint tokens) public returns (bool success);`
    - Must be implemented so that only the Issuer's RTA can perform this action
- `Approval(address indexed tokenOwner, address indexed spender, uint tokens);`
    - Does not have to be implemented. `Approval` should never be called as the functions that emit this event must be implemented to always fail

Securities Exchange Commission Requirements

The SEC has very strict requirements as to the specific roles that are allowed to perform specific actions. Specifically, only the RTA may `mint` and `transferFrom` securities.

Implementers must maintain off-chain services and databases that record and track the Investor's name, physical address, Ethereum address, and security ownership amount. The implementers and the SEC must be able to access the Investor's private information on an as needed basis. Issuers and the RTA must be able to produce a current list of all Investors, including the names, addresses, and security ownership levels for every security at any given moment. Issuers and the RTA must be able to re-issue securities to Investors for a variety of regulated reasons.

Private Investor information must never be publicly exposed on a public blockchain.

Managing Investor Information

Special care and attention must be taken to ensure that the personally identifiable information of Investors is never exposed or revealed to the public.

Issuers who lost access to their address or private keys

There is no recourse if the Issuer loses access to their address to an existing instance of their securities. Special care and efforts must be made by the Issuer to secure and safely store their address and associated private key. The Issuer can reassign ownership to another Issuer but not in the case where the Issuer loses their private key.

If the Issuer loses access, the Issuer's securities must be rebuilt using off-chain services. The Issuer must create (and secure) a new address. The RTA can read the existing Issuer securities with the `getAllTokens` function, and the RTA can `mint` Investor securities accordingly.

Registered Transfer Agents who lost access to their address or private keys

If the RTA loses access, the RTA can create a new Ethereum address, and the Issuer can execute the `setTransferAgent` function to reassign the RTA.

Handling Investors (security owners) who lost access to their addresses or private keys

Investors may "lose" their credentials for a number of reasons: they simply "lost" their credentials, they were hacked or the victim of fraud, they committed securities-related fraud, or a life event (like death) occurred. Because the RTA manages the Issuer's securities, the RTA may authorize ownership related changes of securities (as long as they are properly notarized and verified).

If an Investor (or, say, the Investor's heir) loses their credentials, the Investor must go through a notarized process to notify the RTA of the situation and supply a new Investor address. From there, the RTA can `mint` the "lost" securities to the new Investor address and `burnFrom` the old Investor address (because the RTA knows all Investors' addresses).

Rationale

The are currently no token standards that conform to SEC regulations. The closest token is ERC-884 (Delaware General Corporations Law (DGCL) compatible share token) which states that SEC requirements are out of scope. EIP-1404 (Simple Restricted Token Standard) does not go far enough to address SEC requirements around re-issuing securities to Investors. EIP-1400 (Security Token Standard) has an interesting approach.

Test Cases

Test cases are available at https://github.com/StartEngine/ldgr_smart_contracts/tree/master/test.

Reference Implementation

A reference implementation is available at https://github.com/StartEngine/ldgr_smart_contracts.

Appendix A: LDGR Smart Contract Source Code

LDGRIssuer Smart Contract Source Code

```
pragma solidity ^0.4.24;

import "../libs/TransferAgentControlled.sol";
import "../libs/Owned.sol";
import "./LDGRSecurity.sol";

contract LDGRIssuer is Owned, TransferAgentControlled {
 string public name;
 string public stateFileNumber;
 string public stateOfIncorporation;
 string public physicalAddressOfOperation;
 address[] securities;

 event CreateSecurity(
  address indexed newSecurity,
  string name,
  string symbol
 );

 event TransferAgentUpdated(
  address indexed previousTransferAgent,
  address indexed newTransferAgent
 );

 event PhysicalAddressOfOperationUpdated(
```

```
  string previousPhysicalAddressOfOperation,
  string newPhysicalAddressOfOperation
 );

 constructor (
  address _initialOwner,
  address _initialTransferAgent,
  string _name,
  string _stateFileNumber,
  string _stateOfIncorporation,
  string _physicalAddressOfOperation
 ) Owned(_initialOwner) TransferAgentControlled(_initialTransferAgent) public {
  name = _name;
  stateFileNumber = _stateFileNumber;
  stateOfIncorporation = _stateOfIncorporation;
  physicalAddressOfOperation = _physicalAddressOfOperation;
 }

 function setTransferAgent(address _newTransferAgent) public onlyOwner {
  _setTransferAgent(_newTransferAgent);
 }

 function setPhysicalAddressOfOperation(string _newPhysicalAddressOfOperation) public onlyOwner
{
  _setPhysicalAddressOfOperation(_newPhysicalAddressOfOperation);
 }

 function _setPhysicalAddressOfOperation(string _newPhysicalAddressOfOperation) internal {
  emit PhysicalAddressOfOperationUpdated(physicalAddressOfOperation,
_newPhysicalAddressOfOperation);
  physicalAddressOfOperation = _newPhysicalAddressOfOperation;
```

```
 }

 function _setTransferAgent(address _newTransferAgent) internal {
  require(_newTransferAgent != address(0), "Address cannot be 0.");
  emit TransferAgentUpdated(transferAgent, _newTransferAgent);
  transferAgent = _newTransferAgent;
 }

 function getSecurities() public view returns (address[]) {
  return securities;
 }

 function createSecurity(string _name, string _symbol) public onlyTransferAgent returns
(address) {
  return _createSecurity(_name, _symbol);
 }

 function _createSecurity(string _name, string _symbol) internal returns (address) {
  address newSecurity = new LDGRSecurity(this, _name, _symbol);
  securities.push(newSecurity);
  emit CreateSecurity(newSecurity, _name, _symbol);
  return newSecurity;
 }
}
```

LDGRSecurity Smart Contract Source Code

```
pragma solidity ^0.4.24;

import "../libs/IssuerControlled.sol";
import "./LDGRToken.sol";

contract LDGRSecurity is IssuerControlled {
 string public name;
 string public symbol;
 address public token; // main token 0
 address[] tokens; // all tokens

 event CreateToken(
  address indexed newToken,
  uint256 indexed issuanceNumber
 );

 constructor (
  address _issuer,
  string _name,
  string _symbol
 ) IssuerControlled(_issuer) public {
  name = _name;
  symbol = _symbol;
  token = new LDGRToken(_issuer, _name, _symbol, 0);
  tokens.push(token);
  emit CreateToken(token, 0);
 }
```

```
 function getAllTokens() public view returns (address[]) {
  return tokens;
 }

 function createToken(uint256 _issuanceNumber) public onlyIssuerTransferAgent returns (address) {
  return _createToken(_issuanceNumber);
 }

 function _createToken(uint256 _issuanceNumber) internal returns (address) {
  LDGRToken newToken = new LDGRToken(issuer, name, symbol, _issuanceNumber);
  tokens.push(newToken);
  emit CreateToken(newToken, _issuanceNumber);
  return newToken;
 }
}
```

LDGRToken Smart Contract Source Code

```
pragma solidity ^0.4.24;

/*
Interface for LDGR
*/

import "../libs/IssuerControlled.sol";
import "../libs/openzeppelin/math/SafeMath.sol";

contract LDGRToken is IssuerControlled {
 using SafeMath for uint256;

 string public name;
 string public symbol;
 uint256 public issuanceNumber;
 mapping(address => uint256) balances;
 uint256 public totalSupply;
 uint8 public decimals;

 event Transfer(
  address indexed from,
  address indexed to,
  uint256 value
 );

 event Mint(
  address indexed to,
  uint256 value
```

```
);

event Burn(
 address indexed who,
 uint256 value
);

constructor(
 address _issuer,
 string _name,
 string _symbol,
 uint256 _issuanceNumber
) IssuerControlled(_issuer) public {
 name = _name;
 symbol = _symbol;
 issuanceNumber = _issuanceNumber;
 decimals = 0;
}

function balanceOf(address _investor) public view returns (uint256) {
 return balances[_investor];
}

function transferFrom(address _from, address _to, uint256 _value) public onlyIssuerTransferAgent returns (bool) {
 require(_value <= balances[_from], "Not enough balance.");
 require(_to != address(0), "_to is not valid.");
 balances[_from] = balances[_from].sub(_value);
 balances[_to] = balances[_to].add(_value);
 emit Transfer(_from, _to, _value);
 return true;
}
```

```
 function mint(address _to, uint256 _value) public onlyIssuerTransferAgent returns (bool) {
  require(_to != address(0), "_to is not valid.");
  balances[_to] = balances[_to].add(_value);
  totalSupply = totalSupply.add(_value);
  //emit Mint(_to, _value);
  emit Transfer(address(0), _to, _value);
  return true;
 }

 function burnFrom(address _who, uint256 _value) public onlyIssuerTransferAgent returns (bool) {
  require(_value <= balances[_who], "_value cannot be greater than balance.");
  balances[_who] = balances[_who].sub(_value);
  totalSupply = totalSupply.sub(_value);
  //emit Burn(_who, _value);
  emit Transfer(_who, address(0), _value);
  return true;
 }
}
```