Enjin Coin Yellow Paper

Q4 2017, Minimum Viable Product (Oct 1 - Dec 31)

Version 1.0 *- Updated November 2, 2017*

This yellow paper outlines Enjin Coin's initial milestones in 2017 and a **technical summary** of our platform design. In this quarter we are creating a Minecraft Plugin, Smart Wallet and SDK that will constitute our MVP with working gameplay.

Java and PHP SDK

We'll begin development of the SDK in two separate languages. PHP will be used for the Trusted Platform, and the Java SDK for the game server.

Once the MVP is complete, both SDKs will continue development so each language ends up with the complete set of SDK features.

The final SDK will allow selection / configuration of the Transport Layers, Database, and Ethereum Node. For the MVP we will be focusing on a single configuration.

PHP SDK

- MySQL connection
    - Transaction Requests
    - Apps
    - Identities
    - Events
- Ethereum Connection
    - Using Geth or configurable node
    - Subscribe to Ethereum events and update DB
    - Send PubNub event messages to subscribed sdk nodes per-app
- Public API
    - Allow SDK app servers to connect and provide them with PubNub subscription details.

Java SDK

- Sonatype Central Maven Repository for Artifacts
- Communicate with Trusted Platform API via JSON-RPC
    - In the future this API may be on the same server, so allow internal calls, but it's configurable to an external HTTP/HTTPS API as well.
- Subscribe to PubNub messages from the API
- Provide events to the game engine
    - Identity Linked
    - Transaction Request Accepted by Identity

    - Transaction Request Cancelled by Identity
    - Identity received items
    - Identity received items (transaction confirmed)
    - Identity received items (transaction error)
    - Identity lost items
    - Identity lost items (transaction confirmed)
    - Identity lost items (transaction error)
    - Transaction Request Created
    - Identity Linking Code Generated
- Provide internal SDK functions/api for the game engine to use
    - List all available item types configured on Trusted Platform server
    - List the items/balances of a specific user / wallet
    - List the wallets owned by the Game server
    - Create a Transaction Request
    - List pending Transaction Requests
    - Delete Transaction Request
    - Create identity manually (?)
    - Get event list
        - Can be filtered by time/id to re-sync if the PubNub connection was lost or server was offline

Minecraft Plugin

- Online and offline server support
- Linking to player name/uuid
- Item Vault - A virtual inventory to store block-chain items.
- Token behavior files (JSON)
- Item Stacks
- Permissions
- Currency
- Suite of console commands

API Methods

The API Server provides the following business logic:

- Admin functions
    - Create Identity
    - Generate and return a temporary Identity linking code
        - This is unique across the entire platform, but associates with a app
    - List identities
    - Retrieve specific identity and its data
    - Delete an identity
    - Update an identity
    - Create transaction request
    - Add app wallet
    - Remove app wallet
- Authenticated functions - *(Admin also has access to these)*
    - List transaction requests
    - Retrieve specific transaction request
    - Delete transaction request
- Public functions
    - Authenticate Smart Wallet
    - Get custom token list for app
    - Retrieve custom token information
    - Search player name & return wallet hash + player info

SDK Methods

The SDK needs public functions available to the game code, to create **Transaction Requests** when an ingame transaction is required.

Create Transaction Request

- Player Identity
- Transaction Request Type
    - Buy
    - Sell
    - Send
    - Use
- Recipient
    - Allow specifying a wallet address directly
    - Or a Player Identity
- Icon (Optional)
- Title
    - "108 Silver Coins", or "Builder Rank, etc
- Value
    - Amount
    - Currency (Custom token or ENJ)

Cancel Transaction Request

Get Transaction Request

List Transaction Requests

Get Token Balance(s)

Get Identity

List Identities

Create Identity

Generate Identity Linking Code

List Events

Smart Contracts

ENJToken

- The ERC20 token for "ENJ".
- https://github.com/enjin/contracts/blob/master/solidity/contracts/ENJToken.sol
- Contract deployed at: 0xF629cBd94d3791C9250152BD8dfBDF380E2a3B9c

CustomTokens

- Main contract used to mint Custom Tokens on the Enjin Coin platform.
    - Holds ENJ in reserve for each Custom Token
    - Provides functions to Create, Mint, and Melt tokens.
    - Factory allows deployment of ERC20Adapter for each Custom Token type.
- https://github.com/enjin/smartcontracts/blob/master/contracts/CustomTokens.sol

ERC20Adapter

- Adapter to CustomTokens which provides standard ERC-20 functions and calls the appropriate functions in the CustomTokens contract.
- https://github.com/enjin/smartcontracts/blob/master/contracts/ERC20Adapter.sol

ERC20AdapterFactory

- Factory for deploying ERC20Adapter
- https://github.com/enjin/smartcontracts/blob/master/contracts/ERC20AdapterFactory.sol

PlatformRegistry

- Allows Trusted Platforms to register globally for linking wallets to their API
- https://github.com/enjin/smartcontracts/blob/master/contracts/PlatformRegistry.sol

ENJAllocation

- Helper contract that shows the current total ENJ in circulation.
- https://github.com/enjin/contracts/blob/master/solidity/contracts/ENJAllocation.sol
- Contract deployed at: 0xBaB738C3A9B6122fb002FfB0C747745CeDad3f09

Mobile Smart Wallet

The first Enjin Smart Wallet wallet build is being developed for Android OS then iOS.

Coin Support

- ENJ
- Custom Tokens
- ETH
- ERC-20 tokens

User Interface Sections

- Initial Screen
- Create Wallet
- Backup Wallet
- Select Coins
- My Wallet
- Coin Details
- Receive
- Send
- Send (Advanced mode)
- Transaction Detail
- Items / Custom Tokens
- Transaction Requests Feed / Event Feed
- Transaction Request Details
- Link to Game Account

The Smart Wallet will contact a default Trusted Platform API upon first launch. After calling the API init function, it will subscribe to:

- Price ticker
- Transaction Request notifications

The game will generate an alphanumeric Identity Linking Code (1234ABCD) that the Smart Wallet will connect to a remote ethereum node and check the Platform Registry for the numeric digits of the code (Platform ID) when linking up an account. The following alphabetical code will be sent to the Trusted Platform to automatically link the player's game account to the wallet.

Item Types

Tradable Item

- These items can be equipped and traded normally between players.
- Example: Armor pieces, Weapons
- Simply create a new custom token with parameter:
    - transferable=2

Bound Item

- These items cannot be traded, but can be melted by the player.
- Example: Armor and weapons from raids, "Cursed Helmet", game puzzle-piece
- Simply create a new custom token with parameter:
    - transferable=0

Consumable Item

- Example: Magic potion, Health-Kit, Food, Chest key, Crafting item
- To consume the item, the game SDK can create a Transaction Request with type="use" and the recipient being the game developer's wallet.
- Once this Transaction Request is accepted, the SDK will verify the recipient and type are correct, and apply the benefits in-game.
- Other mechanisms for Consumables will be tested, such as using Smart Contract escrows, Approve, and Multi-Approve functions.

Bancor for Enjin Coin Tokens

Bancor is a on-chain exchange and liquidity platform built on Ethereum.

Enjin Coin (ENJ) is a standard ERC-20 token, with a static 1 Billion token supply.

We will be deploying a Token Relay (ENJBNT) with a 1-2% reserve of ENJ and BNT. This will allow a simple form of liquidity. We will hard-cap the maximum exchange fee to 0.5% in the token changer, but it will be initially set to 0.2%.

Reserve: 10-20M Enjin Coins + Equivalent value in BNT

Custom Tokens and Reserve Value

There will be 2 standard modes of backing custom coins with ENJ.

Fixed Reserve

A static exchange rate and maximum total supply of minted tokens will be defined. Users can always liquidate their held Custom Tokens back to the static number of ENJ held in reserve at the fixed exchange rate.

Bancor Smart Token

Tokens using the ERC20 Adapter will be Bancor Smart Tokens, allowing them to connect to the Bancor network.

A Bancor Token Converter can be launched for each smart token with an ENJ reserve using a configurable "weight". Once the tokens are minted, the total number of custom tokens may fluctuate and the price will dynamically adjust based on market sentiment and demand.

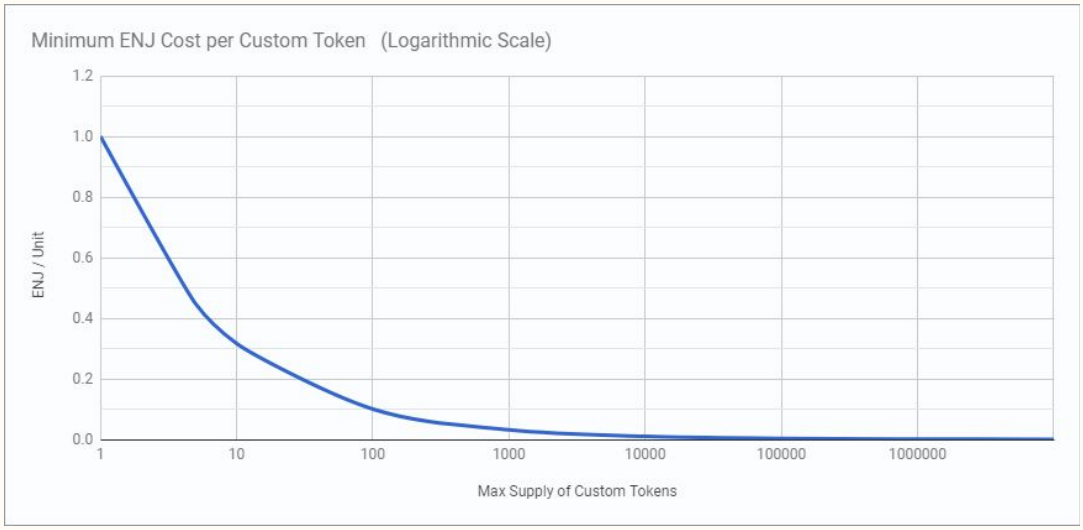
Minimum Reserve Price Calculation

It's necessary to enforce a bare-minimum ENJ price needed to mint custom items.

The way we are tackling this is to use an "economy of scale" formula. The initial supply of custom tokens needs to be minted all at once. The number of tokens created during this initial minting process defines the final minimum price per token. As the total tokens created increases, price per token decreases, by using the reciprocal of the square root:

$$y = \frac{\sqrt{x}}{x}$$

The following graph and minimum minting cost chart illustrates:



Max Items	Min. ENJ Per Item	Total Min. Cost
1	1	1
100	0.1	10
10000	0.01	100
1000000	0.001	1000
100000000	0.0001	10000

Minimum Reserve Price Adjustments

This section is still being researched.

As the cost of ENJ may change over time, the Minimum Reserve Price cannot become unaffordable to new game developers entering the platform. We have a few possible proposals to modulate the Minimum Reserve Price based on ENJ supply:

Based on Percentage of Enjin Coins in true circulation

- The number of ENJ tied-up in minted Custom Tokens could be used to decrease the final Minimum Reserve Price cost.

Manual Adjustment

- The Enjin Team may be allowed to define the amount (0.0%-100.0%) by which the standard Minimum Reserve price is reduced from its final value.

Price Oracle

- An Ethereum Oracle can be designed with proper moderation controls that will peg the approximate ENJ price at any given day or hour. This value can be used to adjust the minimum reserve price formula.

This area is still under debate and will be updated in the future.

Initial Gameplay and Blockchain behavior

Enjin Coin will tackle the most common forms of gameplay first, then expand the platform to cover more complex use-cases and technical optimizations.

Stage I

- The game will recognize and honor blockchain items that players possess
- Players can trade items with other players and the game server by approval of Transaction Requests in the wallet. They will be notified and must approve the transaction.
- Blockchain game items may be consumable but require approval to consume. Stacks may be approved (Example: Use up to 50 magic potions).
- Notifications about transactions sent from the game platform to smart wallets
- Notifications about p2p and p2s transactions sent from player wallets
- Transaction requests from Server to Players involved in the transaction

Transactions on the blockchain

Players may be receiving transactions from non-smart wallets, or might not have their smart wallet running while playing a game. To handle this, the game server and wallets will watch for relevant transactions for all connected identities.

As soon as transactions are seen, the user's existing item balance will appear to update. Any new items appear as ghost items (non-tradeable) until they are confirmed. This will be done seamlessly so users can still transfer any previous confirmed balances that they have, and use the unconfirmed items in the game to various degrees.

Transactions within the game

We will be investigating off-chain notifications, using both standard messaging transport layers and blockchain-specific protocols. µRaiden and Raiden will also be tested, but most of these implementations will be developed after Q4.