# Ravencoin Roadmap

## Phase 1 - Complete

Ravencoin (RVN) is a Proof of Work coin built on the Bitcoin UTXO model. As with other Bitcoin derivatives, RVN coins are distributed to persons augmenting the Raven network by mining Raven. * x1000 coin distribution (21 Billion Total) * 10x faster blocks (1 per minute) * In app CPU mining * ~1.4 Day difficulty adjustment (2016 blocks) * Addresses start with R... for regular addresses, or r... for multisig * Network Port: 8767 * RPC Port: 8766

## Phase 2 - Assets (in progress)

### ASIC Resistance

ASIC Resistance - A published commitment to continual attempts at ASIC resistance. If ASICs are created for x16r, then we will, at a specific block number, modify one of the algorithms to add Equihash, EthHash or similar efforts to increase the resistance to ASIC miners for Raven.

### Asset Support

Ravencoin will be a hard fork that extends Raven to include the ability to issue and transfer assets. The expected release of asset capabilities will be approximately seven months after the release of RVN. Raven will be extended to allow issuing, reissuing, and transfer of assets. Assets can be reissuable or limited to a set supply at the point of issuance. The cost to create assets will be 500 RVN to create any qty of an asset. Each asset name must be unique. Asset names will be limited to A-Z and 0-9, '' *and '.' and must be at least three characters long. The '.' and the* ' cannot be the first, or the last character, or be consecutive.

Examples of valid assets:
THE_GAME
A.TOKEN
123

Examples of invalid assets:
_TOKEN
THEEND.
A..B (consecutive punctuation)
AB
12
.FIRST

The RVN used to issue assets will be sent to a burn address, which will reduce the amount of RVN available.

Asset transfers require the standard RVN transaction fees for transfer from one address to another.

#### Metadata

Metadata about the token can be stored in IPFS. Initially this cannot be changed. If there is a demand, the system can be updated to allow updating the metadata by the token issuer.

#### Rewards

Reward capabilities will be added to allow payment (in RVN) to all holders of an asset. Payments of RVN would be distributed to all asset holders pro rata. This is useful for paying dividends, dividing payments, or rewarding a group of token holders.

Example: A small software company issues an asset GAMECO that represents a share of the project. GAMECO tokens can be traded with others. Once the software company profits, those profits can be distributed to all holders of GAMECO by sending the profits (via RVN) to all holders of GAMECO.

#### Block Size

Raven may increase the blocksize from 1 MB to X MB to allow for more on-chain transactions.

### Phase 3 - Rewards

Rewards allow payment in RVN to asset holders.

### Phase 4 - Unique Assets

Once created, assets can be made unique for a cost of 5 RVN. Only non-divisible assets can be made unique. This moves an asset to a UTXO and associates a unique identifier with the txid. From this point the asset can be moved from one address to another and can be traced back to its origin. Only the issuer of the original asset can make an asset unique.
The costs to make unique assets will be sent to a burn address.

Some examples of unique assets:
* Imagine that an art dealer issues the asset named ART. The dealer can then make unique ART assets by attaching a name or a serialized number to each piece of art. These unique tokens can be transferred to the new owner along with the artwork as a proof of authenticity. The tokens ART:MonaLisa and ART:VenusDeMilo are not fungible and represent distinct pieces of art. * A software developer can issue the asset with the name of their software ABCGAME, and then assign each ABCGAME token a unique id or license key. The game tokens could be transferred as the license transfers. Each token ABCGAME:398222 and ABCGAME: are unique tokens. * In game assets. A game ZYX_GAME could create unique limited edition in-game assets that are owned and used by the game player. Example: ZYX_GAME:Sword005 and ZYX_GAME:Purse * RVN based unique assets can be tied to real world assets. Create an asset named GOLDVAULT. Each gold coin or gold bar in a vault can be serialized and audited. Associated unique assets GOLDVAULT:444322 and GOLDVAULT:555994 can be created to represent the specific assets in the physical gold vault. The public nature of the chain allows for full transparency.

## Phase 5 - Messaging

Messaging to token holders by authorized senders will be layered on top of the Phase 3 unique assets. See KAAAWWW Protocol for additional information.

## Phase 6 - Voting

Voting will be accomplished by creating and distributing parallel tokens to token holders. These tokens can be sent to RVN addresses to record a vote.

## Appendix A - RPC commands for assets

`issue(to_address, asset_name, qty, units=1, reissuable=false)`
Issue an asset with unique name. Unit as 1 for whole units, or 0.00000001 for satoshi-like units. Qty should be whole number. Reissuable is true/false for whether additional units can be issued by the original issuer.

`issuefrom(from_address, to_address, qty, units, units=1, reissuable=false)`
Issue an asset with unique name from a specific address -- allows control of which address/private_key is used to issue the asset. Unit as 1 for whole units, or 0.00000001 for satoshi-like units. Qty should be whole number. Reissuable is true/false for whether additional units can be issued by the original issuer.

`issuemore(to_address, asset_name, qty)`
Issue more of a specific asset. This is only allowed by the original issuer of the asset and if the reissuable flag was set to true at the time of original issuance.

`makeuniqueasset(address, asset_name, unique_id)`
Creates a unique asset from a pool of assets with a specific name. Example: If the asset name is SOFTLICENSE, then this could make unique assets like SOFTLICENSE:38293 and SOFTLICENSE:48382 This would be called once per unique asset needed.

`listassets(assets=*, verbose=false, count=MAX, start=0)`
This lists assets that have already been created. It does not distinguish unique assets.

`listuniqueassets(asset)`
This lists the assets that have been made unique, and the address that owns the asset.

`sendasset(to_address, asset, amount)`
This sends assets from one asset holder to another.

`sendassetfrom(from_address, to_address, asset, amount)`
This sends asset from one asset holder to another, but allows specifying which address to send from, so that if a wallet that has multiple addresses holding a given asset, the send can disambiguate the address from which to send.

`getassettransaction(asset, txid)`

This returns details for a specific asset transaction.

`listassettransactions(asset, verbose=false, count=100, start=0)`
This returns a list of transactions for a given asset.

`reward(from_address, asset, amount, except=[])`
Sends RVN to holders of the the specified asset. The Raven is split pro-rata to holders of the asset. Any remainder that cannot be evenly divided to the satoshi (1/100,000,000 RVN) level will be added to the mining fee. except is a list of addresses to exclude from the distribution - used so that you could exclude treasury shares that do not participate in the reward.

`send_asset(from_address, from_asset, to_asset, amount, except=[])`
Sends an asset to holders of the the specified to_asset. This can be used to send a voting token to holders of an asset. Combined with a messaging protocol explaining the vote, it could act as a distributed voting system.