MasterCoin Complete Specification

vs. 1.0 (First Complete Specification)

"dacoinminster" (jr.willett@gmail.com)

Summary

We claim that *the existing bitcoin network can be used as a protocol layer, on top of which new currency layers with new rules can be built without changing the foundation*. We further claim that the new protocol layers described in this document:

- Will fix the two biggest barriers to widespread bitcoin adoption: instability and insecurity.
- Will financially benefit the entire bitcoin user community, including those who don't use the new protocol layers.
- Will provide initial funds to hire developers to build software which implements the new protocol layers, and ongoing funds to pay for maintenance of this software.
- Will richly reward early adopters of the new protocol, in proportion to how successful it is.

Assumptions

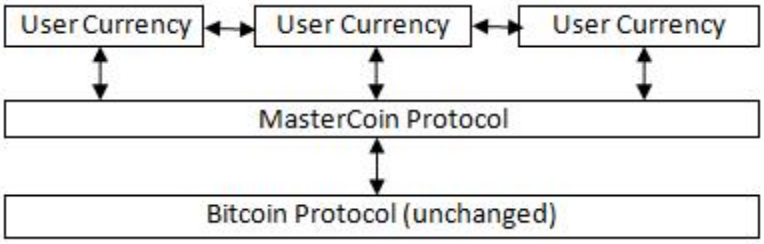
Our claims are built on the following assumptions:

- Alternate block chains compete with bitcoins financially, confuse our message to the world, and dilute our efforts. These barriers interfere with the adoption momentum of bitcoin and the adoption momentum of alternate currencies as well, regardless of how well-conceived their rules may be.
- New protocol layers on top of the bitcoin protocol will increase bitcoin values, consolidate our message to the world, and concentrate our efforts, while still allowing individuals and groups to issue new currencies with experimental new rules. The success of any experimental currency protocol layer will enhance the value and success of the foundational bitcoin protocol.
- Getting consensus and widespread adoption from the bitcoin community is not needed to add protocol layers, since no changes to the foundational bitcoin protocol are required.
- Tiny bitcoin transactions can be encoded into the block chain to support and represent transactions in higher protocol layers.

- A protocol can pay for its own software development, “bootstrapping” itself into existence, utilizing a trusted entity to hold funds and hire developers.
- It is possible to create tools to allow end users to create currency protocol layers which have a stable value, pegged to an external currency or commodity. In this way, users of these currencies can own stabilized virtual currency tied to U.S. Dollars, Euros, ounces of gold, barrels of oil, etc.
- It is possible for users of these new currencies to exchange between currencies with each other using simple rules and no central exchange.

Visualization

The proposed protocol layers can be visualized as follows, with arrows representing users exchanging between currencies:



Note that all transfers of value are still stored in the normal bitcoin block chain, but higher layers of the protocol assign additional meaning to some transactions.

Document History

- Version 0.5 released 1/6/2012
- Version 0.7 released 7/29/2013

Previous versions of this document can be found at https://sites.google.com/site/2ndbtcwpaper/

MasterCoin Design

The “MasterCoins” protocol layer between the existing bitcoin protocol and users’ currencies is intended to be a base upon which anyone can build their own currency. The software implementing MasterCoins will contain simple tools which will allow anyone to design and release their own currency with their own rules without doing any software development.

The “Exodus Address”

Perhaps you have heard of the “Genesis Block” which launched the bitcoin protocol. The MasterCoin protocol has a similar starting point in the block chain, called the “Exodus Address” - the bitcoin address from which the first MasterCoins will be sold. The Exodus Address is:
1EXoDusjGwvnjZUyKkxZ4UHEf77z6A5S4P (please read the “Special Considerations” section below before sending bitcoins to this address!)

Initial distribution of MasterCoins will essentially be a fundraiser to provide money to pay developers to write the software which fully implements the protocol. The distribution is very simple, and will proceed as follows:

1. Anyone sending bitcoins to the Exodus Address before August 31st, 2013 is recognized by the protocol as owning 100x that number of MasterCoins. For instance, if I send 100 bitcoins to the Exodus Address before August 31st, my bitcoin address owns 10,000 MasterCoins after August 31st.

2. Early buyers get additional MasterCoins. In order to encourage adoption momentum, buyers will get an additional 10% bonus MasterCoins if they make their purchase a week before the deadline, 20% extra if they purchase two weeks early, and so on, including partial weeks. Thus, if I send 100 bitcoins to the exodus address 1.5 weeks before August 31st, the protocol recognizes my bitcoin address as owning 11,500 MasterCoins (10000 + 15% bonus).

3. Attempts to send funds to the Exodus Address on or after September 1st 2013 (as determined by bitcoin block chain records) will not be considered MasterCoin purchases.

Note that anyone who purchases MasterCoins also receives the same number of “Test MasterCoins” which will be used for testing new features before they are available for use in the MasterCoin protocol.

Initially, the only valid MasterCoin transaction will be a “simple transfer” (defined later in this document), but the additional features described in this document will also become valid in the future (at specific pre-announced block milestones) once they are fully tested.

Reward MasterCoins

For every 10 MasterCoins sold, an additional “reward MasterCoin” will also be created, which will be awarded to the Exodus Address slowly over the following years. These delayed MasterCoins will ensure that we have plenty of motivation to increase the value of MasterCoins by completing the features desired by users. The reward will be structured so that we receive 50% of the reward by one year after the initial sale, 75% by a year later, 87.5% by a year later, and so on:

$$\text{Reward MasterCoins Percent Vested} = 1 - (0.5^{y})$$

(y = years since MasterCoins sale)

Hiding MasterCoin Protocol Data in the Block Chain

Bitcoin has some little-known advanced features (such as scripting) which many people imagine will enable it to perform fancy new tricks someday. MasterCoin uses exactly NONE of those advanced features, because support for them is not guaranteed in the future, and MasterCoin doesn't need them anyway.

MasterCoin transactions are defined as a series of bitcoin payments from a bitcoin address where the payments match this pattern:

- A “reference” payment (of any amount) to another bitcoin address (called the reference address), such as the person being paid MasterCoins
- One or more “data” payments (of any amount) to fake bitcoin addresses. (A fake bitcoin address can contain 20 bytes of arbitrary data, not including overhead such as the version number and check-sum of the address.) This is data used by the protocol, such as the number of MasterCoins being paid
- A “marker” payment (of any amount) to the Exodus Address marking this series of payments to be interpreted as a MasterCoin transaction.

These payments may be sent one at a time or all at once (via bitcoin's “sendmany” function), and in any order (even if we sent them in a particular order, there is no way to force miners to keep them in that order). The only requirement is they cannot have long time gaps between them (not more than 6 blocks apart in the block chain), and they must be distinguishable from other MasterCoin transactions by using a 6-block time gap or the “sendmany” function.

Putting Data Packets in Order

Since the packets are stored unordered, some consideration must be made to how to tell these payments apart and put data packets in their proper order.

For ordering data packets, each 20-byte payload starts with a 1-byte sequence number which is incremented for each data packet.

In order to distinguish data packets from the reference address, the data packet sequence numbers start at n+1 where n is the sequence number of the reference packet if it were treated as a data packet. Any additional data packets can continue to use up sequence numbers n+2, n+3, and so on until all sequence numbers are used except for n-1.

As an example of how this works, let's imagine a MasterCoin transaction that has two data packets. If the reference address happens to have a sequence number of 62, then the first data packet has a sequence number of 63 and the second has a sequence number of 64. Note that sequence number 255 is followed by 0.

The theoretical limit on the amount of protocol data that could be stored in one MasterCoin transaction is therefore 254 data payments * 19 bytes = 4826 bytes, but we expect that most common transactions should only need one or two data payments. All numbers are stored big-endian (most significant digits first).

Special Considerations to Avoid Invalid Transactions

Not every wallet lets you choose which address bitcoins come from when you make a payment, and MasterCoin transactions must all come from the address which holds the MasterCoins. If a wallet contains bitcoins stored in multiple addresses, the wallet must first consolidate the bitcoins by sending ALL of them to the address which is going to initiate a MasterCoin transaction. Then, any MasterCoin-related bitcoin transactions will be sent from that address.

Wallets which do not allow you to consolidate to one address and send from that address (such as online web wallet providers) will not work for MasterCoin unless they are modified to do so. For this reason, **attempting to purchase MasterCoins from an online web wallet will likely result in the permanent loss of those MasterCoins.**

All transactions should be for amounts above the “dust” threshold, and should include an appropriate fee to ensure that miners include them in the block chain in a timely manner.

If these recommendations are not followed, the MasterCoin transaction may be malformed and therefore invalid.

Transferring MasterCoins

Say you want to transfer 1 MasterCoin to another address. Only 16 bytes are needed, which fits into a single data payment. The data stored is:

1. Transaction type = 0 for simple transfer (32-bit unsigned integer, 4 bytes)
2. Currency identifier = 1 for MasterCoin (32-bit unsigned integer, 4 bytes)
3. Amount to transfer = 100,000,000 (1.00000000 MasterCoins) (64-bit unsigned integer, 8 bytes)

Note that the amount to transfer is multiplied by 100,000,000 before it is stored, which allows for MasterCoins to be sent with the same precision as bitcoins (eight decimal places). The reference payment (described earlier) determines the address receiving the MasterCoins.

Note that if the transfer comes from an address which has been marked as "Savings", there is a time window in which the transfer can be undone. Otherwise MasterCoin transactions are not reversible.

Marking an Address as "Savings"

Say you want to back up your savings wallet in the cloud, but if someone manages to hack into it, you want transactions out of that wallet to be reversible for up to 30 days. Doing this takes 8 bytes, which fits into a single data payment:

1. Transaction type = 10 for marking savings (32-bit unsigned integer, 4 bytes)
2. Reversibility period = 2,592,000 seconds (30 days) (32-bit unsigned integer, 4 bytes)

The maximum reversibility period is 365 days (1,892,160,000 seconds) to avoid accidents. Marking an address as savings is PERMANENT and cannot be undone. If an address is marked as savings, the reversibility rules affect not only MasterCoins, but any MasterCoin-derived child currency stored at that address.

When marking an address as savings, the reference payment should point to a "guardian" address authorized to reverse fraudulent transactions. The guardian address should preferably be from an unused offline or paper wallet.

When a fraudulent transaction is reversed, any pending funds go to the guardian address, rather than going back to the compromised savings address. Also, any funds which remain in the compromised address also go to the guardian wallet.

An address marked as savings can only do simple transfers (transaction type=0). All other transaction types require addresses without a reversibility time window.

Marking a Savings Address as Compromised

Say you notice that the address you marked as savings has been compromised, and you want to reverse transactions and transfer everything to the guardian address. Doing this takes 4 bytes, which fits into a single data payment

1. Transaction type = 11 for marking a compromised savings address (32-bit unsigned integer, 4 bytes)

This transaction must be sent from the guardian address. The reference payment must be to the compromised savings address. Funds from any pending transactions and any remaining funds will then be transferred to the guardian address, both MasterCoins and any currencies derived from MasterCoins.

Advantages of the Savings/Guardian Model

The savings/guardian model is intended to allow the user to take extreme precautions against accidental loss of the savings address (for instance, by storing lots of backups, including in the cloud), and extreme precautions against theft of the guardian address. Although reasonable precautions should be taken, if your savings address gets hacked, or the key to your guardian address gets lost or destroyed, the coins can still be recovered.

This model also facilitates estate planning. You simply give your heir(s) a paper copy to the private key of your savings address, but you keep the guardian address key to yourself. If you die, your heirs can simply transfer the funds out of your savings (they will have to wait for the reversibility period to pass), but they can't steal from you while you are alive since you are the only one with the key to the guardian address and can reverse their transaction if they try.

It should be obvious that anyone accepting MasterCoins or MasterCoin-derived currencies for payment should check that the payment is not reversible before completing the transaction!

Selling MasterCoins for Bitcoins

Say you want to publish an offer to sell 1.5 MasterCoins for 1000 bitcoins. Doing this takes 33 bytes, which fits into two data payments:

1. Transaction type = 20 for currency trade offer for bitcoins (32-bit unsigned integer, 4 bytes)
2. Currency identifier for sale = 1 for MasterCoin (32-bit unsigned integer, 4 bytes)
3. Amount for sale = 150,000,000 (1.50000000 MasterCoins) (64-bit unsigned integer, 8 bytes)
4. Amount of bitcoins desired = 100,000,000,000 (1000.00000000 bitcoins) (64-bit unsigned integer, 8 bytes)
5. Time limit = 10 (10 blocks in which to send payment after counter-party accepts these terms) (8-bit unsigned integer, 1 byte)
6. Minimum bitcoin transaction fee = 10,000,000 (require that the buyer pay a hefty 0.1 BTC transaction fee to the miner, discouraging fake offers) (64-bit unsigned integer, 8 bytes)

Selling MasterCoins for Other MasterCoin-Derived Currencies

Say you want to publish an offer to sell 2.5 MasterCoins for 50 GoldCoins (coins which each represent one ounce of gold, derived from MasterCoins and described later in this document). For the sake of example, we'll assume that GoldCoins have currency identifier 3. Doing this takes 28 bytes, which fits into two data payments:

1. Transaction type = 21 for currency trade offer for another MasterCoin-derived currency (32-bit unsigned integer, 4 bytes)
2. Currency identifier for sale = 1 for MasterCoin (32-bit unsigned integer, 4 bytes)
3. Amount for sale = 250,000,000 (2.50000000 MasterCoins) (64-bit unsigned integer, 8 bytes)
4. Currency identifier desired = 3 for GoldCoin (32-bit unsigned integer, 4 bytes)
5. Amount of GoldCoins desired = 5,000,000,000 (50.00000000 GoldCoins) (64-bit unsigned integer, 8 bytes)

Changing an Offer

Say you decide you want to change the number of coins you are offering for sale, or change the asking price. Simply re-send the offer with the new details. If your change gets into the block chain before someone accepts your old offer, your offer has been updated.

If you decide you want to cancel an offer, simply send the offer again, but enter the number of coins for sale as zero.

Purchasing a Currency Offered For Sale

Say you see an offer such as those listed above, and wish to accept it. Doing so takes 16 bytes, which fits into 1 data payment:

1. Transaction type = 22 for accepting currency trade offer (32-bit unsigned integer, 4 bytes)
2. Currency identifier you are purchasing = 1 for MasterCoin (32-bit unsigned integer, 4 bytes)
3. Amount you are purchasing = 130,000,000 (1.30000000 MasterCoins) (64-bit unsigned integer, 8 bytes, should not exceed number available for sale, but if it does, assume buyer is purchasing all of them)

The reference address should point to the seller's address, to identify whose offer you are accepting.

If you are purchasing with bitcoins, make sure your total expenditures on bitcoin transaction fees while accepting the purchase meet the minimum fee requested. You will need to send the appropriate amount of bitcoins before the time limit expires to complete the purchase. Note that you must send the bitcoins from the same address which initiated the purchase. If you send less than the correct amount of bitcoins, your purchase will be for that amount.

If you are purchasing with MasterCoin or a MasterCoin-derived currency such as GoldCoins, your purchase is complete as soon as you accept the offer (assuming you are recorded in the block chain as the first to accept the offer). If you have less than the correct amount on hand, your purchase will be for that amount.

Note that when only some coins are purchased, the rest are still for sale with the same terms.

Registering a Data Stream

Say you decide you would like to start publishing the price of Gold in the block chain. Registering your data stream takes a varying number of bytes due to the use of null-terminated strings. This example uses 57 bytes, which fits in 3 data payments:

1. Transaction type = 30 for registering a data stream (32-bit unsigned integer, 4 bytes)

2. Parent Currency Identifier = 1 for MasterCoin (32-bit unsigned integer, 4 bytes) (Meaning that the price of Gold will be published in units of MasterCoin)

3. Category = “Commodities\0” (12 bytes)

4. Sub-Category = “Metals\0” (7 bytes)

5. Label = “Gold\0” (5 bytes) (if a second “Gold” is registered in this sub-category, it will be shown as “Gold-2”)

6. Description/Notes = “tinyurl.com/kwejgoig\0” (22 bytes) (Please save space in the block chain by linking to your description!)

7. Display Multiplier = 10,000 (if the ticker publishes 0.00150000, the price of an ounce of gold is currently 15.0000 MasterCoins. (32-bit unsigned integer, 4 bytes)

The reference payment should be to the bitcoin address which will be publishing the data. Only the first payment sent from that address in a given day (as determined by block-chain timestamps) will be considered ticker data.

Each data stream gets a unique identifier, determined by the order in which they were registered. For instance, if your data stream was the third data stream ever registered, your data stream identifier would be 3.

Since anyone can cheaply register a data stream, and thereby create categories and subcategories, we can assume that there will be a lot of noise. Anyone writing code to display data stream categories should note which data streams are the most actively used, and order categories and subcategories by descending activity, thereby pushing unused categories to the bottom of the list.

If you ever need to change the description/notes for your data stream (for instance, if some poor sport takes down your website), simply re-register it from the same address with the same category, subcategory, and label. When re-registering, you can also change the ticker address by choosing a different address for the reference payment (for instance, if your ticker address gets compromised), or change the display multiplier.

Offering a Bet

Say you want to use USDCoins (another hypothetical currency derived from MasterCoin, each USDCoin being worth one U.S. Dollar) to bet $200 that the gold ticker will not rise above 20 MasterCoins/Ounce in the next 30 days at 2:1 odds. For the sake of example, we will assume that USDCoins have currency identifier 5. Creating this bet takes 36 bytes which fits into 2 data payments

1. Transaction type = 40 for creating a bet offer (32-bit unsigned integer, 4 bytes)
2. Bet Currency identifier = 5 for USDCoin (32-bit unsigned integer, 4 bytes)
3. Data Stream identifier = 3 for the Gold ticker, per our data stream example (32-bit unsigned integer, 4 bytes)
4. Bet Type = 35 for "Will not exceed on or before" (See table below) (16-bit unsigned integer, 2 bytes)
5. Bet threshold = 200,000 (0.00200000 BTC, which equates to a ticker value of 20 per our data stream example) (32-bit unsigned integer, 4 bytes)
6. Days out = 30 (16-bit unsigned integer, 2 bytes)
7. Amount of wager = 20,000,000,000 (200.00000000 USDCoins) (64-bit unsigned integer, 8 bytes)
8. Amount of counter-wager = 10,000,000,000 (100.00000000 USDCoins) (64-bit unsigned integer, 8 bytes)

By offering $200 against $100, the desired 2:1 odds are implied. It is not possible to change a bet (you must cancel and then broadcast a new bet). To cancel your bet, rebroadcast it with all the same data except set the amount of wager to zero.

Table of Bet Types

0	Will equal on	32	Will equal on or before
1	Will not equal on	33	Will not equal on or before
2	Will exceed on	34	Will exceed on or before
3	Will not exceed on	35	Will not exceed on or before
4	Will be below on	36	Will be below on or before
5	Will not be below on	37	Will not be below on or before

Accepting a Bet

Say you see a bet which you would like to accept. Simply publish the inverse bet with matching odds, and the MasterCoin protocol will match them automatically (that is, everyone parsing MasterCoin data will mark both bets as accepted). Here is what a bet matching our last example published 5 days later (with 25 days to go) would look like:

1. Transaction type = 40 for creating a bet offer (32-bit unsigned integer, 4 bytes)
2. Bet Currency identifier = 5 for USDCoin (32-bit unsigned integer, 4 bytes)
3. Data Stream identifier = 3 for the Gold ticker, per our data stream example (32-bit unsigned integer, 4 bytes)
4. Bet Type = 34 for "Will exceed on or before" (See table above) (16-bit unsigned integer, 2 bytes)
5. Bet threshold = 200,000 (0.00200000 BTC, which equates to a ticker value of 20 per our data stream example) (32-bit unsigned integer, 4 bytes)
6. Days out = 25 (16-bit unsigned integer, 2 bytes)
7. Amount of wager = 5,000,000,000 (50.00000000 USDCoins) (64-bit unsigned integer, 8 bytes)
8. Amount of counter-wager = 10,000,000,000 (100.00000000 USDCoins) (64-bit unsigned integer, 8 bytes)

Note that this bet will be matched against only half of the previous example, because while the odds match (2:1 vs. 1:2), the amount of this bet is for less. This bet is only for $50, so would only win $100 if they win, as opposed to the full $200. Once the bets are matched, the first bet still has $100 available for someone else to bet $50 against.

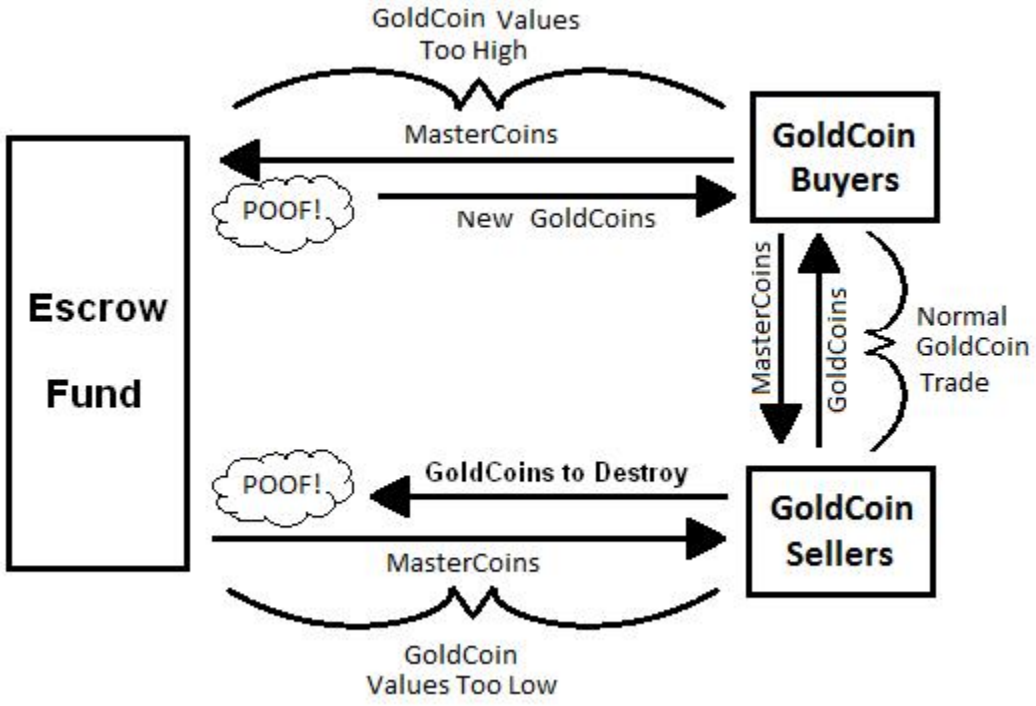
Once GoldCoins reach a value of 20 or the bet deadline passes, the bet winner gets 99.5% of the money at stake. The other 0.5% goes to the creator of the data stream.

User Currencies

The most important feature of MasterCoins is the built-in support for users to create their own currencies out of existing MasterCoins. For the purposes of demonstrating how user currencies will work, we will use an example currency called “GoldCoins”, which are intended to track the value of one ounce of gold, and which may be stored, transferred, bought, and sold similarly to MasterCoins.

Stability Concept

So how do we drive the value of these GoldCoins to their target value, when demand for them may surge and decline? The price of GoldCoins is decided by the balance of supply and demand. Since we can’t control the demand for GoldCoins, we must control the supply. The key to accomplishing this is to use an escrow fund which holds MasterCoins, as shown below:



The escrow fund operates like a battery on the power grid, charging when there is excess energy then discharging where there isn't enough. When there are too few GoldCoins (GoldCoin price is too high), the escrow fund releases new GoldCoins, and the escrow-battery “charges” by holding MasterCoins in escrow. When there are too many GoldCoins (GoldCoin price is too low), the escrow fund purchases some of the excess GoldCoins, thereby “discharging” the escrow-battery as it releases the stored MasterCoins.

New Currency Creation

Say you want to create the GoldCoin currency described above, using the Gold data stream we defined. Doing so will use a varying number of bytes, due to the use of a null-terminated string. This example uses 22 bytes, which fits in 2 data payments:

1. Transaction type = 100 for creating a new child currency (32-bit unsigned integer, 4 bytes)
2. Data Stream identifier = 3 for the Gold ticker, per our data stream example (32-bit unsigned integer, 4 bytes)
3. Escrow fund delay = 4 for 4 days (see below) (8-bit unsigned integer, 1 byte)
4. Escrow fund aggression factor = 1,000,000 for 1% (See below) (32-bit unsigned integer, 4 bytes)
5. Currency Name = “GoldCoin\0” (9 bytes)

As with data streams, currencies are awarded currency identifiers in the order in which they are created. MasterCoin is currency identifier 1 (bitcoin is 0) so if GoldCoin is the second MasterCoin-derived currency, it will get a currency identifier of 3.

The currency held in escrow is the parent currency of the data stream. In this example it is MasterCoins, but it could also be any currency derived from MasterCoins. For instance, GoldCoins could later be held in escrow to support a currency whose data stream uses GoldCoins as a parent currency.

The escrow fund delay of 4 days means that the price of GoldCoins must be too high (or too low) for 4 days in a row before the escrow fund will take any action.

The escrow fund aggression factor determines how aggressively the escrow fund corrects the price of GoldCoins when their price diverges from their target. An escrow fund with aggression factor of 0 will never take any action. If the aggression factor is 100%, the escrow fund will take the maximum possible action (buying every GoldCoin for sale above the target price, or selling new GoldCoins to every buyer below the target price).

In the case of a 1% aggression factor, the escrow fund's first action will be to fix 1% of the error. If the error the next day is still in the same direction, the escrow fund will fix 2% of the error, then 3% the next day, and so on until it reaches 100% or the error changes direction. Once the error changes its direction, the escrow fund has done its job and it starts counting again from zero.

Unhealthy Escrow Funds

What if the price of MasterCoins falls 95%, and the value of the escrow fund is now only 5% of the target value of all GoldCoins? Using the battery analogy, this escrow fund now has less “charge” and is therefore less capable of intervening to correct prices.

The protocol handles this situation by adjusting the aggression factor accordingly. In the example of GoldCoins given above, the 1% aggression factor would be multiplied by 5% to get 0.05%, meaning that the adjustments will be of much smaller magnitude, and it will take a lot longer to get to 100% aggression.

Note that escrow funds holding funds worth more than their currency do not get more aggressive. That is, if the GoldCoins escrow fund is worth twice the value of all GoldCoins in existence, the aggression factor is still 1%.

Maintaining Escrow Fund Health

Given a reasonably stable MasterCoin, escrow funds should generally grow healthier over time. Our GoldCoin escrow fund, when it does act, is buying GoldCoins when they are cheap, and selling them when they are expensive. Thus it will generally tend to make a profit, and the MasterCoins held by the escrow fund will grow. The larger the escrow fund, the lower the chance of the currency failing to maintain its value.

When an escrow fund is unhealthy, lowering the aggression factor makes the escrow fund take more profitable trades, which increases the likelihood of recovery. For instance, if it is buying excess GoldCoins, the cheapest 0.05% can be purchased at a better average price than the cheapest 1% on the market.

Escrow funds should generally be tuned to act slowly. This will allow arbitrage traders to do the heavy lifting, as the knowledge that the escrow fund will eventually get the price back to the target makes for a self-fulfilling prophecy when traders act on that knowledge. If the escrow fund acts too quickly, it loses money when the bitcoin version of a security leads the real-world version, as would happen if someone was engaging in insider trading anonymously using the bitcoin version.

Appendix A – Horrible, Awful, Very Bad Things

Speaking of insider trading, it should be clear by now that MasterCoins can be used for some very bad things. Anyone working on an implementation of the MasterCoin protocol should be very careful to warn users to not break the law of their country of residence. It is up to the user to know the laws of their country, and not (for instance) engage in sports betting in a country where sports betting is not legal.

Also, we are not securities experts, but one has to imagine that insider trading is illegal most places, if not everywhere. Also, if you think bitcoin has a reputation problem for money laundering now, just wait until you can store “USDCoins” in the block chain! At last, drug dealers and human traffickers will have the perfect currency to enable their work.

However, stable distributed currencies will be incredibly useful in a huge number of legal applications, and even modest success of this protocol could make early investors (and even those who simply hold bitcoins) very rich. MasterCoins, much like guns, are just tools, capable of being used for good or for evil. We urge our early adopters to consider how they may use MasterCoin for good, and if you get rich, how to use that money for good. It will take a lot of work to make the good outshine the evil.

Appendix B – Disclaimer

Investing in experimental currencies is really, absurdly risky. This paper is not investment advice, and anyone predicting what will happen with experimental currencies such as those described here is indulging in the wildest sort of speculation, and that includes the speculations in the previous appendix.

Please consult your financial adviser before investing in ANY wild scheme such as this (hint: they will probably tell you to RUN and not look back unless you assure them that it is money you are totally prepared to lose).

Anyone who puts their rent money or life savings into an experiment of this type is a fool, and deserves the financial ruin they will inevitably reap from this or some other risky enterprise.