Techpaper

Version: 1.4

support@aequator.io
www.aequator.io

DBC MetaChain eG & Co. KG - Techpaper

June 29, 2018

Contents

1 Abstract **3**

2 Variables and Definitions **4**

3 Starting Setup **5**
3.1 Coinspecs 5
3.1.1 Hard facts 7

4 Proof of Trust **8**
4.1 Consensus 8
4.1.1 Full Nodes - FN 8
4.1.2 Clients - CN 10
4.1.3 Mobile Devices - MD 11
4.2 Block Signing 12
4.2.1 tx Signing 12
4.2.2 Hashing 13
4.2.3 Lazy Flagging 13
4.3 Scalability 13
4.4 Block Reward 14

5 Network protocol **14**

6 Document Versions **16**

1 Abstract

This paper is intended to outline the technical aspects of the MetaChain technology including the starting concept of a fork.
It will go deeper into the proof of trust concept, chapter 4 on page 8, the consensus within the network, chapter 4.1 on page 8, the block signing techniques, chapter 4.2 on page 12, the scaling of the network, chapter 4.3 on page 13 as well as the network protocol structure, chapter 5 on page 14.
This document is aimed for persons with higher technical or cryptographic knowledge.

The current sourcecode which may already include the implementations of the described algorithms and proposals can be found in our github[1].

Do you feel you can support our team?
Do you have the skills we need and want to enrichen our team?
Please contact us via email[2] or through slack[3] in the #dev channel.

[1] `https://github.com/mc-aeq`
[2] `support@aequator.io`
[3] `https://goo.gl/TqvHN2`

2 Variables and Definitions

The following variables and definitions will be used in this document. All relevant formulas are based on these.

- tx ... Transaction
- BS ... Block Size
- BW ... Block Reward
- C ... Consensus
- FN ... Full Node
- FNL ... Full Node List
- CL ... Client
- MD ... Mobile Device
- RS ... Reed Solomon address
- BN ... Block Number where $BN_{genesis} \equiv 0$
- TF ... Trust Factor
- S ... Stake
- B ... Bonus (Good Block)
- M ... Malus (Bad Block)

3 Starting Setup

We decided to fork DECRED[4] for various reasons. Firstly, they developed a digital currency that works completely independent from outside influences and third parties. Secondly, it is based on open source technology. Thirdly, the hybrid solution of PoW and PoS fits perfectly our purpose of mutual control of these two systems. Finally, the stakeholder governance is revolutionary.

Keydata of DECRED:

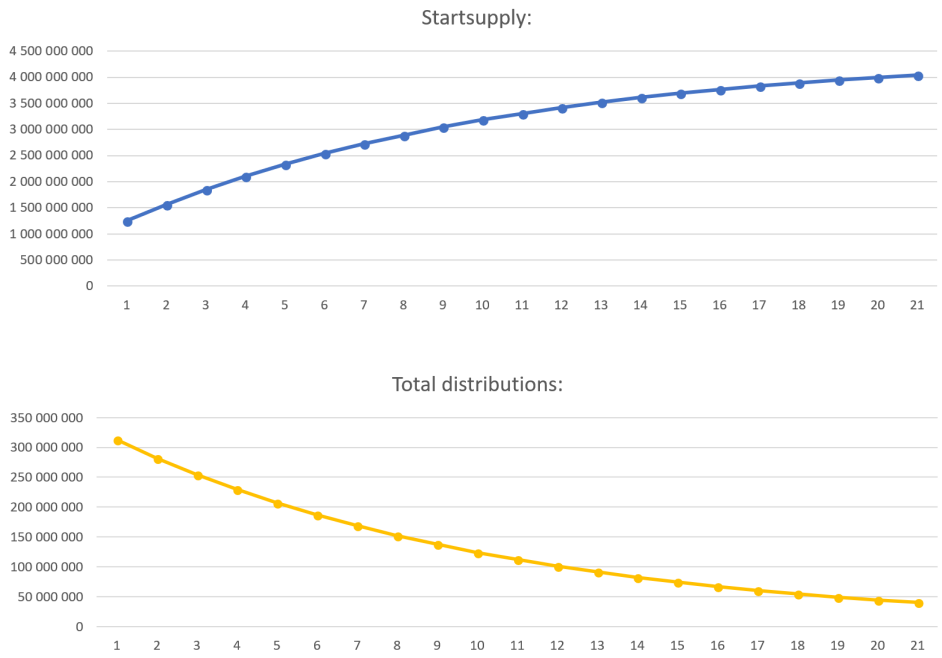
- Very experienced core development team
- Launched in 2016 by bitcoin developers that engineered btcsuite
- It takes full advantage of a distinctive hybrid of proof-of-work (PoW) and proof-of-stake (PoS)
- Voting system based on ticket purchases that reflect the PoS provisions
- Based completly on open source technology[5]
- Cross-platform wallets for easy usage

3.1 Coinspecs

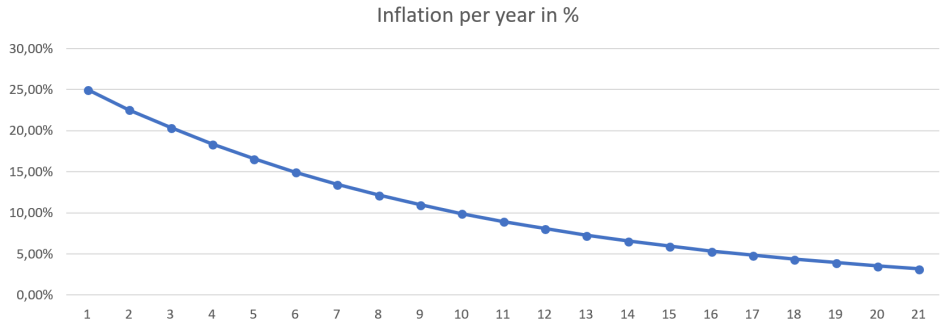
The startsupply of the DECRED based ÆQUATOR blockchain will be 1.125 billion AEQ. This supply will increase over time due to blockrewards distributed as PoW and PoS hybrid. The distribution for the blockreward is split up as follows:

- 75%: PoW
- 20%: PoS (which is split up between 5 parties (so called tickets) that are included in the PoS concept. For further information please visit the DECRED website for voting tickets[6])
- 5%: ORG. The organisational part is again split up internally to 2% for PR and 3% point of sale acceptances. The usage of the PR/Marketing budget will be transparent to the community by providing a watchable public key, as well as in regular business reports due to the organizational structure.

[4] `https://www.decred.org/`
[5] `https://github.com/decred/`
[6] `https://docs.decred.org/faq/proof-of-stake/voting-tickets/`



We have implemented an incentive that is decoupled from the block reward mechanism. There is also an additional inflation factor in the first 20 years.



AEQ has a block-reward reduction ten times a year so it will slowly decrease. That will prevent miners from affecting the network in an unproductive way, like it happened with Bitcoin.

3.1.1 Hard facts

Name (Ticker): ÆQUATOR (AEQ)

Type: PoW & PoS Hybrid (later switch to MetaChain with PoT)

PoW Algorithm: Blake2B (14r)

Block Time: 60 seconds

Difficulty Retarget : Every block

Mined / Minted Maturity : 256 blocks

Max Coin Supply: No maximum coin supply but ongoing inflation ten times a year (all 52.560 blocks)

Inflation first year: 8%

Inflation following years: 3%

Rule change activation quorum: 75%

Tickets per block: 5

Maximum fresh stakes per block: 20

P2P & RPC Port: Announced after the ICO

4 Proof of Trust

The Proof of Trust is a new concept to generate a network of clients who can be trusted. This is achieved by using three different clients on different hardware (full nodes, clients on regular devices like PCs or Raspberry PIs and clients on mobile devices) which have to agree upon the current transactions within a block. For easier purpose of writing this document, all data within a single block will be referred to as transaction, regardless whether the data is a currency transaction or some other kind of payload.
To find out what devices are allowed and selected within the proof of trust network, we need a consensus within the network which authorizes and defines the upcoming devices in future blocks. This procedure is outlined in chapter 4.1 on page 8.
The signing of the transactions within a block will be closer described in chapter 4.2 on page 12.

4.1 Consensus

A consensus within a network with unknown devices, in this case not known if they are trustworthy or not, is very hard to establish. Different algorithms like the *byzantine fault tolerance* (BFT) or modified versions of this are already well known, but in this case not needed.
BFT is needed to ensure that information within a non-trustworthy network is communicated safely and a decision is made, even though there are parties involved that are faulty. In our case we're not working with a decision making algorithm to achieve consensus within the network. We're going back to basics and use statistics in order to decide the next devices which participate in our network. Due to a malus system, faulty nodes will be detected, punished and automatically removed from the network, so that we don't have to work with them any longer. This leads us to a network that automatically establishes trust between the peers.
The network itself will always hold a buffer of the next upcoming devices (triplets) in hand, because of our forward scaling network design, chapter 4.3 on page 13. All devices in the buffer will calculate the same block information to verify the output of the other devices and possibly take over due to failure. This buffer is dynamically adjusted to be fully set for the next 5 minutes by $n_{\text{devices in the buffer}} = \left\lfloor 5 \times \frac{10 \times t_{\text{Blocktime max}}}{\sum_{i=(BN-10)}^{BN} t_{BNi}} \right\rfloor$

4.1.1 Full Nodes - FN

The FN are the one that have stored all blockchain transaction information and are the ones who generate blocks with the help of clients (CL) and mobile devices (MD). Every single FN gets rated on a scale of 0 to 65.535, with 65.535 being the most trustworthy, in future referring to as the trust factor TF - this represents 2 bytes for easier network communication. The TF will be established with the following variables and formulas

that are calculated on every node, for every other node - $\forall FN \in FNL$. This allows the network to make the decision very easy, since the one with the highest trust is the next one which will generate a block. This also eliminates the point where faulty nodes may compromise the network. If the decision is not correct as per statistics, they will get a penalty and sooner or later removed from the network, thus making their faulty input irrelevant $\models$.
The following formulas are based on $\mathbb{N}$ and $\lfloor a, b \rfloor, \lceil a, b \rceil$ are defined as chooser of either the bigger or smaller number in $\mathbb{N}$. The resulting TF is $\mathbb{Z}$. In case of TF being the same for two FN, the age of the wallet is used as tiebreaker.

t_{wallet} : The age when the connected wallet was first created or used.
Maximum Value: 20% (13.107 TF)
Formula: $13107 \times \frac{BN_{\text{current height}} - BN_{\text{first appearance}}}{BN_{\text{current height}}}$

t_S : The age and value of the stake connected with the wallet. This is a time based function with a reset once this FN generated a block.
Maximum Value: 20% (13.107 TF)
Formula: $13107 \times \lceil \frac{S_{\text{wallet}}}{|S_{\text{BNcurrent height - 2880}} - S_{\text{BNcurrent height - 1440}}|}, 1 \rceil \times (1 - \frac{BN_{\text{last generated block}}}{BN_{\text{current block}}})$

B : A good block in terms of successful block signing

M : A bad block where block signing didn't work, or any other malus within the system.
Maximum Value: 60% (39.321 TF)
Formula: $39321 \times \lfloor \frac{\sum B+1-2^{\sum M}+1}{\sum B+1}, 0 \rfloor$

Closer description of the different TF variable functions:

t_{wallet} : When we talk about trust it's also important how long a participant is continuing his work within the blockchain. The longer a person is helping the network to do their work job, the more likely it will be that he will not harm this network. Therefore t_{wallet} is a steadily increasing value each block based on the first appearance of the RS of the connected wallet.

t_S : PoS is normally heavily based on S with the premise that the higher the stake, the more likely one will not harm the system. This concept is fine, but it always works only in the favor of the ones already having enough money to buy more stakes. t_S uses the trading volume within the time range of 48h-24h, if the block time would be maximum. This is set relative to the connected stake and also resets, once a block was generated. In the beginning this function will be relevant due to fewer blocks in the system and therefore the counter value might be the full 20%. Having more blocks in the system means that we have a bigger network and thus the function reduces its importance automatically. This means that the bigger the network gets, the less we focus on S but on the B, M function. You can see an example of the t_S scaling in figure 1 on page 10.

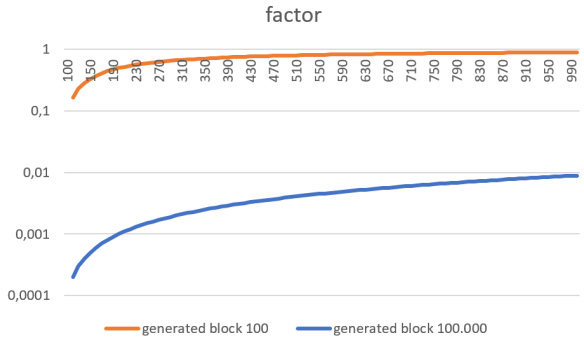


Figure 1: Staking factor with different block heights within the next 1000 blocks (16.6 hours with maximum blocktime), Y-scale is logarithmic

B, M : This is the most important factor in calculating the TF due to the 60% weight in the formula. It uses the number of good transactions and the number of bad transactions (which might not only be bad blocks but also penalties from spreading false information) to determine the trustworthiness of the node. Since we're measuring the bad transactions in 2^n, bad transactions weigh exponentially more than good transactions[7], thus allowing the network to identify and remove faulty devices easily. With this calculation it's always possible for faulty devices to become trustworthy again, but this means lots and lots of good calculations for the network.
This method also is no disadvantage for new users in the network if they want to provide a full node. When they start the amount of $B \equiv 0$ and $M \equiv 0$, leading the formula to output the full trust factor of 60%, meaning that we provide them with full trust when they start. Ergo they do have a high chance of generating a block to prove that they are trustworthy.
The decline of trust and how rapid the network looses trust in a FN is shown in figure 2 on page 11. This figure shows, that even with an astonishing 100.000 good blocks, it only takes 15 bad blocks to reduce the reputation by 50% and with 17 bad blocks there is no chance for this FN to participate anymore in the block generation.

4.1.2 Clients - CN

Clients are regular devices like PCs or Raspberry PIs that don't hold the full blockchain information and thus are not allowed to participate as FN. In the event of restocking the FN buffer, see figure 3 on page 11, the last added FN, A, takes randomly a different FN, B - step 2, which was selected to generate a block and is not within their geoIP range. B now selects, again randomly, a directly connected CN, C - step 3, to participate in this new triplet (A, C plus a to be decided MD). By doing a 2-step selection, where all steps are broadcasted to the other triplets in the buffer, a manipulation can be easily

[7] `https://en.wikipedia.org/wiki/Power_of_two`

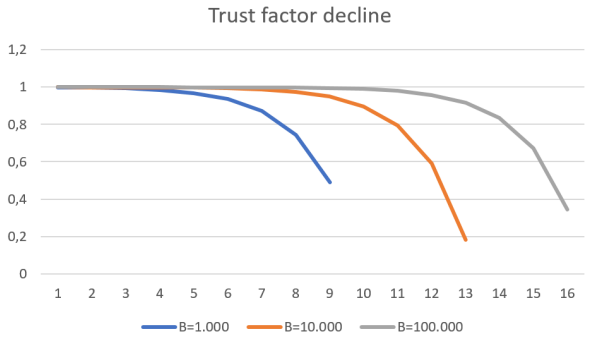


Figure 2: Trust factor decline with more M regarding to different B

seen and punished in the network.

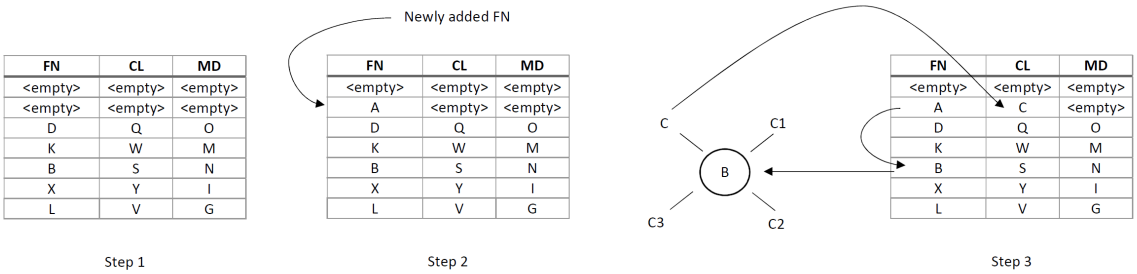


Figure 3: 2 step process of finding a new CL for a new FN in the block generation process

4.1.3 Mobile Devices - MD

Selecting MD in the process is quite similar to selecting CN. The only difference is, that the MD on the FN is not selected randomly; the selection is based on the wallet usage of this client. Meaning that the more frequent the mobile client is used, this can be determined by a flag used in the transaction payload; the higher the chances are for him to participate in the block generating process, since the usage counter increases by every transaction. This is the same principle as with PoS - the more a person is using and benefiting from this technology, the more he wants this technology to succeed. Once a cell phone disconnects from the FN or gets chosen as participant in the block generating process, the local counter of their usage is reset and needs to be rebuilt.
If there is no connected MD on a FN this FN will not provide the selection service and a different FN has to be chosen. When there is a tie in the usage counter, this also counts as if there is no usage at all and all counters are set to 0, the selection process will be random.

4.2 Block Signing

The signing of a block is straight forward with any transaction included in the block, no matter what category and subject combinations they are containing (for an explanation about category and subject, see chapter 5 on page 14).
Since we're not in a *PoW* field we're not forced to use any nonce to change our hash, even though the chain is quite similar constructed to the ones with *PoW*. A generated block consists of multiple *tx* which are hashed in a *merkle tree*. If the number of the *tx* is uneven, the last *tx* shall be hashed twice to merge into a full merkle tree. The first transaction in every block is the call to the smart contract which is responsible for splitting up the block reward. The parameters of this call are yet to be defined, but it's most certain that the call will contain the block reward and the *RS* (triplet) which will receive the block reward.
This is unique to other blockchains - we don't use a basecoin transaction to send out the block reward, we use a smart contract call. This is needed in order to ensure the proper distribution of the block reward between the contributing devices. Furthermore this will allow us to integrate a dual beneficiation system for our *PoT* devices. In the beginning of the blockchain there are plenty of blocks to discover. In this timerange the bonification for the devices will be taken out of the block reward. As soon as all blocks are found, the system will switch automatically, due to the smart contract, from a blockreward beneficiation system to a transaction fee system where every *tx* will have to pay a certain fee, in order to be processed. This will be done fully automated without any changes required within the blockchain, meaning that there is no need for a manual update of any clients.

4.2.1 *tx* Signing

tx may have as a requirement to be signed, for example asset based *tx*. In this case we use elliptical curve digital signature algorithms (*ECDSA*), as most other blockchain technologies do. Due to the high incorporation of mobile devices in our network we have one major difference. *tx* that require *ECDSA* signing will have a preceding bit field in the size of 1 byte which contains a flag defining the signing algorithm. Our blockchain will accept both signing techniques: *secp256k1* and *secp256r1*.
The two algorithms are not compatible with each other, meaning a *secp256k1* signature cannot be verified with a *secp256r1* algorithm and vice versa. The reason for us to integrate both systems lies in the mobile devices. Bitcoin only supports *secp256k1* which means that all mobile wallets for Bitcoin are using software to sign the *tx* which would allow an attacker to mess with the signature and manipulate it without the user or the software recognizing it. This is due to the lack of native SDK support for *secp256k1*.
Androids SDK hardware backed key store and iOS's secure enclave both have SDK support for *secp256r1*. By including *secp256r1* as signing algorithm, we can ensure that mobile devices use underlying hardware to sign their *tx*, meaning that we increase security in mobile device transactions.

4.2.2 Hashing

Bitcoin, Ethereum and other blockchain technologies rely on sha2-256 or sha2-512 for their hashing. This is due to the fast processing and low memory overhead, allowing higher hash rates. *SHA2* was released in 2001 and is currently included in the Secure Hash Standard from *NIST*, since then are currently no major attacks on *SHA2* known. This doesn't mean that there won't be any attacks in the future and *NIST* therefore acknowledged in 2015 a new design, based on a different principle as *SHA3*.
Since *PoT*'s block generation is not based on hashing power, the only allowed hashing algorithm will be the new *SHA3* algorithm, to be prepared for future withdrawal of *SHA2* from the *SHS* by *NIST*.
SHA3 is also not vulnerable against length extension attacks[8], *SHA2* on the contrary is vulnerable.

4.2.3 Lazy Flagging

In a blockchain the most important part are the blocks and their connection with each other. This means that a tx is only valid if it's within a block and if it is signed. *Lazy Flagging* is a new technique where this part becomes optional.
One may ask why this should be integrated since it conflicts the principle of blockchain technology - this is partly true, but not every tx is required to be 100% proven before being processed. For example in messaging software it may be not needed to wait for network confirmation of a message, so the message, when flagged lazy, can be transmitted before the block is generated. The message itself will still be included in the block, but the data can be transmitted earlier. Introducing this flagging technique into the blockchain grants future applications more flexibility in data processing and handling.
To flag a tx as lazy, the corresponding bit in the preceding bit field must be set, absence of this bit will default to regular processing.

4.3 Scalability

Currently one of the bigger questions is on how a network scales by increasing demand. As it is with Bitcoin right now, there is a bottleneck whenever there are bursts of transactions due to the low block size and the autocorrecting algorithm known as difficulty, which adjusts the network to mine approximately 1 block every ten minutes.
This design leads to the problem, that with increasing demand the network will not scale when the maximum number of transactions per block per timeframe is reached, leading us to the conclusion that it is possible for this network design to reach a deadlock. The deadlock will be reached when $\varnothing \sum tx > \frac{BS}{\text{sizeof}(tx)}$

The current design flaws and increasing demand for blockchain technology in our daily lives forced us to use a forward scalability design, meaning that the more transactions

[8] `https://en.wikipedia.org/wiki/Length_extension_attack`

within the network, the faster it will get.
To achieve this we have two conditions that result in a block to be generated:

1. $\Delta t_{tx} > 60s$
2. $\sum tx = BS$

So whenever the maximum timeframe of 60s is passed or the block is full, a block will be generated. This also means that the more transactions there are within the network, the more blocks will be generated per minute leading to a forward scaling network. The maximum time for a transaction to be embedded in a block therefore is 60s and the minimum time depends on the devices in the signing process, which could be as low as 1s.

To achieve this type of scalability the PoT consensus needs to have a stack of devices for the next n blocks, so that in case of failure of generating the block or rejection, due to manipulation, the next devices can take over in this process. This also requires that all the devices in this stack are participating in the calculation for the current block, to shorten the time of recovery upon overtaking after rejection.

4.4 Block Reward

The total number of base coins is 8.978.156.324.
85% (7.631.432.875) of the coins will be generated and distributed within the first few weeks due to our algorithm. 15% (1.346.723.448) will remain in the issuance budget. Those 15% will be used as block reward within the first two years, leading to an hourly block reward of 76.860 coins, or 1281 coins per block at maximum block time. Normally the block reward would be defined as per block, due to our forward scalability, chapter 4.3 on page 13, it is possible that we have more blocks than one per minute. If this is the case the smart contract which is responsible for the distribution of the block reward, will scale the reward down to reflect the hourly coin reward. The scaling is done after the following principle: $BR = 1281 \times \frac{60}{\Delta t_{\text{block timestamp last, block timestamp new}}}$
After the two year time range when the issuance budget is fully used up, the smart contract automatically changes to use the included transaction fees in the block as reward.

5 Network protocol

The network protocol used is based on a header and payload structure. The header is defined in the file *netMessage.h* as shown in listing 1 and visualized in figure 4 on page 15.

Listing 1: Protocol Message Header

```
1 struct sHeader {
2     uint32_t        ui32tPayloadSize;
```

```
3     uint32_t        ui32tChainIdentifier;
4     uint16_t        ui16tSubject;
5     uint8_t         ui8tChecksum[4];
6 };
```

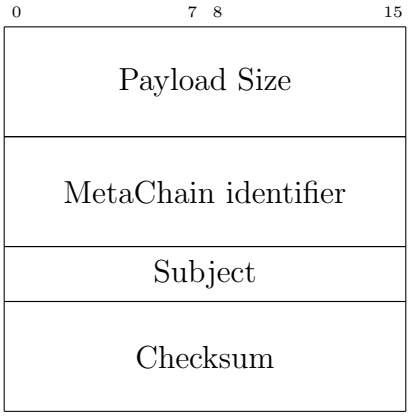


Figure 4: struct sHeader

The header has a total size of 14 bytes, including the size of the payload in bytes, a hash checksum for the payload to proof the integrity of the received data, the chain identifier and a subject. The two bytes for the subject are used to define the content or reason for this message. The subject is defined as visualized in figure 5 on page 15. We use the first

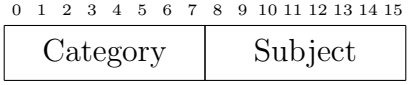


Figure 5: closer description of the subject

8 bits to define a category, for example a peer list communication. The second 8 bits are used for defining the closer action that is triggered, for example *GET_PEER_LIST*. There is also a 2 byte MetaChain identifier included in the header, because every coin within the MetaChain can have different category and subject mappings. This has to be defined by the individual coin itself - a lack of this definition by an additional module results into standard mapping, provided by the base coin. Since we have a dynamic size of payload we don't need further information in the general header. All related information about the payload can be packed into the payload and thus also is checked about integrity with the checksum. The full and updated list of category and subject fields can be found in the file *netMessage.h* in the github repository.

For a closer description of how the payload data is treated, what requirements are for specific category and subject combinations, please refer to our documents which are uploaded to github and are discussed in our slack development channels

6 Document Versions

- Version 1.0: 05.08.2017
initial commit of the document
- Version 1.1: 08.08.2017
added some graphics for better illustration and fixed some minor typos.
- Version 1.2: 17.08.2017
fixed some typos.
- Version 1.3: 11.09.2017
reworking content in order to improve readability (thanks to crowi)
- Version 1.4: 27.06.2018
including fork technical details