OpenCoin

a protocol for privacy preserving electronic cash payments

Version: 0.4 - draft (July 2022)

This version of the protocol build on previous work by the following authors:

Jörg Baach
Nils Toedtmann
J. K. Muennich
M. Ryden
J. Suhr

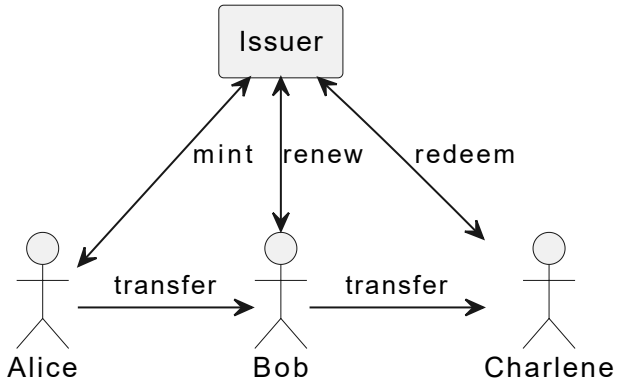
For more information go to **https://opencoin.org**

Intro

We propose a protocol that allows cash-like payments in the electronic world. It is based on the invention by David Chaum [1]. The main focus are *untraceable* payments, which means that even though there is a central entity (called the issuer, something like a bank), this central entity can't see the transactions happening. This is good for the privacy of the users.

The focus of the project is the protocol. This means we standardize the way we exchange messages and their content in order to make electronic cash payments. But we don't deliver an implementation here. That is the scope of other project(s). OpenCoin is the foundation to build upon.

How does it work?



This is a high level (but strongly simplified) image describing the basic system. We have three participants: Alice and Bob are normal users, while the Issuer is something like a bank, capable of minting coins. It also acts as an exchange for 'real-world' currency. At this high level it works as follows:

1. Alice asks the Issuer to **mint** coins. This is done in a special way using *blind signatures*, which means that the coins *can't be linked to her* later on.
2. Alice then **transfers** the coins to Bob. She can do that any way she wants, e.g. using WhatsApp, Email or any other system of her choice (also depending on what her client software supports). This could even be done by printing the coins and handing them over.
3. Bob then **renews** the coins. He swaps the coins he got from Alice for fresh coins. This way he protects himself against Alice "accidentally" using the coins

somewhere else. One can spend opencoins only once, so *double spending* needs to be ruled out, and this is done by immediately renewing received coins.
4. Bob might **transfer** the coins to yet another person, Charlene
5. Charlene decides to **redeem** the coins, meaning she asks the issuer to swap the opencoins for real-world money.

On **blind signatures**: at the core a coin is a serial number with a signature from the mint. In order to ensure that *a coin can't be traced back to the original client* we use blind signatures.
Imagine Bob hands in a coin that Alice had minted. In order to ensure the coin can't be traced back to Alice, the issuer has to sign the serial number without seeing it. In non-technical terms Alice puts the serial number in an envelope (along with carbon copy paper), and the issuer actually signs the envelope. Because of the carbon copy paper the signature presses through onto the serial number. Alice can then open up the envelope and has a signed serial, without the issuer ever seeing it.

Who is it for?

OpenCoin (the protocol) allows the development of applications for electronic cash. So firstly OpenCoin is targeted at developers. These applications however should allow everyone to make and receive electronic payments. It still requires somebody to run the central issuer. This issuer would issue an OpenCoin based electronic money system. Because electronic money is quite regulated in Europe (and other countries), the issuer would be most likely a regulated electronic money provider or a bank. We think, that a central bank would be the best issuer, because central banks issue money anyhow. But nothing technical stops you from using OpenCoin for your private project [2].

Alternatives

Why don't just use one of the alternatives?

Bitcoin / blockchain

Bitcoin (or blockchain in the more general form) is basically the opposite of OpenCoin: transfers have to happen within the system, they are visible to everybody, there is no central instance, there is no guaranteed value you can redeem the bitcoins for.

OpenCoin on the contrary makes the transfers invisible and untraceable, and has a central instance that is able to guarantee a value if you redeem the OpenCoin.

One could say that bitcoin behaves more like gold, while OpenCoin behaves more like cash.

GNU Taler

GNU Taler is build around the same central idea as OpenCoin. It started later, and is more complete than OpenCoin. They differ in the way the take care of the renewal step and coin splitting. They also make more assumptions regarding the clients (e.g. clients having key identifying them), they have clearer roles (e.g. consumer and merchant) and by all of this hope to get around the inherent problems of untraceable transfers, e.g. tax-ability.

The trade-off seems to be that their system is harder is more complex and harder to understand. We also doubt that these complexities are necessary to reach the stated goals. We also doubt that the goals can really be reached, and also find that the system's documentation is quite hard to understand. This might be because they deliver implementations for all necessary software components, and are not really targeted at other implementations of they system.

Because of all this one could say that GNU Taler is less open to other developers.

Problems

- Tax
- Money laundering
- Blackmail and other crime

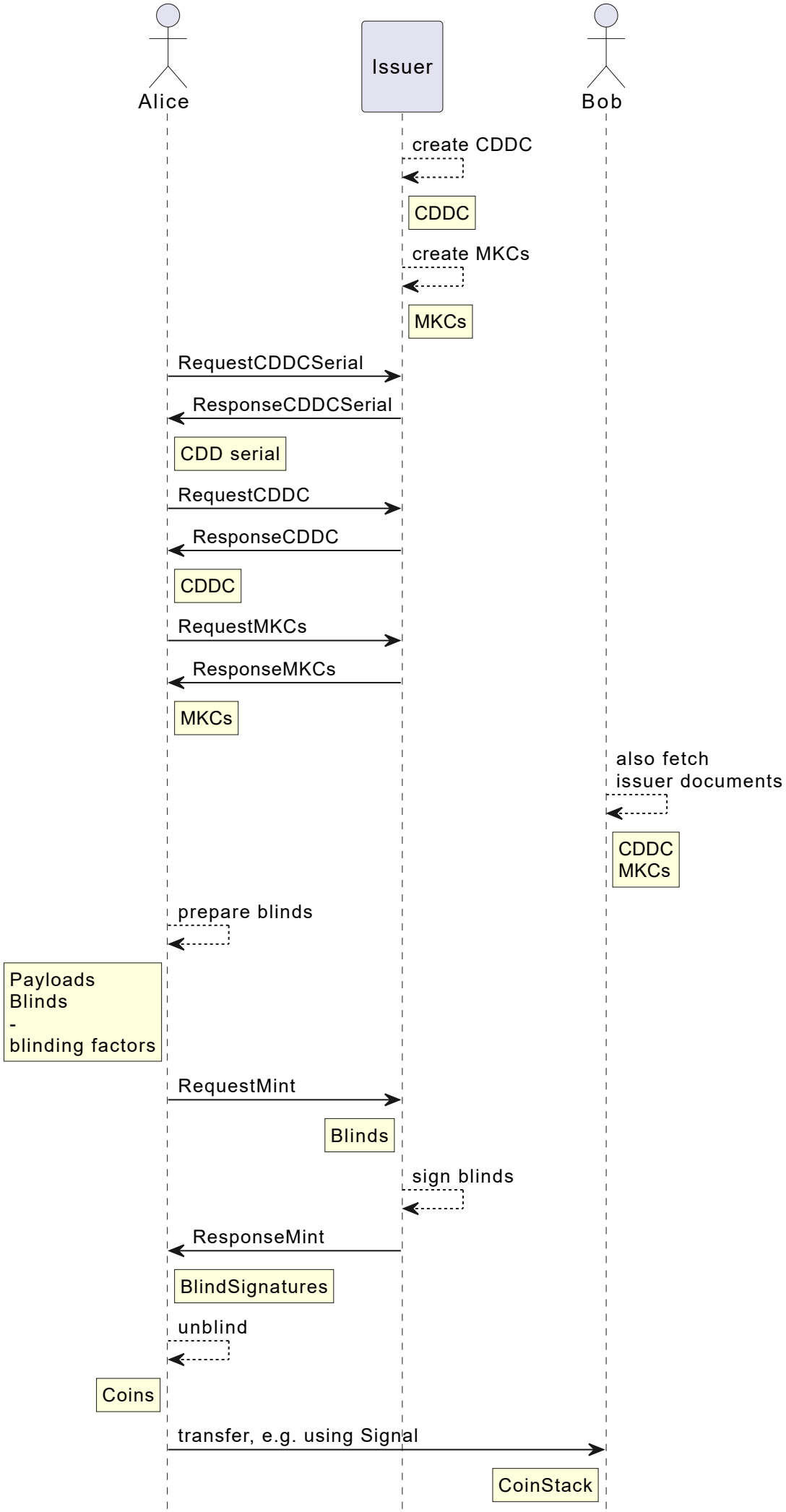
The OpenCoin protocol

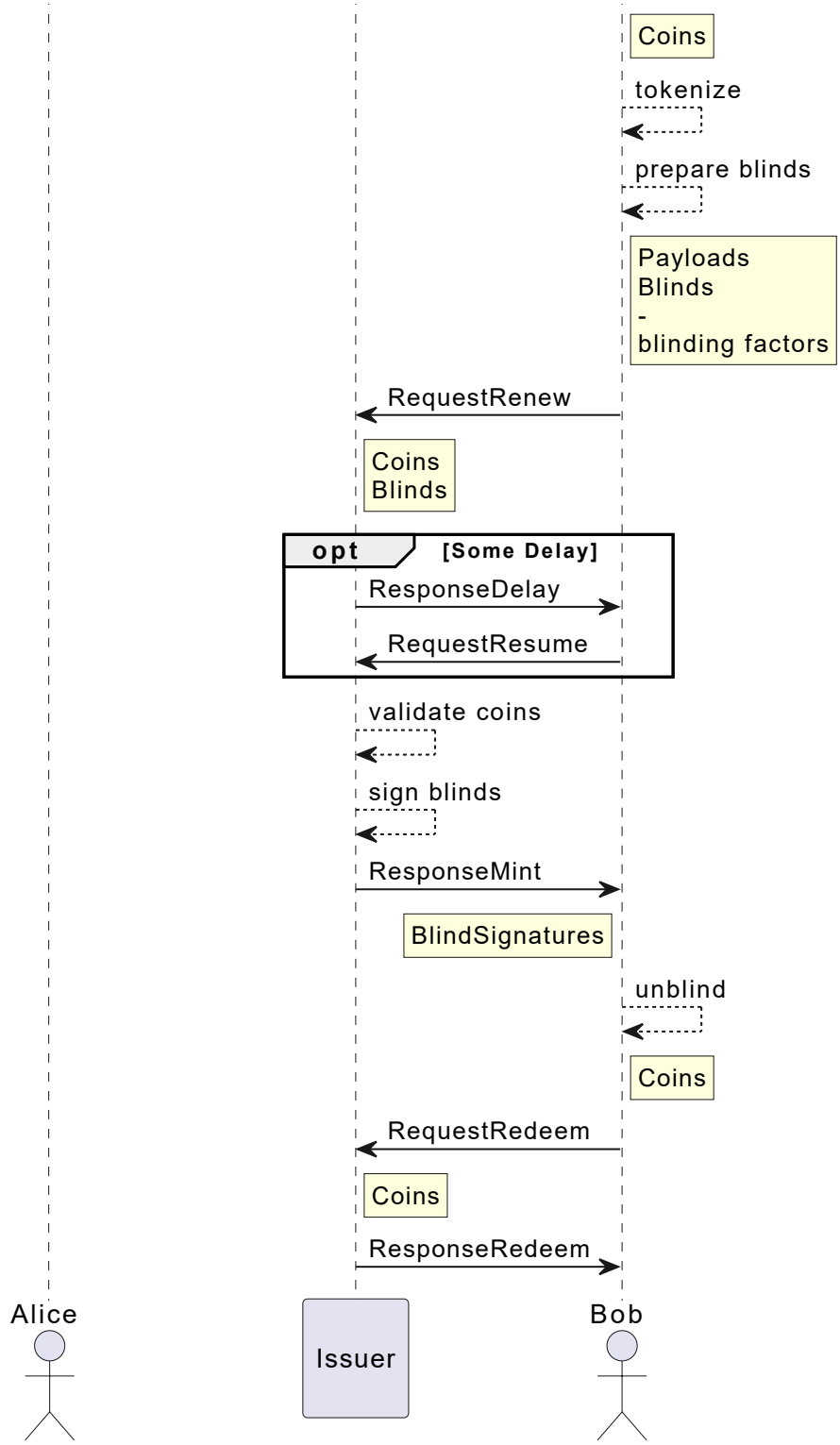
Assumptions

The exchange of messages MUST happen over a **secure channel**. For HTTPS this means TLS, but other channels like messengers provide their own. For an email exchange GPG would be recommended. Either way, it is the responsibility of the developer to take care of the transport security.

When requesting the issuer to mint or redeem coins some form of **authentication & authorization** is most likely required - the issuer needs to secure payment for the coins, or make a payment somewhere for redeemed coins. Because auth* might already be provided by the transport layer, we don't include it in the OpenCoin protocol.

Overview


Alice
Issuer
Bob
create CDDC
CDDC
create MKCs
MKCs
RequestCDDCSerial
ResponseCDDCSerial
CDD serial
RequestCDDC
ResponseCDDC
CDDC
RequestMKCs
ResponseMKCs
MKCs
also fetch
issuer documents
CDDC
MKCs
prepare blinds
Payloads
Blinds
-
blinding factors
RequestMint
Blinds
sign blinds
ResponseMint
BlindSignatures
unblind
Coins
transfer, e.g. using Signal
CoinStack



Description

This is a high level description of the actual steps, details follow in the chapters in Details.

Participants

Issuer can mint, refresh and redeem coins. This entity will probably an account handling system (a.k.a. bank) behind it for doing actual real-world payments. The issuer is trusted to handle coins and payments correctly, but is *not trusted* regarding privacy - the target of OpenCoin is to protect the privacy of the transfers.

Alice and **Bob** are clients using OpenCoin. Technically they are software clients, and they represent the *users* of the system.[3] They need to be known by the customer in order to mint or redeem coins. Authentication could be required to renew coin. This would allow a "closed" system, in which accounts of the users could be monitored.

Steps in the protocol

create CDDC

The issuer creates a pair of cryptographic keys (the currency keys), and signs a *Currency Description Document Certificate* (CDDC) with its secret key. This contains information about the currency, like denominations, urls but also the public key. This is the top document which establishes the trust in all other elements.

Not mentioned in the CDDC but probably somewhere on the issuer website is the relation between opencoins and actual real-world money. Let's say the currency of an example issuer is called "opencent".[4] The rule might be that one opencent is given out for one EUR cent, and redeemed for one EUR cent, effectively binding the opencent to the EUR.

create MKCs

For each denomination in the currency separate minting keys are generated, and a *Mint Key Certificate* (MKC) for them as well. Those MKCs are signed the secret currency key. The mint keys are only valid for a defined period of time.[5]

RequestCDDCSerial

RequestCDDCSerial asks for the current serial number of the CDDC. The currency description could change over time, maybe because urls have changed. Every time a new CDDC is created, with a new, increasing serial number. The clients need to make sure to always use the most current CDDC, but they can cache it, allowing them to skip the next

step.

ResponseCDDCSerial

ResponseCDDSerial contains the current serial of the CDDC.

RequestCDDC

RequestCDDC asks for a CDDC. If no serial is provided, the message asks for the most current CDDC.

ResponseCDDC

ResponseCDDC contains the CDDC

RequestMKCs

RequestMKCs asks for the Mint Key Certificates. The client can specify specific denominations or *mint key ids*. An unspecified request will return all current MKCs.

ResponseMKCs

ResponseMCKs contains the MKCs

prepare blinds

This step prepares a coin. In essence this is a Payload with a serial number, which is later on signed by the issuer using a denomination specific mint key. The "envelope" mentioned above really means that the serial is blinded using a separate random secret **blinding factor** for each serial number. This factor is needed later on to "open up the envelope", reversing the blinding operation. Hence, the client has to store the blinding factor for later on. As the blinding factor is individual for each serial number, a reference number is created to reference serial, blinding factor and Blind.

The blinds contain the reference, the blind to be signed, and the mint key id for the denomination or value of the coin.

RequestMint

RequestMint hands in the Blinds created in the step before, asking for the blind to be signed.

Most likely the issuer has authenticated the client. The mint key id tells the issuer what denomination to use for the signing operation. This will allow the issuer to deduct a payment for the minting operation (outside OpenCoin).

The message also carries a transaction_reference (a random number), in case there is a delay in the minting process. The client can then later on ask again for the signatures to be delivered using the same transaction_reference.

sign blinds

The issuer uses the secret minting key for the desired operation to sign the Blind, creating Blind Signatures.

ResponseMint

ResponseMint contains the Blind Signatures for the Blinds.

unblind

The client will unblind the Blind Signature using the before stored secret blinding factor. This gives the client the signature for the serial number, and both together give the Coin.

CoinStack

When sending coins multiple coins can be combined into a CoinStack. This CoinStack can also have a "subject", maybe containing an order reference - the reason the CoinStack is handed over in the first place.

The transfer of the CoinStack is out of scope of the OpenCoin protocol. We imagine multiple ways: using a messenger like Signal, using email or using the Browser. A CoinStack can also be encoded using a QR code, and maybe printed out and sent using normal postal mail.

Anyhow, the point of this step is that Alice transfers a CoinStack to Bob. And because she is a fair user, she will delete all coins that were contained in the CoinStack on her side.

tokenize

Coins that are received need to be swapped for new ones, in order to protect the receiver against double spending. Bob needs to decide which new coins he wants to have, and tokenize amount in the right way to have a good selection of future coin sizes. [6]

prepare blinds

Knowing the right coin selection from the step before Bob prepares Blinds the same way Alice has done with hers, creating Payloads (containing serials), and storing the blinding secrets under a reference.

RequestRenew

The renewal process is effectively the same as in minting new coins, but it is paid for in opencoins, instead of making a payment in the background using accounts. Hence, the RequestRenew message needs to contain Coins that have a value that matches the sum of value of the Blinds. The message also contains a transaction_reference in case a delay happens.

ResponseDelay

This step is optional.

If something takes a while at the issuer when signing the blinds, either while handling a RequestMint or RequestRenew a ResponseDelay can be sent back to indicate that the client should try again sometime later. This allows the network connection be closed in the meantime. Hopefully operations resume in a short time.

Delays should be avoided on the issuer side.

RequestResume

Bob will try a suitable amount of time later on to resume the transaction (the renewal in this case). The RequestResume message will send over the transaction reference, and the issuer will hopefully respond with a ResponseMint message, or with another ResponseDelay.

validate coins

Bob validates the Coins, just as Alice did.

RequestRedeem

Bob might want to swap some or all of the Coins he holds for real-world currency at the issuer. He sends in the coins in a RequestRedeem message. This effectively takes the coins out of circulation, and the issuer will make a payment to Bob's account. This requires the client to be authenticated for this step, which again is outside the OpenCoin protocol.

ResponseRedeem

The issuer confirms that everything went ok using the ResponseRedeem message.

Details

Cryptographic operations

- hashes
- signatures

Building blocks

Elements of messages, but never used standalone

RSA Public Key

Fields

- **modulus**:
- **public_exponent**:
- **type**:

Example

```
{
  "modulus":
"8004826974ed9eecc9261c6a695cd3f1bd33710ef3ba1ca8fbb1425d20f305020e7c80904d6d6e8a4358bf92
6f920e6167c2c780d9f34db6abe06a51c8ff2571",
  "public_exponent": 65537,
  "type": "rsa public key"
}
```

Source

Fields

- **cdd**:
    - **additional_info**:
    - **cdd_expiry_date**:
    - **cdd_location**:
    - **cdd_serial**:
    - **cdd_signing_date**:
    - **currency_divisor**:
    - **currency_name**:
    - **denominations**:
    - **id**:
    - **info_service**:
    - **invalidation_service**:
    - **issuer_cipher_suite**:
    - **issuer_public_master_key**:
    - **protocol_version**:
    - **renewal_service**:
    - **type**:
    - **validation_service**:
- **signature**:
- **type**:

Example


```
{
  "cdd": {
    "additional_info": "",
    "cdd_expiry_date": "2023-07-08T20:09:52.501723",
    "cdd_location": "https://opencent.org",
    "cdd_serial": 1,
    "cdd_signing_date": "2022-07-08T20:09:52.501723",
    "currency_divisor": 100,
    "currency_name": "OpenCent",
    "denominations": [1, 2, 5],
    "id": "85c24031572f2e0a04a41a29eb74990f4651c7f0b4afc0b53cfa03bed30822e1",
    "info_service": [
        [10, "https://opencent.org"]
      ],
    "invalidation_service": [
        [10, "https://opencent.org"]
```

```
      ],
    "issuer_cipher_suite": "RSA-SHA512-CHAUM86",
    "issuer_public_master_key": {
      "modulus":
"8004826974ed9eecc9261c6a695cd3f1bd33710ef3ba1ca8fbb1425d20f305020e7c80904d6d6e8a4358bf9
26f920e6167c2c780d9f34db6abe06a51c8ff2571",
      "public_exponent": 65537,
      "type": "rsa public key"
    },
    "protocol_version": "https://opencoin.org/1.0",
    "renewal_service": [
        [10, "https://opencent.org"]
      ],
    "type": "cdd",
    "validation_service": [
        [10, "https://opencent.org"],
      [20, "https://opencent.com/validate"]
    ]
  },
  "signature":
"2bfa4a4c85a49f7c0493bef54cef40892cb23a613b3268d21689493f5a7825e93b22baa8cfc59f8dbf79d59
16348e586eb046f16a16cda7182e7e85d9746e7ff",
  "type": "cdd certificate"
}
```


Source

MKC

A *Mint Key Certificate*.

Fields

- **mint_key**:
    - **cdd_serial**:
    - **coins_expiry_date**:
    - **denomination**:
    - **id**:
    - **issuer_id**:
    - **public_mint_key**:
    - **sign_coins_not_after**:
    - **sign_coins_not_before**:
    - **type**:
- **signature**:

- **type**:

Example


```
{
  "mint_key": {
    "cdd_serial": 1,
    "coins_expiry_date": "2023-10-16T20:09:52.501723",
    "denomination": 1,
    "id": "bac419d0d8c235e31dae3d5419944e904169c12c3799087f4f9684176fd76d05",
    "issuer_id": "85c24031572f2e0a04a41a29eb74990f4651c7f0b4afc0b53cfa03bed30822e1",
    "public_mint_key": {
      "modulus":
"cdabcaff7484d35f43a7d9e2f51eabe23783c351be84e4ed39f955a012357ebdf56e71e1ac0c15994317b23
f45345acdd03bc02af9cd1dd72143ce33b26b4d27",
      "public_exponent": 65537,
      "type": "rsa public key"
    },
    "sign_coins_not_after": "2023-07-08T20:09:52.501723",
    "sign_coins_not_before": "2022-07-08T20:09:52.501723",
    "type": "mint key"
  },
  "signature":
"71b1c58d449634ca3cf719f82ba324573d7c32c7a18c6f25e7432d3efcc9fb4d661e5a9087f3ed5184d2e59
87784cb50ae8bb354479401869cc13ac2db8ae790",
  "type": "mint key certificate"
}
```


Source

Payload

Fields

- **cdd_location**:
- **denomination**:
- **issuer_id**:
- **mint_key_id**:
- **protocol_version**:
- **serial**:
- **type**:

Example

```
{
  "cdd_location": "https://opencent.org",
  "denomination": 1,
  "issuer_id": "85c24031572f2e0a04a41a29eb74990f4651c7f0b4afc0b53cfa03bed30822e1",
  "mint_key_id": "bac419d0d8c235e31dae3d5419944e904169c12c3799087f4f9684176fd76d05",
  "protocol_version": "https://opencoin.org/1.0",
  "serial": "93608b9fe7375a19df2ee880639ceb63cab925e111c1adcf564d96738be9cb75",
  "type": "payload"
}
```

Source

Blind

Fields

- **blinded_payload_hash**:
- **mint_key_id**:
- **reference**:
- **type**:

Example

```
{
  "blinded_payload_hash":
"c6db722a94f7c500878c13cb9025d6003e6611db066ea71dc1c34b2005933b6431314ae12719394679c7623a
69f637dad6ecce300c36988da9d6df3e8c384815",
  "mint_key_id": "bac419d0d8c235e31dae3d5419944e904169c12c3799087f4f9684176fd76d05",
  "reference": "a0",
  "type": "blinded payload hash"
}
```

Source

Blind Signature

Fields

- **blind_signature**:
- **reference**:
- **type**:

Example

```
{
  "blind_signature":
"68f1e187086ad2d6333cc6b798397dda2390db6abd3ea603557afa54ac65600f5048232a34922bbcb4c7584f
4dd4004500b45d79ef43f33673defab6dc109186",
  "reference": "a0",
  "type": "blind signature"
}
```

Source

Coin

Fields

coin

- **payload**:
- **signature**:
- **type**:

Example

```
{
  "payload": {
    "cdd_location": "https://opencent.org",
    "denomination": 1,
    "issuer_id": "85c24031572f2e0a04a41a29eb74990f4651c7f0b4afc0b53cfa03bed30822e1",
    "mint_key_id": "bac419d0d8c235e31dae3d5419944e904169c12c3799087f4f9684176fd76d05",
    "protocol_version": "https://opencoin.org/1.0",
    "serial": "93608b9fe7375a19df2ee880639ceb63cab925e111c1adcf564d96738be9cb75",
    "type": "payload"
  },
  "signature":
"1932c7352ba97a24cbdddade9747d93b22db145defbdd0441606772695f0ddb439dea17a9ce09f8ea0ef590
bea57293d40fef463372060b6eb4a50c4ab7194d0",
  "type": "coin"
}
```


Source

Messages

RequestCDDSerial Message

Fields

- **message_reference**:
- **type**:

Example


```
{
  "message_reference": 1,
  "type": "request cdd serial"
}
```


Source

ResponseCDDSerial Message

- **cdd_serial**:
- **message_reference**:
- **status_code**:
- **status_description**:
- **type**:

Fields

Example

```
{
  "cdd_serial": 1,
  "message_reference": 1,
  "status_code": 200,
  "status_description": "ok",
  "type": "response cdd serial"
}
```

Source

RequestCDDC Message

- **cdd_serial**:
- **message_reference**:
- **type**:

Fields

Example

```
{
  "cdd_serial": 1,
  "message_reference": 2,
  "type": "request cddc"
}
```

Source

ResponseCDDC Message

Fields

- **cddc**:
- **message_reference**:
- **status_code**:
- **status_description**:
- **type**:

Example

```
{
  "cddc": {
    "cdd": {
      "additional_info": "",
      "cdd_expiry_date": "2023-07-08T20:09:52.501723",
      "cdd_location": "https://opencent.org",
      "cdd_serial": 1,
      "cdd_signing_date": "2022-07-08T20:09:52.501723",
      "currency_divisor": 100,
      "currency_name": "OpenCent",
      "denominations": [1, 2, 5],
      "id": "85c24031572f2e0a04a41a29eb74990f4651c7f0b4afc0b53cfa03bed30822e1",
      "info_service": [
        [10, "https://opencent.org"]
      ],
      "invalidation_service": [
        [10, "https://opencent.org"]
      ],
      "issuer_cipher_suite": "RSA-SHA512-CHAUM86",
      "issuer_public_master_key": {
        "modulus":
"8004826974ed9eecc9261c6a695cd3f1bd33710ef3ba1ca8fbb1425d20f305020e7c80904d6d6e8a4358bf926f920e6167c2c780d9f34db6abe06a51c8ff2571",
        "public_exponent": 65537,
        "type": "rsa public key"
      },
      "protocol_version": "https://opencoin.org/1.0",
      "renewal_service": [
        [10, "https://opencent.org"]
      ],
      "type": "cdd",
      "validation_service": [
        [10, "https://opencent.org"],
        [20, "https://opencent.com/validate"]
      ]
```

```
    },
    "signature":
"2bfa4a4c85a49f7c0493bef54cef40892cb23a613b3268d21689493f5a7825e93b22baa8cfc59f8dbf79d59
16348e586eb046f16a16cda7182e7e85d9746e7ff",
    "type": "cdd certificate"
  },
  "message_reference": 2,
  "status_code": 200,
  "status_description": "ok",
  "type": "response cddc"
}
```

Source

RequestMKCs Message

- **denominations**:
- **message_reference**:
- **mint_key_ids**:
- **type**:

Fields

Example

```
{
  "denominations": [1, 2, 5],
  "message_reference": 3,
  "mint_key_ids": [],
  "type": "request mint key certificates"
}
```

Source

ResponseMKCs Message

- **keys**:
- **message_reference**:
- **status_code**:
- **status_description**:
- **type**:

Fields

Example


```
{
  "keys": [
    {
      "mint_key": {
        "cdd_serial": 1,
        "coins_expiry_date": "2023-10-16T20:09:52.501723",
        "denomination": 1,
        "id": "bac419d0d8c235e31dae3d5419944e904169c12c3799087f4f9684176fd76d05",
        "issuer_id": "85c24031572f2e0a04a41a29eb74990f4651c7f0b4afc0b53cfa03bed30822e1",
        "public_mint_key": {
          "modulus":
"cdabcaff7484d35f43a7d9e2f51eabe23783c351be84e4ed39f955a012357ebdf56e71e1ac0c15994317b23
f45345acdd03bc02af9cd1dd72143ce33b26b4d27",
          "public_exponent": 65537,
          "type": "rsa public key"
        },
        "sign_coins_not_after": "2023-07-08T20:09:52.501723",
        "sign_coins_not_before": "2022-07-08T20:09:52.501723",
        "type": "mint key"
      },
      "signature":
"71b1c58d449634ca3cf719f82ba324573d7c32c7a18c6f25e7432d3efcc9fb4d661e5a9087f3ed5184d2e59
87784cb50ae8bb354479401869cc13ac2db8ae790",
      "type": "mint key certificate"
    },
    {
      "mint_key": {
        "cdd_serial": 1,
        "coins_expiry_date": "2023-10-16T20:09:52.501723",
        "denomination": 2,
        "id": "3f19f49247122834f1f46fa1602be004f7b1b159da04935e5957c6f509ccfea7",
        "issuer_id": "85c24031572f2e0a04a41a29eb74990f4651c7f0b4afc0b53cfa03bed30822e1",
        "public_mint_key": {
          "modulus":
"815def3cad88224295806821379fb11abd18a87b205aee79db4d65181e4a09a385526ca72e968e672b94135
f931a45ebdeeae29e4740372ffbd25b97cfa81c6d",
          "public_exponent": 65537,
          "type": "rsa public key"
        },
        "sign_coins_not_after": "2023-07-08T20:09:52.501723",
        "sign_coins_not_before": "2022-07-08T20:09:52.501723",
        "type": "mint key"
      },
      "signature":
"40bbaa15442c029d49e869364a4731abb04dc601505b84a8da804b22841dda07dd0f3fdc77febe58705bc73
a31cd8b9c9d791b17f1502ca6745c2d0a110f8c0b",
```

```
      "type": "mint key certificate"
    },
    {
      "mint_key": {
        "cdd_serial": 1,
        "coins_expiry_date": "2023-10-16T20:09:52.501723",
        "denomination": 5,
        "id": "6c6da7d032dff8b2489dca9398d1fa2d9ff11ed8bfd3e4144deb1ceaa7eb8818",
        "issuer_id": "85c24031572f2e0a04a41a29eb74990f4651c7f0b4afc0b53cfa03bed30822e1",
        "public_mint_key": {
          "modulus":
"9264f36d49bfb333856bad3a3769b7334be69830bdffaffe3cf792ce8e179b9dcebbe68c708fe88394ed3b1
4baadde2d58bde1ad6d09fc7e9e011c40cb3875f1",
          "public_exponent": 65537,
          "type": "rsa public key"
        },
        "sign_coins_not_after": "2023-07-08T20:09:52.501723",
        "sign_coins_not_before": "2022-07-08T20:09:52.501723",
        "type": "mint key"
      },
      "signature":
"3a489dced0e11d79598cea3107b5acfe07060e378ad0df679f7aeaef12f9cd75fb1fc9f58be83d032c0503a
00f46bd282ab870976a20a119d07025051f101899",
      "type": "mint key certificate"
    }
  ],
  "message_reference": 3,
  "status_code": 200,
  "status_description": "ok",
  "type": "response mint key certificates"
}
```


Source

RequestMint Message

Fields

- **blinds**:
- **message_reference**:
- **transaction_reference**:
- **type**:

Example


```
{
  "blinds": [
    {
      "blinded_payload_hash":
"c6db722a94f7c500878c13cb9025d6003e6611db066ea71dc1c34b2005933b6431314ae12719394679c7623
a69f637dad6ecce300c36988da9d6df3e8c384815",
      "mint_key_id": "bac419d0d8c235e31dae3d5419944e904169c12c3799087f4f9684176fd76d05",
      "reference": "a0",
      "type": "blinded payload hash"
    },
    {
      "blinded_payload_hash":
"39a10d283dd0443389f2c5cf4a83f281770079b0816a1b2e1a1fac2c53e3644b89306921d5ebc2de2d96077
f9125d375ffe280d3c3468a606db3f7b7f2bd6421",
      "mint_key_id": "3f19f49247122834f1f46fa1602be004f7b1b159da04935e5957c6f509ccfea7",
      "reference": "a1",
      "type": "blinded payload hash"
    },
    {
      "blinded_payload_hash":
"182def4dd07bfd73c403486b40c66b43e9df253b182115d108173f30d84041015776adee8f76623ba40e3be
0bb3aeb1daecb30ad2714ff2a7bfb7e3924128ddf",
      "mint_key_id": "6c6da7d032dff8b2489dca9398d1fa2d9ff11ed8bfd3e4144deb1ceaa7eb8818",
      "reference": "a2",
      "type": "blinded payload hash"
    }
  ],
  "message_reference": 4,
  "transaction_reference":
"f8b995ff44baa7df7c848ae67de72cfc55d241a7d393aa3392c6c3f0bd269551",
  "type": "request mint"
}
```


Source

ResponseMint Message

Fields

- **blind_signatures**:
- **message_reference**:
- **status_code**:
- **status_description**:
- **type**:

Example


```
{
  "blind_signatures": [
    {
      "blind_signature":
"68f1e187086ad2d6333cc6b798397dda2390db6abd3ea603557afa54ac65600f5048232a34922bbcb4c7584
f4dd4004500b45d79ef43f33673defab6dc109186",
      "reference": "a0",
      "type": "blind signature"
    },
    {
      "blind_signature":
"4338c5154ce6a878d31aca476bf1d79d633df976a534f8fbefb24930c75c9bd25e87c1d0a2bda1ada7915ec
7717a2aaba27dd7fd5ea58db900c4c5cf1c33cad5",
      "reference": "a1",
      "type": "blind signature"
    },
    {
      "blind_signature":
"7bf88ddc25e9c0c65813c25d694cb444777c54b385b06b59e3d783e9abe9be71ccebe820295cb2c0968619d
7ab83daf4f3ba180b787a8612c2c76913d0774125",
      "reference": "a2",
      "type": "blind signature"
    }
  ],
  "message_reference": 4,
  "status_code": 200,
  "status_description": "ok",
  "type": "response mint"
}
```


Source

CoinStack Message

Fields

- **coins**:
- **subject**:
- **type**:

Example


```
{
  "coins": [
    {
      "payload": {
        "cdd_location": "https://opencent.org",
        "denomination": 1,
        "issuer_id": "85c24031572f2e0a04a41a29eb74990f4651c7f0b4afc0b53cfa03bed30822e1",
        "mint_key_id":
"bac419d0d8c235e31dae3d5419944e904169c12c3799087f4f9684176fd76d05",
        "protocol_version": "https://opencoin.org/1.0",
        "serial": "93608b9fe7375a19df2ee880639ceb63cab925e111c1adcf564d96738be9cb75",
        "type": "payload"
      },
      "signature":
"1932c7352ba97a24cbdddade9747d93b22db145defbdd0441606772695f0ddb439dea17a9ce09f8ea0ef590
bea57293d40fef463372060b6eb4a50c4ab7194d0",
      "type": "coin"
    },
    {
      "payload": {
        "cdd_location": "https://opencent.org",
        "denomination": 2,
        "issuer_id": "85c24031572f2e0a04a41a29eb74990f4651c7f0b4afc0b53cfa03bed30822e1",
        "mint_key_id":
"3f19f49247122834f1f46fa1602be004f7b1b159da04935e5957c6f509ccfea7",
        "protocol_version": "https://opencoin.org/1.0",
        "serial": "ffe8691a219a543bc1f51f3dd697a5e17fe9e5a6dfbf0537a515766d19f216a5",
        "type": "payload"
      },
      "signature":
"202a2724f3007a43e6f082f381acb91fcfc908c6a3170c30d1e95e9d13dea03f9042d6cd0ff73a85ad8df0b
82ede07d1427dd0fc9c999cdf96656734e6999e00",
      "type": "coin"
    },
    {
      "payload": {
        "cdd_location": "https://opencent.org",
        "denomination": 5,
        "issuer_id": "85c24031572f2e0a04a41a29eb74990f4651c7f0b4afc0b53cfa03bed30822e1",
        "mint_key_id":
"6c6da7d032dff8b2489dca9398d1fa2d9ff11ed8bfd3e4144deb1ceaa7eb8818",
        "protocol_version": "https://opencoin.org/1.0",
        "serial": "9912a994b8a2372c23ed0c13f8f3f4bef1b3b05b4123ab40d5cc73858dbedc9b",
        "type": "payload"
      },
      "signature":
"6cc66cb2109120199c997608fab249d06b8b9ca0f1cb52289931d6ddf3ed7350b337922bac196abf2dbfcaa
6618d590fe175be31f83fc3264fbdb14e4b29e550",
      "type": "coin"
    }
```

```
  ],
  "subject": "a little gift",
  "type": "coinstack"
}
```

Source

RequestRenew Message

Coins that are received need to be swapped for new ones, in order to protect the receiver against double spending. Otherwise, the sender could keep a copy of the coins and try to use the coins again. Before doing so we need to ask: what coin sizes should be chosen for the coins to be minted?

What if we have not the right coin selection for an amount to pay? Imagine that the price is 5 opencent, but we just have coins in the sizes: 2, 2, 2.

One solution would be to require the recipient to give change. This would make the protocol more complicated, and would just shift the problem to the recipient. Another approach is to allow partial spending coins, but this again makes the protocol more complicated.[7]

The easy way out is to aim for a selection of coins that allows us to pay *any* amount below or equal to the sum of all coins. E.g. if we own the value of 6 opencent it would be advisable to have coin selection 2,2,1,1 in order to pay all possible amounts. This also prevents *amount tracing*, where an awkward price (13.37) asks for an awkward coin exchange at the issuer beforehand.

So, we need to look at the combined sum of coins received and coins already in possession, and needs to find the right coin selection to be able to make all possible future coin transfers. We will then know which coins to keep, and what blinds to make and paying for the minting using *all* the just received coins and using *some* existing coins.

Fields

- **blinds**:
- **coins**:
- **message_reference**:
- **transaction_reference**:
- **type**:

Example


```
{
  "blinds": [
    {
      "blinded_payload_hash":
"56436a49564ee8153bbaae034be9fbe6e314067f7b6de70c47219b36dd5103403614ef5fc93d7eb0879ce5c
8ccf1017a9d6f74e0c3a1c06d4b62a206565e4ef7",
      "mint_key_id": "3f19f49247122834f1f46fa1602be004f7b1b159da04935e5957c6f509ccfea7",
      "reference": "b0",
      "type": "blinded payload hash"
    },
    {
      "blinded_payload_hash":
"6f1e2f96c277b0529f8bf097b8b57d6a5e950db7e2f64d70b847fcb09cb84596bf1dceccf14078bfb59acf2
6bd71e6f85e4a588d859980796051dff937be58e9",
      "mint_key_id": "3f19f49247122834f1f46fa1602be004f7b1b159da04935e5957c6f509ccfea7",
      "reference": "b1",
      "type": "blinded payload hash"
    },
    {
      "blinded_payload_hash":
"f8fad20dda2dc1ed0da2727d6f81c94e5011ccbc1bcdd2e89905fdc69d78310ac510c5e52d5f7675b8a4259
905fc8fef703bab6d73c907e31b5657ea9090d81",
      "mint_key_id": "3f19f49247122834f1f46fa1602be004f7b1b159da04935e5957c6f509ccfea7",
      "reference": "b2",
      "type": "blinded payload hash"
    },
    {
      "blinded_payload_hash":
"7a7a001b8b79667eb79388b1e6e69e9618f2613f561b01282d625d4c20b3e09845d9b0ef2f5298d2ff6b628
0061cbdf16b71170ad2051a128f7360baa817940e",
      "mint_key_id": "3f19f49247122834f1f46fa1602be004f7b1b159da04935e5957c6f509ccfea7",
      "reference": "b3",
      "type": "blinded payload hash"
    }
  ],
  "coins": [
    {
      "payload": {
        "cdd_location": "https://opencent.org",
        "denomination": 1,
        "issuer_id": "85c24031572f2e0a04a41a29eb74990f4651c7f0b4afc0b53cfa03bed30822e1",
        "mint_key_id":
"bac419d0d8c235e31dae3d5419944e904169c12c3799087f4f9684176fd76d05",
        "protocol_version": "https://opencoin.org/1.0",
        "serial": "93608b9fe7375a19df2ee880639ceb63cab925e111c1adcf564d96738be9cb75",
        "type": "payload"
      },
      "signature":
"1932c7352ba97a24cbdddade9747d93b22db145defbdd0441606772695f0ddb439dea17a9ce09f8ea0ef590
bea57293d40fef463372060b6eb4a50c4ab7194d0",
```

```
      "type": "coin"
    },
    {
      "payload": {
        "cdd_location": "https://opencent.org",
        "denomination": 2,
        "issuer_id": "85c24031572f2e0a04a41a29eb74990f4651c7f0b4afc0b53cfa03bed30822e1",
        "mint_key_id":
"3f19f49247122834f1f46fa1602be004f7b1b159da04935e5957c6f509ccfea7",
        "protocol_version": "https://opencoin.org/1.0",
        "serial": "ffe8691a219a543bc1f51f3dd697a5e17fe9e5a6dfbf0537a515766d19f216a5",
        "type": "payload"
      },
      "signature":
"202a2724f3007a43e6f082f381acb91fcfc908c6a3170c30d1e95e9d13dea03f9042d6cd0ff73a85ad8df0b
82ede07d1427dd0fc9c999cdf96656734e6999e00",
      "type": "coin"
    },
    {
      "payload": {
        "cdd_location": "https://opencent.org",
        "denomination": 5,
        "issuer_id": "85c24031572f2e0a04a41a29eb74990f4651c7f0b4afc0b53cfa03bed30822e1",
        "mint_key_id":
"6c6da7d032dff8b2489dca9398d1fa2d9ff11ed8bfd3e4144deb1ceaa7eb8818",
        "protocol_version": "https://opencoin.org/1.0",
        "serial": "9912a994b8a2372c23ed0c13f8f3f4bef1b3b05b4123ab40d5cc73858dbedc9b",
        "type": "payload"
      },
      "signature":
"6cc66cb2109120199c997608fab249d06b8b9ca0f1cb52289931d6ddf3ed7350b337922bac196abf2dbfcaa
6618d590fe175be31f83fc3264fbdb14e4b29e550",
      "type": "coin"
    }
  ],
  "message_reference": 5,
  "transaction_reference": "e4f0b9b0d835ade72ee71d7d5f5bd6fd",
  "type": "request renew"
}
```


Source

ResponseDelay Message

Fields

- **message_reference**:
- **status_code**:
- **status_description**:
- **type**:

Example

```
{
  "message_reference": 5,
  "status_code": 300,
  "status_description": "ok",
  "type": "response delay"
}
```

Source

RequestResume Message

Fields

- **message_reference**:
- **transaction_reference**:
- **type**:

Example

```
{
  "message_reference": 6,
  "transaction_reference": "e4f0b9b0d835ade72ee71d7d5f5bd6fd",
  "type": "request resume"
}
```

Source

RequestRedeem Message

Fields

- **coins**:
- **message_reference**:
- **type**:

Example

```
{
  "coins": [
    {
      "payload": {
        "cdd_location": "https://opencent.org",
        "denomination": 2,
        "issuer_id": "85c24031572f2e0a04a41a29eb74990f4651c7f0b4afc0b53cfa03bed30822e1",
        "mint_key_id":
"3f19f49247122834f1f46fa1602be004f7b1b159da04935e5957c6f509ccfea7",
        "protocol_version": "https://opencoin.org/1.0",
        "serial": "efa42f53f82bae3b7a2cb5e9b9ee1549b0adab6f2aab5887a83dd31921ffd1fb",
        "type": "payload"
      },
      "signature":
"2f67f48830dd28f4692a5208e87e238f12b9458fa732d3f6de9c8e96b9471a4e2c1f9fb514df75f3907ae51
68118852c61c62f791ae3b41fd6eb08dafc9615e4",
      "type": "coin"
    },
    {
      "payload": {
        "cdd_location": "https://opencent.org",
        "denomination": 2,
        "issuer_id": "85c24031572f2e0a04a41a29eb74990f4651c7f0b4afc0b53cfa03bed30822e1",
        "mint_key_id":
"3f19f49247122834f1f46fa1602be004f7b1b159da04935e5957c6f509ccfea7",
        "protocol_version": "https://opencoin.org/1.0",
        "serial": "de8e5f5180fe3c029c9ad002e806246ce9d51a0d8f970f58ccae2e480be4bd31",
        "type": "payload"
      },
      "signature":
"3370d7e5ad2a2547ee3c5e2deae7d67230162db224553306804fa638a7cd54eaddf437625fd5a9930b6c1e9
36f7c70f8bc9df761f51600a3ac9a104f438ebf27",
      "type": "coin"
    }
  ],
  "message_reference": 7,
  "type": "request redeem"
}
```

Source

ResponseRedeem Message

Fields

- **message_reference**:
- **status_code**:
- **status_description**:
- **type**:

Example

```
{
  "message_reference": 7,
  "status_code": 200,
  "status_description": "ok",
  "type": "response redeem"
}
```

Source

Appendix

FAQ

Scope

Having said all of the above, we scope the protocol, and it's description in the following way:

Targeted at developers - developers should be enabled (and motivated) by the OpenCoin protocol to implement standard confirming software components and apps. However, we hope that this documentation is also understandable for the interested user (or founder, investor, auditor, etc.)

Just the protocol - we don't deliver any ready to use implementations. This allows us to fully focus on the protocol, and keeps a separation to actual implementations.

Easy to understand - we try to avoid complexity. This affects the protocol itself as well as its documentation. This means: if you, the reader, don't understand a sentence or a concept, please contact us. We will improve the description. Being easy to understand is one of the main goals of OpenCoin.

Only the core - lots of developments have happened since we started. Take the example of messengers like Signal, Telegram or WhatsApp. The have opened new ways to transport messages, and they take care of identifying the communication partner. This especially means that message transport and authentication stays out of scope.

History and old results

- Project history
- Project papers
- Crypto report
- Legal report
- Code bases (v1, sandbox, javascript implementation)

Artifacts

Details of the documents in the artifacts directory

JSON Schemata

Ideas

- use .oc file ending
- oc over html
- opencoin.org as a web interface demo provider, that can handle .oc files

Building blocks for writing

create CDDC
create MKCs
RequestCDDSerial
ResponseCDDSerial
RequestCDDC
ResponseCDDC
RequestMKCs
ResponseMKCs
prepare blinds
RequestMint
sign blinds
ResponseMint
unblind
transfer CoinStack, e.g. using Signal
tokenize sum
prepare blinds
RequestRenew
ResponseDelay
RequestResume
validate coins
RequestRedeem
ResponseRedeem

Header

Fields

Example

```
1
```

Source

Request

Fields

Example

```
1
```

Source

Response

Fields

Example

```
1
```

Source

1. David Chaum, “Blind signatures for untraceable payments”, Advances in Cryptology - Crypto ‘82, Springer-Verlag (1983), 199-203. ↩

2. Please check with your lawyer if this is a good idea. ↩

3. To keep the diagram simple we have left out Charlene who was mentioned above in "How does it work?". Bob does everything she does. ↩

4. "opencent" refers to the specific example currency. The generic term "opencoin" refers to any currency following the OpenCoin protocol (of which opencent is one). ↩

5. This is to minimize damage in case the mint keys get compromised. ↩

6. It might be that also some existing coins might be needed to be swapped to get a good coin selection. See Renew. ↩

7. GNU Taler experiments with this approach: in essence coins don't have serials but keys, which can sign a partial amount to be spent. This requires more smartness to avoid double spending, introducing new problems to be solved. ↩