

# OpenCoin

---

*a protocol for privacy preserving electronic cash payments*

Version: 0.4 - draft (July 2022)

Copyright (2022) J. Baach, N. Toedtmann

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

Jörg Baach

Nils Toedtmann

J. K. Muennich

M. Ryden

J. Suhr



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

For more information go to <https://opencoin.org>



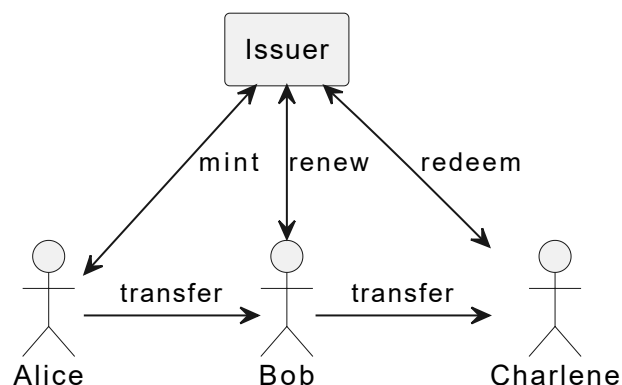
# Intro

---

We propose a protocol that allows cash-like payments in the electronic world. It is based on the invention by David Chaum<sup>1</sup>. 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 <sup>2</sup>.

## 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

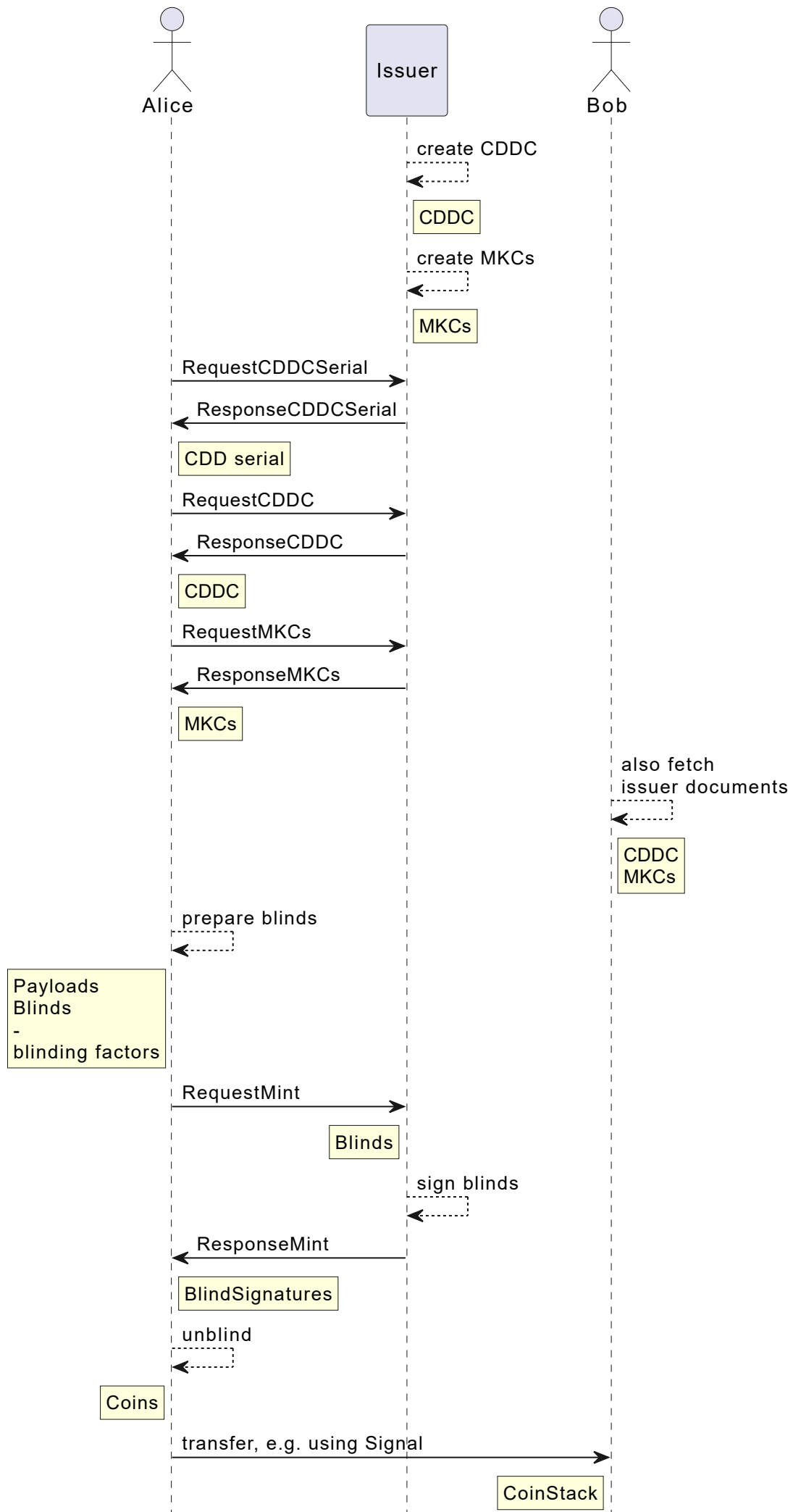
---

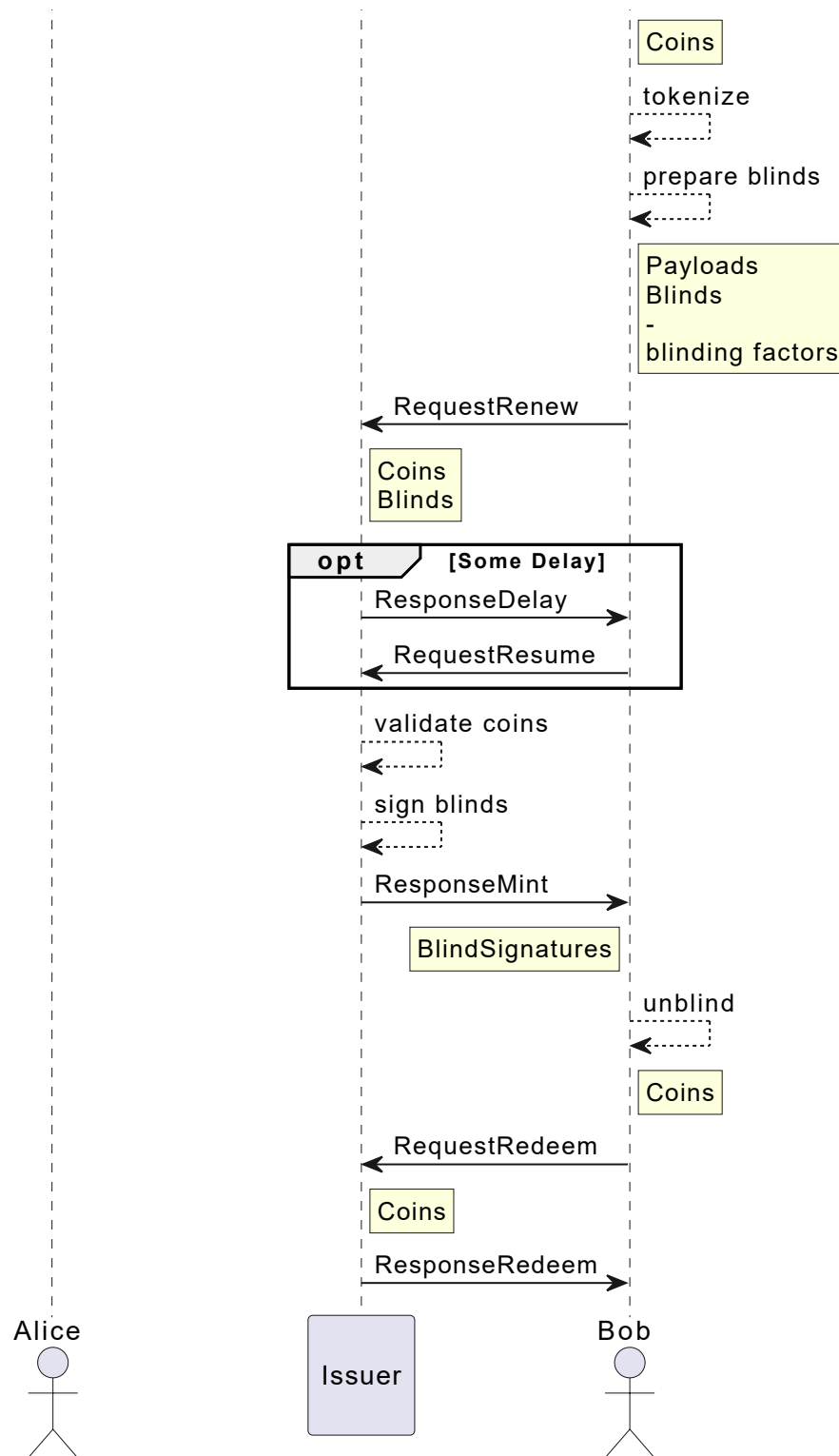
## Assumptions

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





## Description

This is a high level description of the actual steps, details follow in the chapters in [Details](#).<sup>3</sup>

## Participants

**Issuer** can mint, refresh and redeem coins. This entity will probably have 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.<sup>4</sup> 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".<sup>5</sup> 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.<sup>6</sup>

### 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. <sup>7</sup>

## **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

## Field Reference

### Field Types

This lists all the fields used in the protocol. All:

- String: A JSON string.
- Integer: A JSON integer.
- BigInt: A JSON string containing a large number represented as the hex representation of it.
- DateTime: A JSON string containing the ISO representation of a date.
- List: A JSON list that can contain all the possible field types mentioned here.
- URL: A JSON string containing the URL of a resource. Useful for round-robin, but sets a preference.  
The lower the higher the priority.
- WeightedURLList: A list of 2 element tuples [Int, URL].
- Schema / Object: A JSON object that conforms to the given schema.

All fields are mandatory, but can be empty in case of strings.

### Fields

- **additional\_info**: A field where the issuer can store additional information about the currency.

Type: String

Used in: [CDDC](#)

- **blind\_signature:** The signature on the blinded hash of a payload.

Type: BigInt

Used in: [ResponseMint Message](#)

- **blind\_signatures:** A list of BlindSignatures.

Type: List of [BlindSignatures](#)

Used in: [ResponseMint Message](#)

- **blinded\_payload\_hash:** The blinded hash of a payload.

Type: BigInt

Used in: [Blind](#)

- **blinds:** A List of Blinds

Type: List of [Blinds](#)

Used in: [RequestMint Message](#), [RequestRenew Message](#)

- **cdd:** Contains the Currency Description Document (CDD)

Type: [CDD](#)

Used in: [CDDC](#)

- **cdd\_expiry\_date:** When does the CDD expire?

The cdd should not be used or validated after this date.

Type: String

Used in: [CDDC](#)

- **cdd\_location:** Hint to download the CDD if not available anyway.

Useful for clients to “bootstrap” a yet unknown currency.

Type: URL

Used in: [CDDC](#), [Payload](#)

- **cdd\_serial:** The version of the CDD.

Should be increased by 1 on a new version.

Type: Int

Used in: [CDDC](#), [MKC](#), [RequestCDDC Message](#), [ResponseCDDSerial Message](#)

- **cdd\_signing\_date**: When was the CDD signed?

Type: DateTime

Used in: [CDDC](#)

- **cddc**: A full Currency Description Document Certificate.

Type: [CDDC](#)

Used in: [ResponseCDDC Message](#)

- **coins**: A list of coins.

Type: List of [Coins](#)

Used in: [CoinStack Message](#), [RequestRedeem Message](#), [RequestRenew Message](#)

- **coins\_expiry\_date**: Coins expire after this date.

Do not use coins after this date. the coins before this date.

Type: DateTime

Used in: [MKC](#)

- **currency\_divisor**: Used to express the value in units of 'currency name'.

Example: a divisor of 100 can be used express cent values for EUR or USD.

Type: Int

Used in: [CDDC](#)

- **currency\_name**: The name of the currency, e.g. Dollar.

Use the name of the 'full' unit, and not its fraction, e.g. 'dollar' instead of 'cent', and use the currency\_divisor to express possible fractions.

Type: String

Used in: [CDDC](#)

- **denomination**: The value of the coin(s).

Type: Int

Used in: [MKC](#), [Payload](#)

- **denominations:** The list of possible denominations.  
Should be chosen wisely and listed in increasing value.  
Type: List of Int  
Used in: [CDDC](#), [RequestMKCs Message](#)
- **id:** Identifier, a somewhat redundant hash of the [PublicKey](#)  
This is just a visual helper, and MUST not be relied on. Calculate the hash of the key in the client.  
Type: BigInt  
Used in: [CDD](#), [CDDC](#), [MintKey](#), [MKC](#)
- **info\_service:** A list of locations where more information about the currency can be found.  
This refers to human-readable information.  
Type: WeightedURLList  
Used in: [CDDC](#)
- **issuer\_cipher\_suite:** description  
Type: String  
Used in: [CDDC](#)
- **issuer\_id:** description  
Type: String  
Used in: [MKC](#), [Payload](#)
- **issuer\_public\_master\_key:** description  
Type: String  
Used in: [CDDC](#)
- **keys:** description  
Type: String  
Used in: [ResponseMKCs Message](#)
- **message\_reference:** description



Type: String

Used in: [RequestCDDSerial Message](#), [RequestCDDC Message](#), [RequestMint Message](#), [RequestMKCs Message](#), [RequestRedeem Message](#), [RequestRenew Message](#), [RequestResume Message](#), [ResponseCDDSerial Message](#), [ResponseCDDC Message](#), [ResponseDelay Message](#), [ResponseMint Message](#), [ResponseMKCs Message](#), [ResponseRedeem Message](#)

- **mint\_key:** description

Type: String

Used in: [MKC](#)

- **mint\_key\_id:** description

Type: String

Used in: [Blind](#), [Payload](#)

- **mint\_key\_ids:** description

Type: String

Used in: [RequestMKCs Message](#)

- **mint\_service:** description

Type: String

Used in: [CDDC](#)

- **modulus:** description

Type: String

Used in: [PublicKey](#)

- **payload:** description

Type: String

Used in: [Coin](#)

- **protocol\_version:** description

Type: String

Used in: [CDDC](#), [Payload](#)

- **public\_exponent:** description  
Type: String  
Used in: [PublicKey](#)
- **public\_mint\_key:** description  
Type: String  
Used in: [MKC](#)
- **redeem\_service:** description  
Type: String  
Used in: [CDDC](#)
- **reference:** description  
Type: String  
Used in: [ResponseMint Message](#), [Blind](#)
- **renew\_service:** description  
Type: String  
Used in: [CDDC](#)
- **serial:** description  
Type: String  
Used in: [Payload](#)
- **sign\_coins\_not\_after:** description  
Type: String  
Used in: [MKC](#)
- **sign\_coins\_not\_before:** description  
Type: String  
Used in: [MKC](#)
- **signature:** description  
Type: String  
Used in: [CDDC](#), [Coin](#), [MKC](#)

- **status\_code:** description

Type: String

Used in: [ResponseCDDSerial Message](#), [ResponseCDDC Message](#), [ResponseDelay Message](#), [ResponseMint Message](#), [ResponseMKCs Message](#), [ResponseRedeem Message](#)

- **status\_description:** description

Type: String

Used in: [ResponseCDDSerial Message](#), [ResponseCDDC Message](#), [ResponseDelay Message](#), [ResponseMint Message](#), [ResponseMKCs Message](#), [ResponseRedeem Message](#)

- **subject:** description

Type: String

Used in: [CoinStack Message](#)

- **transaction\_reference:** description

Type: String

Used in: [RequestMint Message](#), [RequestRenew Message](#), [RequestResume Message](#)

- **type:** description

Type: String

Used in: [ResponseMint Message](#), [Blind](#), [CDDC](#), [CDDC](#), [Coin](#), [CoinStack Message](#), [MKC](#), [MKC](#), [Payload](#), [RequestCDDSerial Message](#), [RequestCDDC Message](#), [RequestMint Message](#), [RequestMKCs Message](#), [RequestRedeem Message](#), [RequestRenew Message](#), [RequestResume Message](#), [ResponseCDDSerial Message](#), [ResponseCDDC Message](#), [ResponseDelay Message](#), [ResponseMint Message](#), [ResponseMKCs Message](#), [ResponseRedeem Message](#), [PublicKey](#)

## Schemata

Elements of messages, but never used standalone

# CDD

## Fields

- **additional\_info:**
- **cdd\_expiry\_date:**
- **cdd\_location:**
- **cdd\_serial:**
- **cdd\_signing\_date:**
- **currency\_divisor:**
- **currency\_name:**
- **denominations:**
- **id:**
- **info\_service:**
- **redeem\_service:**
- **issuer\_cipher\_suite:**
- **issuer\_public\_master\_key:**
- **protocol\_version:**
- **renew\_service:**
- **type:**
- **mint\_service:**

## Example

```
1  {
2    "additional_info": "",
3    "cdd_expiry_date": "2023-07-08T20:09:52.501723",
4    "cdd_location": "https://opencent.org",
5    "cdd_serial": 1,
6    "cdd_signing_date": "2022-07-08T20:09:52.501723",
7    "currency_divisor": 100,
8    "currency_name": "OpenCent",
9    "denominations": [1, 2, 5],
10   "id": "85c24031572f2e0a04a41a29eb74990f4651c7f0b4afc0b53cfa03bed30822e1",
11   "info_service": [
12     [10, "https://opencent.org"]
13   ],
14   "redeem_service": [
15     [10, "https://opencent.org"]
16   ],
17   "issuer_cipher_suite": "RSA-SHA512-CHAUM86",
18   "issuer_public_master_key": {
19     "modulus":
20     "8004826974ed9eccc9261c6a695cd3f1bd33710ef3ba1ca8fbb1425d20f305020e7c80904d6d6e8a4358bf9
21     26f920e6167c2c780d9f34db6abe06a51c8ff2571",
```

```

20     "public_exponent": 65537,
21     "type": "rsa public key"
22 },
23 "protocol_version": "https://opencoin.org/1.0",
24 "renew_service": [
25     [10, "https://opencent.org"]
26 ],
27 "type": "cdd",
28 "mint_service": [
29     [10, "https://opencent.org"],
30     [20, "https://opencent.com/validate"]
31 ]
32 }

```

## Source

## CDDC

## Fields

- **cdd:**
- **signature:**
- **type:**

## Example

```

1  {
2    "cdd": {
3      "additional_info": "",
4      "cdd_expiry_date": "2023-07-08T20:09:52.501723",
5      "cdd_location": "https://opencent.org",
6      "cdd_serial": 1,
7      "cdd_signing_date": "2022-07-08T20:09:52.501723",
8      "currency_divisor": 100,
9      "currency_name": "OpenCent",
10     "denominations": [1, 2, 5],
11     "id": "85c24031572f2e0a04a41a29eb74990f4651c7f0b4afc0b53cfa03bed30822e1",
12     "info_service": [
13       [10, "https://opencent.org"]
14     ],
15     "redeem_service": [
16       [10, "https://opencent.org"]
17     ],
18     "issuer_cipher_suite": "RSA-SHA512-CHAUM86",
19     "issuer_public_master_key": {

```

```

20     "modulus":
21     "8004826974ed9eccc9261c6a695cd3f1bd33710ef3ba1ca8fbb1425d20f305020e7c80904d6d6e8a4358bf9
22     26f920e6167c2c780d9f34db6abe06a51c8ff2571",
23     "public_exponent": 65537,
24     "type": "rsa public key"
25 },
26 "protocol_version": "https://opencoin.org/1.0",
27 "renew_service": [
28     [10, "https://opencent.org"]
29 ],
30 "type": "cdd",
31 "mint_service": [
32     [10, "https://opencent.org"],
33     [20, "https://opencent.com/validate"]
34 ]
35 },
36 "signature":
37 "2bfa4a4c85a49f7c0493bef54cef40892cb23a613b3268d21689493f5a7825e93b22baa8cfc59f8dbf79d59
38 16348e586eb046f16a16cda7182e7e85d9746e7ff",
39 "type": "cdd certificate"
40 }

```

[Source](#)

## PublicKey

### Fields

- **modulus:**
- **public\_exponent:**
- **type:**

### Example

```

1  {
2    "modulus":
3    "8004826974ed9eccc9261c6a695cd3f1bd33710ef3ba1ca8fbb1425d20f305020e7c80904d6d6e8a4358bf92
4    6f920e6167c2c780d9f34db6abe06a51c8ff2571",
5    "public_exponent": 65537,
6    "type": "rsa public key"
7  }

```

[Source](#)

See [MKC](#)

## MintKey

### Fields

- **cdd\_serial:**
- **coins\_expiry\_date:**
- **denomination:**
- **id:**
- **issuer\_id:**
- **public\_mint\_key:**
- **sign\_coins\_not\_after:**
- **sign\_coins\_not\_before:**
- **type:**

### Example

```
1  {
2    "cdd_serial": 1,
3    "coins_expiry_date": "2023-10-18T21:16:36.768709",
4    "denomination": 1,
5    "id": "8a33fcb2f71570e96e5d4c7d2dc2e8ced4d15cde9559c0f1429249d0d6d1c321",
6    "issuer_id": "2b751c58feedf6ac349e5437d557aaf78bd9040abff4def10f7bc2abe6703bd8",
7    "public_mint_key": {
8      "modulus":
9      "abd105d65e807d4160e44161b1c51e519fa16cdf91629e0c0e179f1f2c54b2593ba59add56ef8f04d302196
10     658358dc049887182f5331c39d7d4d028e01f4851",
11      "public_exponent": 65537,
12      "type": "rsa public key"
13    },
14    "sign_coins_not_after": "2023-07-10T21:16:36.768709",
15    "sign_coins_not_before": "2022-07-10T21:16:36.768709",
16    "type": "mint key"
17  }
```

[Source](#)

# MKC

*A Mint Key Certificate.*

## Fields

- **mint\_key:**
- **signature:**
- **type:**

## Example

```
1  {
2    "mint_key": {
3      "cdd_serial": 1,
4      "coins_expiry_date": "2023-10-16T20:09:52.501723",
5      "denomination": 1,
6      "id": "bac419d0d8c235e31dae3d5419944e904169c12c3799087f4f9684176fd76d05",
7      "issuer_id": "85c24031572f2e0a04a41a29eb74990f4651c7f0b4afc0b53cfa03bed30822e1",
8      "public_mint_key": {
9        "modulus":
10       "cdabcaff7484d35f43a7d9e2f51eabe23783c351be84e4ed39f955a012357ebdf56e71e1ac0c15994317b23
11       f45345acdd03bc02af9cd1dd72143ce33b26b4d27",
12       "public_exponent": 65537,
13       "type": "rsa public key"
14     },
15     "sign_coins_not_after": "2023-07-08T20:09:52.501723",
16     "sign_coins_not_before": "2022-07-08T20:09:52.501723",
17     "type": "mint key"
18   },
19   "signature":
20   "71b1c58d449634ca3cf719f82ba324573d7c32c7a18c6f25e7432d3efcc9fb4d661e5a9087f3ed5184d2e59
21   87784cb50ae8bb354479401869cc13ac2db8ae790",
22   "type": "mint key certificate"
23 }
```

## Source

## Payload



## Fields

- **cdd\_location:**
- **denomination:**
- **issuer\_id:**
- **mint\_key\_id:**
- **protocol\_version:**
- **serial:**
- **type:**

## Example

```
1 {  
2   "cdd_location": "https://opencent.org",  
3   "denomination": 1,  
4   "issuer_id": "85c24031572f2e0a04a41a29eb74990f4651c7f0b4afc0b53cfa03bed30822e1",  
5   "mint_key_id": "bac419d0d8c235e31dae3d5419944e904169c12c3799087f4f9684176fd76d05",  
6   "protocol_version": "https://opencoin.org/1.0",  
7   "serial": "93608b9fe7375a19df2ee880639ceb63cab925e111c1adcf564d96738be9cb75",  
8   "type": "payload"  
9 }
```

## Source

## Blind

## Fields

- **blinded\_payload\_hash:**
- **mint\_key\_id:**
- **reference:**
- **type:**

## Example

```

1 {
2   "blinded_payload_hash":
3     "c6db722a94f7c500878c13cb9025d6003e6611db066ea71dc1c34b2005933b6431314ae12719394679c7623a
4     69f637dad6ecce300c36988da9d6df3e8c384815",
5   "mint_key_id": "bac419d0d8c235e31dae3d5419944e904169c12c3799087f4f9684176fd76d05",
6   "reference": "a0",
7   "type": "blinded payload hash"
8 }

```

[Source](#)

## BlindSignature

### Fields

- **blind\_signature:**
- **reference:**
- **type:**

### Example

```

1 {
2   "blind_signature":
3     "68f1e187086ad2d6333cc6b798397dda2390db6abd3ea603557afa54ac65600f5048232a34922bbcb4c7584f
4     4dd4004500b45d79ef43f33673defab6dc109186",
5   "reference": "a0",
6   "type": "blind signature"
7 }

```

[Source](#)

## Coin

### Fields

coin

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

- **type:**

## Example

```
1  {
2    "payload": {
3      "cdd_location": "https://opencent.org",
4      "denomination": 1,
5      "issuer_id": "85c24031572f2e0a04a41a29eb74990f4651c7f0b4afc0b53cfa03bed30822e1",
6      "mint_key_id": "bac419d0d8c235e31dae3d5419944e904169c12c3799087f4f9684176fd76d05",
7      "protocol_version": "https://opencoin.org/1.0",
8      "serial": "93608b9fe7375a19df2ee880639ceb63cab925e111c1adcf564d96738be9cb75",
9      "type": "payload"
10   },
11   "signature":
12     "1932c7352ba97a24cbdddade9747d93b22db145defbdd0441606772695f0ddb439dea17a9ce09f8ea0ef590
13     bea57293d40fef463372060b6eb4a50c4ab7194d0",
14   "type": "coin"
15 }
```

## Source

## RequestCDDSerial Message

## Fields

- **message\_reference:**
- **type:**

## Example

```
1  {
2    "message_reference": 1,
3    "type": "request cdd serial"
4  }
```

## Source

## ResponseCDDSerial Message

- **cdd\_serial:**
- **message\_reference:**
- **status\_code:**
- **status\_description:**
- **type:**

### Fields

### Example

```
1 {  
2   "cdd_serial": 1,  
3   "message_reference": 1,  
4   "status_code": 200,  
5   "status_description": "ok",  
6   "type": "response cdd serial"  
7 }
```

### [Source](#)

## RequestCDDC Message

- **cdd\_serial:**
- **message\_reference:**
- **type:**

### Fields

### Example

```
1 {  
2   "cdd_serial": 1,  
3   "message_reference": 2,  
4   "type": "request cddc"  
5 }
```

### [Source](#)

# ResponseCDDC Message

## Fields

- **cddc:**
- **message\_reference:**
- **status\_code:**
- **status\_description:**
- **type:**

## Example

```
1  {
2    "cddc": {
3      "cdd": {
4        "additional_info": "",
5        "cdd_expiry_date": "2023-07-08T20:09:52.501723",
6        "cdd_location": "https://opencent.org",
7        "cdd_serial": 1,
8        "cdd_signing_date": "2022-07-08T20:09:52.501723",
9        "currency_divisor": 100,
10       "currency_name": "OpenCent",
11       "denominations": [1, 2, 5],
12       "id": "85c24031572f2e0a04a41a29eb74990f4651c7f0b4afc0b53cfa03bed30822e1",
13       "info_service": [
14         [10, "https://opencent.org"]
15       ],
16       "redeem_service": [
17         [10, "https://opencent.org"]
18       ],
19       "issuer_cipher_suite": "RSA-SHA512-CHAUM86",
20       "issuer_public_master_key": {
21         "modulus":
22         "8004826974ed9eccc9261c6a695cd3f1bd33710ef3ba1ca8fbb1425d20f305020e7c80904d6d6e8a4358bf9
23         26f920e6167c2c780d9f34db6abe06a51c8ff2571",
24         "public_exponent": 65537,
25         "type": "rsa public key"
26       },
27       "protocol_version": "https://opencoin.org/1.0",
28       "renew_service": [
29         [10, "https://opencent.org"]
30       ],
31       "type": "cdd",
32       "mint_service": [
33         [10, "https://opencent.org"],
34         [20, "https://opencent.com/validate"]
35       ]
36     }
37   }
```

```

34     },
35     "signature":
36     "2bfa4a4c85a49f7c0493bef54cef40892cb23a613b3268d21689493f5a7825e93b22baa8cfc59f8dbf79d59
16348e586eb046f16a16cda7182e7e85d9746e7ff",
36     "type": "cdd certificate"
37 },
38 "message_reference": 2,
39 "status_code": 200,
40 "status_description": "ok",
41 "type": "response cddc"
42 }

```

[Source](#)

## RequestMKCs Message

- **denominations:**
- **message\_reference:**
- **mint\_key\_ids:**
- **type:**

## Fields

## Example

```

1  {
2    "denominations": [1, 2, 5],
3    "message_reference": 3,
4    "mint_key_ids": [],
5    "type": "request mint key certificates"
6  }

```

[Source](#)

## ResponseMKCs Message

- **keys:**
- **message\_reference:**
- **status\_code:**
- **status\_description:**
- **type:**

# Fields

## Example

```
1  {
2    "keys": [
3      {
4        "mint_key": {
5          "cdd_serial": 1,
6          "coins_expiry_date": "2023-10-16T20:09:52.501723",
7          "denomination": 1,
8          "id": "bac419d0d8c235e31dae3d5419944e904169c12c3799087f4f9684176fd76d05",
9          "issuer_id": "85c24031572f2e0a04a41a29eb74990f4651c7f0b4afc0b53cfa03bed30822e1",
10         "public_mint_key": {
11           "modulus":
12             "cdabcaff7484d35f43a7d9e2f51eabe23783c351be84e4ed39f955a012357ebdf56e71e1ac0c15994317b23
13             f45345acdd03bc02af9cd1dd72143ce33b26b4d27",
14           "public_exponent": 65537,
15           "type": "rsa public key"
16         },
17         "sign_coins_not_after": "2023-07-08T20:09:52.501723",
18         "sign_coins_not_before": "2022-07-08T20:09:52.501723",
19         "type": "mint key"
20       },
21       "signature":
22         "71b1c58d449634ca3cf719f82ba324573d7c32c7a18c6f25e7432d3efcc9fb4d661e5a9087f3ed5184d2e59
23         87784cb50ae8bb354479401869cc13ac2db8ae790",
24       "type": "mint key certificate"
25     },
26     {
27       "mint_key": {
28         "cdd_serial": 1,
29         "coins_expiry_date": "2023-10-16T20:09:52.501723",
30         "denomination": 2,
31         "id": "3f19f49247122834f1f46fa1602be004f7b1b159da04935e5957c6f509ccfea7",
32         "issuer_id": "85c24031572f2e0a04a41a29eb74990f4651c7f0b4afc0b53cfa03bed30822e1",
33         "public_mint_key": {
34           "modulus":
35             "815def3cad88224295806821379fb11abd18a87b205aee79db4d65181e4a09a385526ca72e968e672b94135
36             f931a45ebdeae29e4740372ffbd25b97cfa81c6d",
37           "public_exponent": 65537,
38           "type": "rsa public key"
39         },
40         "sign_coins_not_after": "2023-07-08T20:09:52.501723",
41         "sign_coins_not_before": "2022-07-08T20:09:52.501723",
42         "type": "mint key"
43       },
44       "signature":
45         "40bbaa15442c029d49e869364a4731abb04dc601505b84a8da804b22841dda07dd0f3fdc77febe58705bc73
46         a31cd8b9c9d791b17f1502ca6745c2d0a110f8c0b",
```

```

39     "type": "mint key certificate"
40 },
41 {
42     "mint_key": {
43         "cdd_serial": 1,
44         "coins_expiry_date": "2023-10-16T20:09:52.501723",
45         "denomination": 5,
46         "id": "6c6da7d032dff8b2489dca9398d1fa2d9ff11ed8bfd3e4144deb1ceaa7eb8818",
47         "issuer_id": "85c24031572f2e0a04a41a29eb74990f4651c7f0b4afc0b53cfa03bed30822e1",
48         "public_mint_key": {
49             "modulus":
50 "9264f36d49bfb333856bad3a3769b7334be69830bdfaffe3cf792ce8e179b9dcebbe68c708fe88394ed3b1
51 4baadde2d58bde1ad6d09fc7e9e011c40cb3875f1",
52             "public_exponent": 65537,
53             "type": "rsa public key"
54         },
55         "sign_coins_not_after": "2023-07-08T20:09:52.501723",
56         "sign_coins_not_before": "2022-07-08T20:09:52.501723",
57         "type": "mint key"
58     },
59     "signature":
60 "3a489dced0e11d79598cea3107b5acfe07060e378ad0df679f7aeaef12f9cd75fb1fc9f58be83d032c0503a
61 00f46bd282ab870976a20a119d07025051f101899",
62     "type": "mint key certificate"
63 }
64 ],
65 "message_reference": 3,
66 "status_code": 200,
67 "status_description": "ok",
68 "type": "response mint key certificates"
69 }

```

[Source](#)

## RequestMint Message

### Fields

- **blinds:**
- **message\_reference:**
- **transaction\_reference:**
- **type:**



## Example

```
1  {
2    "blinds": [
3      {
4        "blinded_payload_hash":
5        "c6db722a94f7c500878c13cb9025d6003e6611db066ea71dc1c34b2005933b6431314ae12719394679c7623
6        a69f637dad6ecce300c36988da9d6df3e8c384815",
7        "mint_key_id": "bac419d0d8c235e31dae3d5419944e904169c12c3799087f4f9684176fd76d05",
8        "reference": "a0",
9        "type": "blinded payload hash"
10     },
11     {
12       "blinded_payload_hash":
13       "39a10d283dd0443389f2c5cf4a83f281770079b0816a1b2e1a1fac2c53e3644b89306921d5ebc2de2d96077
14       f9125d375ffe280d3c3468a606db3f7b7f2bd6421",
15       "mint_key_id": "3f19f49247122834f1f46fa1602be004f7b1b159da04935e5957c6f509ccfea7",
16       "reference": "a1",
17       "type": "blinded payload hash"
18     },
19     {
20       "blinded_payload_hash":
21       "182def4dd07bfd73c403486b40c66b43e9df253b182115d108173f30d84041015776adee8f76623ba40e3be
22       0bb3aeb1daecb30ad2714ff2a7bfb7e3924128ddf",
23       "mint_key_id": "6c6da7d032dff8b2489dca9398d1fa2d9ff11ed8bfd3e4144deb1ceaa7eb8818",
24       "reference": "a2",
25       "type": "blinded payload hash"
26     }
27   ],
28   "message_reference": 4,
29   "transaction_reference":
30   "f8b995ff44baa7df7c848ae67de72cfc55d241a7d393aa3392c6c3f0bd269551",
31   "type": "request mint"
32 }
```

## Source

## ResponseMint Message

### Fields

- **blind\_signatures:**
- **message\_reference:**
- **status\_code:**
- **status\_description:**
- **type:**

## Example

```
1  {
2    "blind_signatures": [
3      {
4        "blind_signature":
5        "68f1e187086ad2d6333cc6b798397dda2390db6abd3ea603557afa54ac65600f5048232a34922bbcb4c7584
6        f4dd4004500b45d79ef43f33673defab6dc109186",
7        "reference": "a0",
8        "type": "blind signature"
9      },
10     {
11       "blind_signature":
12       "4338c5154ce6a878d31aca476bf1d79d633df976a534f8fbefb24930c75c9bd25e87c1d0a2bda1ada7915ec
13       7717a2aaba27dd7fd5ea58db900c4c5cf1c33cad5",
14       "reference": "a1",
15       "type": "blind signature"
16     },
17     {
18       "blind_signature":
19       "7bf88ddc25e9c0c65813c25d694cb444777c54b385b06b59e3d783e9abe9be71ccebe820295cb2c0968619d
20       7ab83daf4f3ba180b787a8612c2c76913d0774125",
21       "reference": "a2",
22       "type": "blind signature"
23     }
24   ],
25   "message_reference": 4,
26   "status_code": 200,
27   "status_description": "ok",
28   "type": "response mint"
29 }
```

## Source

## CoinStack Message

### Fields

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

## Example

```
1  {
2    "coins": [
3      {
4        "payload": {
5          "cdd_location": "https://opencent.org",
6          "denomination": 1,
7          "issuer_id": "85c24031572f2e0a04a41a29eb74990f4651c7f0b4afc0b53cfa03bed30822e1",
8          "mint_key_id":
9            "bac419d0d8c235e31dae3d5419944e904169c12c3799087f4f9684176fd76d05",
10         "protocol_version": "https://opencoin.org/1.0",
11         "serial": "93608b9fe7375a19df2ee880639ceb63cab925e111c1adcf564d96738be9cb75",
12         "type": "payload"
13       },
14       "signature":
15         "1932c7352ba97a24cbdddade9747d93b22db145defbdd0441606772695f0ddb439dea17a9ce09f8ea0ef590
16         bea57293d40fef463372060b6eb4a50c4ab7194d0",
17       "type": "coin"
18     },
19     {
20       "payload": {
21         "cdd_location": "https://opencent.org",
22         "denomination": 2,
23         "issuer_id": "85c24031572f2e0a04a41a29eb74990f4651c7f0b4afc0b53cfa03bed30822e1",
24         "mint_key_id":
25           "3f19f49247122834f1f46fa1602be004f7b1b159da04935e5957c6f509ccfea7",
26         "protocol_version": "https://opencoin.org/1.0",
27         "serial": "ffe8691a219a543bc1f51f3dd697a5e17fe9e5a6dfbf0537a515766d19f216a5",
28         "type": "payload"
29       },
30       "signature":
31         "202a2724f3007a43e6f082f381acb91fcfc908c6a3170c30d1e95e9d13dea03f9042d6cd0ff73a85ad8df0b
32         82ede07d1427dd0fc9c999cdf96656734e6999e00",
33       "type": "coin"
34     },
35     {
36       "payload": {
37         "cdd_location": "https://opencent.org",
38         "denomination": 5,
39         "issuer_id": "85c24031572f2e0a04a41a29eb74990f4651c7f0b4afc0b53cfa03bed30822e1",
40         "mint_key_id":
41           "6c6da7d032dff8b2489dca9398d1fa2d9ff11ed8bfd3e4144deb1ceaa7eb8818",
42         "protocol_version": "https://opencoin.org/1.0",
43         "serial": "9912a994b8a2372c23ed0c13f8f3f4bef1b3b05b4123ab40d5cc73858dbedc9b",
44         "type": "payload"
45       },
46       "signature":
47         "6cc66cb2109120199c997608fab249d06b8b9ca0f1cb52289931d6ddf3ed7350b337922bac196abf2dbfcaa
48         6618d590fe175be31f83fc3264fbdb14e4b29e550",
49       "type": "coin"
50     }
51   ]
52 }
```

```
42 ],
43 "subject": "a little gift",
44 "type": "coinstack"
45 }
```

[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.<sup>8</sup>

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

```
1  {
2    "blinds": [
3      {
4        "blinded_payload_hash":
5        "56436a49564ee8153bbae034be9fbe6e314067f7b6de70c47219b36dd5103403614ef5fc93d7eb0879ce5c
6        8ccf1017a9d6f74e0c3a1c06d4b62a206565e4ef7",
7        "mint_key_id": "3f19f49247122834f1f46fa1602be004f7b1b159da04935e5957c6f509ccfea7",
8        "reference": "b0",
9        "type": "blinded payload hash"
10     },
11     {
12       "blinded_payload_hash":
13       "6f1e2f96c277b0529f8bf097b8b57d6a5e950db7e2f64d70b847fcb09cb84596bf1dceccf14078bfb59acf2
14       6bd71e6f85e4a588d859980796051dff937be58e9",
15       "mint_key_id": "3f19f49247122834f1f46fa1602be004f7b1b159da04935e5957c6f509ccfea7",
16       "reference": "b1",
17       "type": "blinded payload hash"
18     },
19     {
20       "blinded_payload_hash":
21       "f8fad20dda2dc1ed0da2727d6f81c94e5011ccbc1bcd2e89905fdc69d78310ac510c5e52d5f7675b8a4259
22       905fc8fef703bab6d73c907e31b5657ea9090d81",
23       "mint_key_id": "3f19f49247122834f1f46fa1602be004f7b1b159da04935e5957c6f509ccfea7",
24       "reference": "b2",
25       "type": "blinded payload hash"
26     },
27     {
28       "blinded_payload_hash":
29       "7a7a001b8b79667eb79388b1e6e69e9618f2613f561b01282d625d4c20b3e09845d9b0ef2f5298d2ff6b628
30       0061cbdf16b71170ad2051a128f7360baa817940e",
31       "mint_key_id": "3f19f49247122834f1f46fa1602be004f7b1b159da04935e5957c6f509ccfea7",
32       "reference": "b3",
33       "type": "blinded payload hash"
34     }
35   ],
36   "coins": [
37     {
38       "payload": {
39         "cdd_location": "https://opencent.org",
40         "denomination": 1,
41         "issuer_id": "85c24031572f2e0a04a41a29eb74990f4651c7f0b4afc0b53cfa03bed30822e1",
42         "mint_key_id":
43         "bac419d0d8c235e31dae3d5419944e904169c12c3799087f4f9684176fd76d05",
44         "protocol_version": "https://opencoin.org/1.0",
45         "serial": "93608b9fe7375a19df2ee880639ceb63cab925e111c1adcf564d96738be9cb75",
46         "type": "payload"
47       },
48       "signature":
49       "1932c7352ba97a24cbdddade9747d93b22db145defbdd0441606772695f0ddb439dea17a9ce09f8ea0ef590
50       bea57293d40fef463372060b6eb4a50c4ab7194d0",

```

```

40     "type": "coin"
41   },
42   {
43     "payload": {
44       "cdd_location": "https://opencent.org",
45       "denomination": 2,
46       "issuer_id": "85c24031572f2e0a04a41a29eb74990f4651c7f0b4afc0b53cfa03bed30822e1",
47       "mint_key_id":
"3f19f49247122834f1f46fa1602be004f7b1b159da04935e5957c6f509ccfea7",
48       "protocol_version": "https://opencoin.org/1.0",
49       "serial": "ffe8691a219a543bc1f51f3dd697a5e17fe9e5a6dfbf0537a515766d19f216a5",
50       "type": "payload"
51     },
52     "signature":
"202a2724f3007a43e6f082f381acb91fcfc908c6a3170c30d1e95e9d13dea03f9042d6cd0ff73a85ad8df0b
82ede07d1427dd0fc9c999cdf96656734e6999e00",
53     "type": "coin"
54   },
55   {
56     "payload": {
57       "cdd_location": "https://opencent.org",
58       "denomination": 5,
59       "issuer_id": "85c24031572f2e0a04a41a29eb74990f4651c7f0b4afc0b53cfa03bed30822e1",
60       "mint_key_id":
"6c6da7d032dff8b2489dca9398d1fa2d9ff11ed8bfd3e4144deb1ceaa7eb8818",
61       "protocol_version": "https://opencoin.org/1.0",
62       "serial": "9912a994b8a2372c23ed0c13f8f3f4bef1b3b05b4123ab40d5cc73858dbedc9b",
63       "type": "payload"
64     },
65     "signature":
"6cc66cb2109120199c997608fab249d06b8b9ca0f1cb52289931d6ddf3ed7350b337922bac196abf2dbfcaa
6618d590fe175be31f83fc3264fbdb14e4b29e550",
66     "type": "coin"
67   }
68 ],
69 "message_reference": 5,
70 "transaction_reference": "e4f0b9b0d835ade72ee71d7d5f5bd6fd",
71 "type": "request renew"
72 }

```

[Source](#)

## ResponseDelay Message

## Fields

- **message\_reference:**
- **status\_code:**
- **status\_description:**
- **type:**

## Example

```
1 {  
2   "message_reference": 5,  
3   "status_code": 300,  
4   "status_description": "ok",  
5   "type": "response delay"  
6 }
```

## Source

## RequestResume Message

## Fields

- **message\_reference:**
- **transaction\_reference:**
- **type:**

## Example

```
1 {  
2   "message_reference": 6,  
3   "transaction_reference": "e4f0b9b0d835ade72ee71d7d5f5bd6fd",  
4   "type": "request resume"  
5 }
```

## Source

# RequestRedeem Message

## Fields

- **coins:**
- **message\_reference:**
- **type:**

## Example

```
1  {
2    "coins": [
3      {
4        "payload": {
5          "cdd_location": "https://opencent.org",
6          "denomination": 2,
7          "issuer_id": "85c24031572f2e0a04a41a29eb74990f4651c7f0b4afc0b53cfa03bed30822e1",
8          "mint_key_id":
9            "3f19f49247122834f1f46fa1602be004f7b1b159da04935e5957c6f509ccfea7",
10         "protocol_version": "https://opencoin.org/1.0",
11         "serial": "efa42f53f82bae3b7a2cb5e9b9ee1549b0adab6f2aab5887a83dd31921ffd1fb",
12         "type": "payload"
13       },
14       "signature":
15         "2f67f48830dd28f4692a5208e87e238f12b9458fa732d3f6de9c8e96b9471a4e2c1f9fb514df75f3907ae51
16         68118852c61c62f791ae3b41fd6eb08dafc9615e4",
17       "type": "coin"
18     },
19     {
20       "payload": {
21         "cdd_location": "https://opencent.org",
22         "denomination": 2,
23         "issuer_id": "85c24031572f2e0a04a41a29eb74990f4651c7f0b4afc0b53cfa03bed30822e1",
24         "mint_key_id":
25           "3f19f49247122834f1f46fa1602be004f7b1b159da04935e5957c6f509ccfea7",
26         "protocol_version": "https://opencoin.org/1.0",
27         "serial": "de8e5f5180fe3c029c9ad002e806246ce9d51a0d8f970f58ccae2e480be4bd31",
28         "type": "payload"
29       },
30       "signature":
31         "3370d7e5ad2a2547ee3c5e2deae7d67230162db224553306804fa638a7cd54eaddf437625fd5a9930b6c1e9
32         36f7c70f8bc9df761f51600a3ac9a104f438ebf27",
33       "type": "coin"
34     }
35   ],
36   "message_reference": 7,
37   "type": "request redeem"
38 }
```



[Source](#)

## ResponseRedeem Message

### Fields

- **message\_reference:**
- **status\_code:**
- **status\_description:**
- **type:**

### Example

```
1 {  
2   "message_reference": 7,  
3   "status_code": 200,  
4   "status_description": "ok",  
5   "type": "response redeem"  
6 }
```

[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. They 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. It is easier to follow along with the above diagram open in a second window (or printout). ↩
  4. To keep the diagram simple we have left out Charlene who was mentioned above in "How does it work?". Bob does everything she does. ↩
  5. "opencent" refers to the specific example currency. The generic term "opencoin" refers to any currency following the OpenCoin protocol (of which opencent is one). ↩
  6. This is to minimize damage in case the mint keys get compromised. ↩
  7. It might be that also some existing coins might be needed to be swapped to get a good coin selection. See Renew. ↩
  8. 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. ↩