API Documentation for API connecting to IPFS

The API serves as the connection to the IPFS network. It communicates to the IPFS node, that is running on the same machine as the API to publish and retrieve data from the network.

Current API port: 35.178.146.101:80

The API has the following six endpoints:

1. /retrieveData **GET**
2. /publishPassport **POST**
3. /addEvent **POST**
4. /retrieveEvent **GET**
5. /addMutableProduct **POST**
6. /retrieveMutableLog **GET**
7. /getQrCode **GET**

Endpoint description

1. Endpoint: /retrieveData

Method: GET

Endpoint for fetching passport, retrieving a passport is done by sending one CID to the http endpoint.

```
{
    "CID":"k51qzi5uqu5dg756xkngqneb6bu6agewf1dtrvj7vmzt6x2k41jzys8du6lod1"
}
```

The return value received is the public data of the passport and the encrypted data.

```
{
        "ProductName" : "Iron",
        "MaterialId" : "asasd21",
        "OrderId" : "kk221",
        "Dimensions" : "XXYYZZ",
        "Plant" : "Oxelosund, Sweden",
        "Entrydate" : datetime,
        "Certificates": (bytes of files or attatchments or whatever),
        "data" : ENCRYPTED DATA 1
        "remanufacturing_events" : own_pubkey,
        "shipping"  : own_pubkey,
        "makes" : own_pubkey,
        "made_from"  : own_pubkey
}
```

The exact keys and values are not completely set in stone and could potentially differ for each passport depending on what is decided.

2. Endpoint: /publishPassport

Method: POST

Used to publish the main immutable part of the passport as well as, striped down products that does not have a complete passport.

It reads the full body of the http request:

{
"ProductName" : ENCRYPTED DATA 1
"MaterialId" : "asasd21",
"OrderId" : "kk221",
"Dimensions" : "XXYYZZ",
"Plant" : "Oxelosund, Sweden",
"Entrydate" : datetime,
"Certificates": (bytes of files or attatchments or whatever),
"Data" : "ENCRYPTED DATA 1",
"Type" : "simple/complete",
"ProductType" : "Steel/iron",
(optional) "Made_by": "[{'CID': 'xxx', 'ProductType': 'Steel','Datetime': ' datetime'}]"
}

If the "Type" is "simple" no keys will be generated and all data will be published to the barebone DPP and any data in the Made_by field will be ignored.

If it is "complete" four IPNS keys will be generated (reman, shipping, makes,made_by).
These keys get sent to the CA with a HTTP POST so that the private keys can be retrieved.
If the field Made_by exists, then that gets published to the IPFS network and the made_by key points to the created log.
Then the components get their Makes log updated.

New passport uploaded:

{
"MaterialId" : "asasd21",
"OrderId" : "kk221",
"Dimensions" : "XXYYZZ",
"Plant" : "Oxelosund, Sweden",
"Entrydate" : datetime,
"Certificates": (bytes of files or attatchments or whatever),
"data" : ENCRYPTED DATA 1,
"ProductType" : "Steel/iron",
"remanufacturing_events" : own_pubkey,
"shipping" : own_pubkey,
"makes" : own_pubkey,
"made_byu" : own_pubkey
}

This file is dynamic, so the keys and values are not rigid and only depends on the http request.
To the user it returns the CID for the published passport.

```
{
    "CID":"QmXJaXhgWF4Jcv2FVdAjJmL5PXSa1fCrH5zn8xV66f9kyb"

}
```

3. Endpoint: /addEvent

Method: POST

Used to add events to a passport, the event can be remanufacturing or shipping (could be other that follow the same structure if needed).

To add an event, it needs the following keys and values in the request:

```
{
        "Key" : "the public key for the event (reman,shipping)",
        "Eventtype" : "type of event such as ground_transport, repair",
        "Datetime" : "datetime",
        "Data" : "Encrypted event data"
}
```

When an event is added the corresponding private key to the given public key is needed, so for this, a check is done to see if the private key exists locally and if not then a request HTTP GET request is done to the CA.

If the call is successful, it will return that it works, otherwise you will be prompted with an exception.

4. Endpoint: /retrieveEvent

Method: GET

Used to fetch either the latest added event or all events, (the functionality to retrieve a specific event could be added if needed but is not currently supported).

The request needs the following keys:

```
{
    "Key" : "the public key for the event log (reman,shipping)",
    "Type" : "decides if the last or all events get retrieved"
}
```

To retrieve all events type should be “AllEvents”
To retrieve last event type should be “LastEvent”

If the type is “LastEvent” then the return is

```
{
    "Eventtype" : "type of event such as ground_transport, repair",
    "Datetime" : "datetime",
    "Data" : "Encrypted event data 2"
}
```

If the type is “AllEvents” then the return is one Json for each of the events

```
{
    "Key" : "the public key for the event (reman,shipping)",
    "Eventtype" : "type of event such as ground_transport, repair",
    "Datetime" : "datetime",
    "Data" : "Encrypted event data"
}
{
    "Key" : "the public key for the event (reman,shipping)",
    "Eventtype" : "type of event such as ground_transport, repair",
    "Datetime" : "datetime",
    "Data" : "Encrypted event data"
}
```

If needed this could be changed to

```
[
    {
        "Key" : "the public key for the event (reman,shipping)",
        "Eventtype" : "type of event such as ground_transport, repair",
        "Datetime" : "datetime",
        "Data" : "Encrypted event data 1"
    },
    {
        "Key" : "the public key for the event (reman,shipping)",
        "Eventtype" : "type of event such as ground_transport, repair",
        "Datetime" : "datetime",
        "Data" : "Encrypted event data 2"
    }
]
```

5. Endpoint: /addMutableProduct

Method: POST

Used for adding/changing a made by or makes product.

To add a products CID to either made or make the following keys and values are needed in the request:

```
{
        "Key" : "the public key (made,makes)",
        "CID" : "CID of the product to be added",
        "ProductType" : "The typ of product (bolt, wheel)",
        "Datetime" : "datetime",
}
```

This adds the CID to the made or make log that the public key point towards.

When a product is added the corresponding private key to the given public key is needed, so for this, a check is done to see if the private key exists locally and if not, then a request HTTP GET request is done to the CA.

To change an already existing product CID (such as a CID that points to a “barebone” DPP) one additional field is needed.

```
{
        "Key" : "the public key (made,makes)",
        "CID" : "CID of the product to be added",
        "ProductType" : "The typ of product (bolt, wheel)",
        "Datetime" : "datetime",
        "CIDToReplace" : "The CID that should be changed"
}
```

This update made or makes log that the public key point towards, removing the information of the CIDToReplace with the new CID and its information.

When a product is added the corresponding private key to the given public key is needed, so for this, a check is done to see if the private key exists locally and if not then a request HTTP GET request is done to the CA.

If the call is successful, it will return that it works, otherwise you will be prompted with an exception.

Different keys and values can be used if need be.

6. Endpoint: /retrieveMutableLog

Method: GET

Used to retrieve a list/log of all the CIDs stored on the public key.

The request only needs the public key

```
{
        "Key" : "the public key (made,makes,reman,shipping)",
}
```

It will return the specified public keys log and if there is nothing to be returned you will be prompted with an exception.

```
[
  {
    "CID": "QmXnxhPHzvPxRk3W1szK8Gygi6ZSioXXDKMU39fAnZRLmi",
    "ProductType": "Bolt",
    "Datetime": "14:35 2024-02-27"
  },
  {
    "CID": "zzz",
    "ProductType": "Wheel",
    "Datetime": "14:37 2024-02-27"
  }
]
```

If more information is wanted from a public key, you can make a recursive call to retrieve it using the /retrieveData endpoint.

7. Endpoint: /getQrCode

Method: GET

Used to generate a QR code from a CID.

The request only needs the CID for the product you want to receive the QR code for.

```
{
        "CID" : "CID for the product that you want to retrieve the QR code for”
}
```

The return of this endpoint is

```
{
    "Filename": "QmQo56nSyBuAAryx6tuRmJ44mHW92dzhwMJdNkW7Pq8vHV",
    "Content": “Base64 representation of the QR code”
}
```