



UBIRCH

Concept Paper Data-UPP

Version: 8 June 2023




Table of Contents

1	JWT	4
1.1	JWT for Producer of devices:.....	4
1.2	JWT for data collector:.....	4
1.3	JWT for data processor	4
2	Specification	5
3	Component connections	6
4	Certification Process	8
4.1	Sensor.....	8
4.2	Data Collector	8
5	Verification Process	11

This concept draft is based on Certificate Creation & Verification Diagrams.

1 JWT

For this whole procedure it is required to use JSON Web Tokens (JWT).

 The payload in the JWT allows the creation of specific tokens, with different scopes to:

- Anchor UPPs
- Verify UPPs
- create Things
- get Info of Things
- ...

Another important element is the group, which needs to be provided by the Ubirch consulting. The group dependency limits the access to device attributes and UPPs and is also used for the accounting.

There are three different Tokens, which are required for this scenario.

1.1 JWT for Producer of devices:

The JWT for the producer of devices enable the creation of things and the request of APIconfig data for each device.

1.2 JWT for data collector:

The JWT for the data collector enables the anchoring of UPPs for devices, which belong to the designated group. This Token can be requested from the Ubirch consulting.

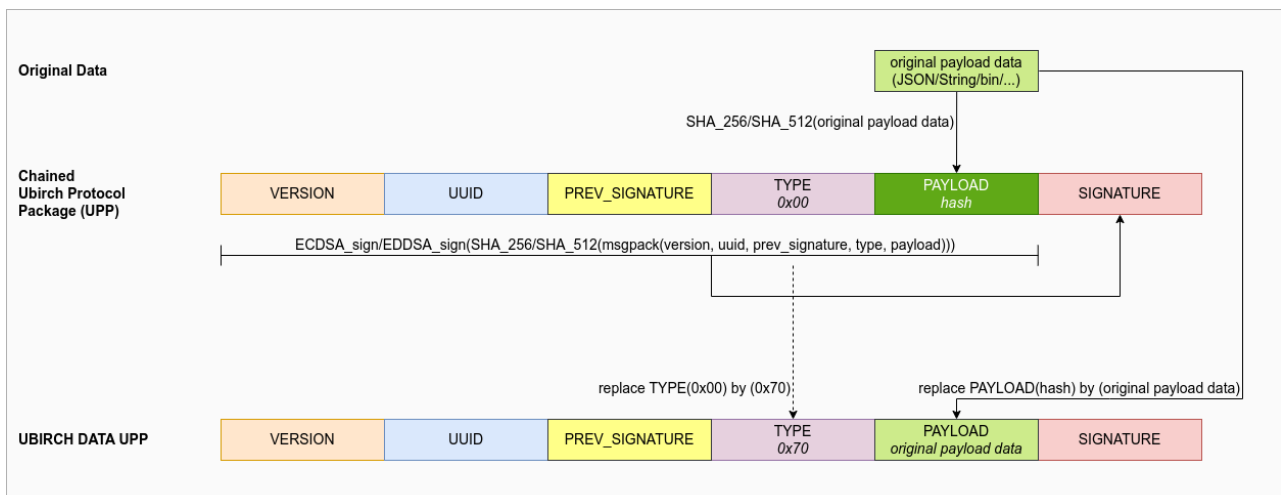
1.3 JWT for data processor

The JWT for the data processor enables the verification of data, via UPPs which were previously anchored. The verification token is only valid for a designated group and can be requested from Ubirch consulting.

2 Specification

To combine Data and UPP, based on a chained UPP, the fields Type and Payload are replaced:


- the **TYPE** field indicates the payload type, which will be replaced from **0x00** to **0x70**
- the **PAYLOAD** field is normally used for the hash of the data to anchor. This hash is also used for the later verification. In this case the **hash** will be replaced by the **original payload data**.



1 Specification

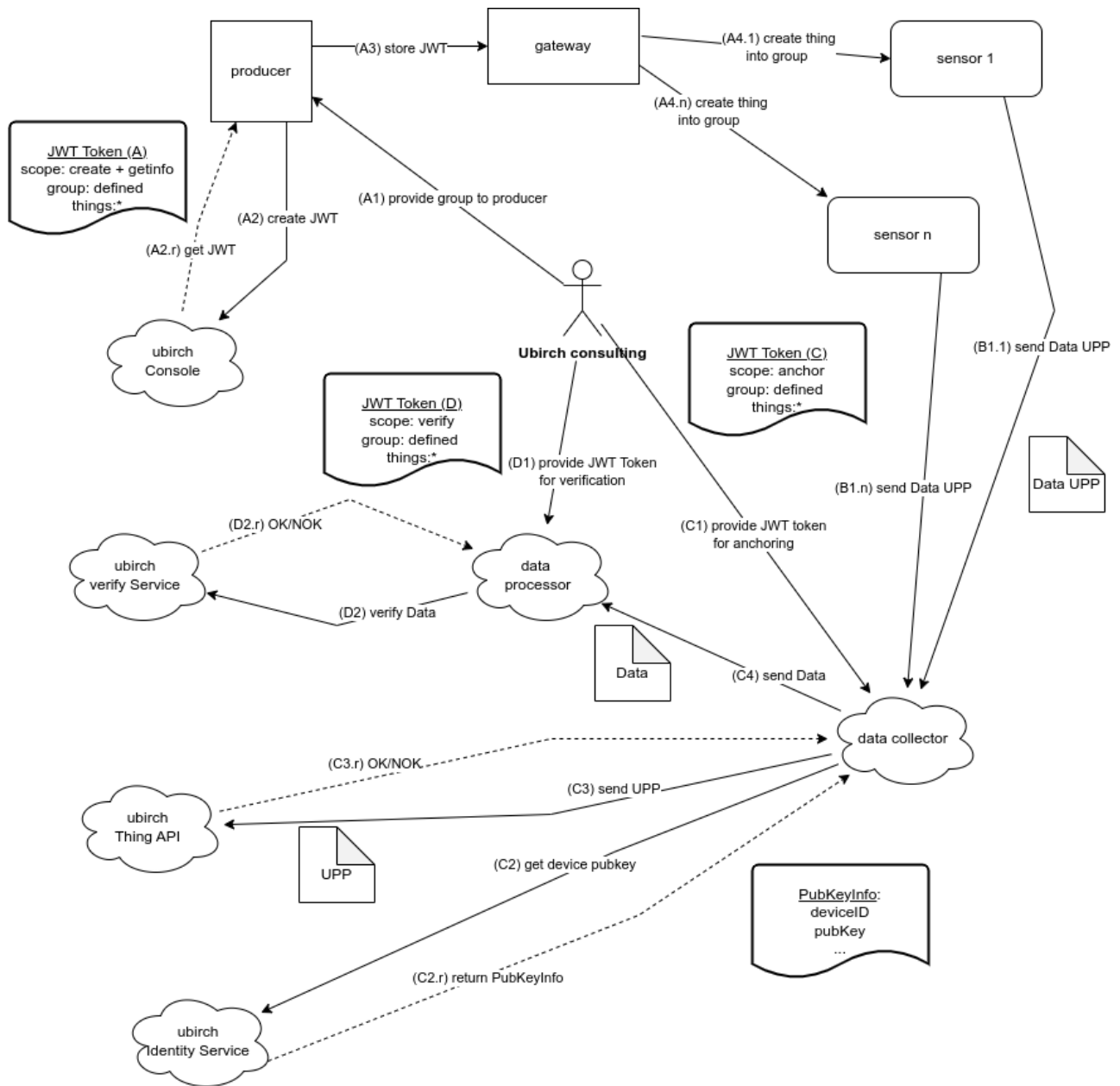
3 Component connections

The graph below shows the connections between all components, which are currently in the scope of the project.

 The end-user is currently not part of the scope, but will be integrated in the near future as well.

Besides the connections, there is also a segmentation and a chronological order in the graph. The segmentation is marked with letters and the chronology with numbers.

- (A) belongs to the creation of devices
- (B) belongs to the transmission of data-UPPs from the sensors
- (C) belongs to the separation of data and UPP and the transmission of those
- (D) belongs to the verification



2 Component connections

4 Certification Process

The Certification process is divided in two main parts. The first part takes place on the sensor, the second part at the data collector.

- i** Depending on the cryptographic algorithm of the device, the hashing algorithm is SHA256 for ECDSA and SHA512 for EDDSA(ED25519).

4.1 Sensor

The sensor starts by creating a regular chained UPP, where the `TYPE` is set to `0x00` and the `PAYLOAD` consists of the **hash** of the original payload data.

To finalize the UPP, the `SIGNATURE` is generated and added as the last field.

(Until now this is the standard procedure)

Now, to combine the data with the UPP and create a Data-UPP:

- **unpack the msgpack UPP**
 - this will lead to a list of fields: (`VERSION` , `UUID` , `PREV_SIGNATURE` , `TYPE` , `PAYLOAD hash` , `SIGNATURE`)
- copy `VERSION` , `UUID` and `PREV_SIGNATURE`
- replace the `TYPE` from **0x00** to **0x70**
- replace the `PAYLOAD hash` by the **original payload data**
- copy the `SIGNATURE`
- **pack the msgpack Data-UPP**

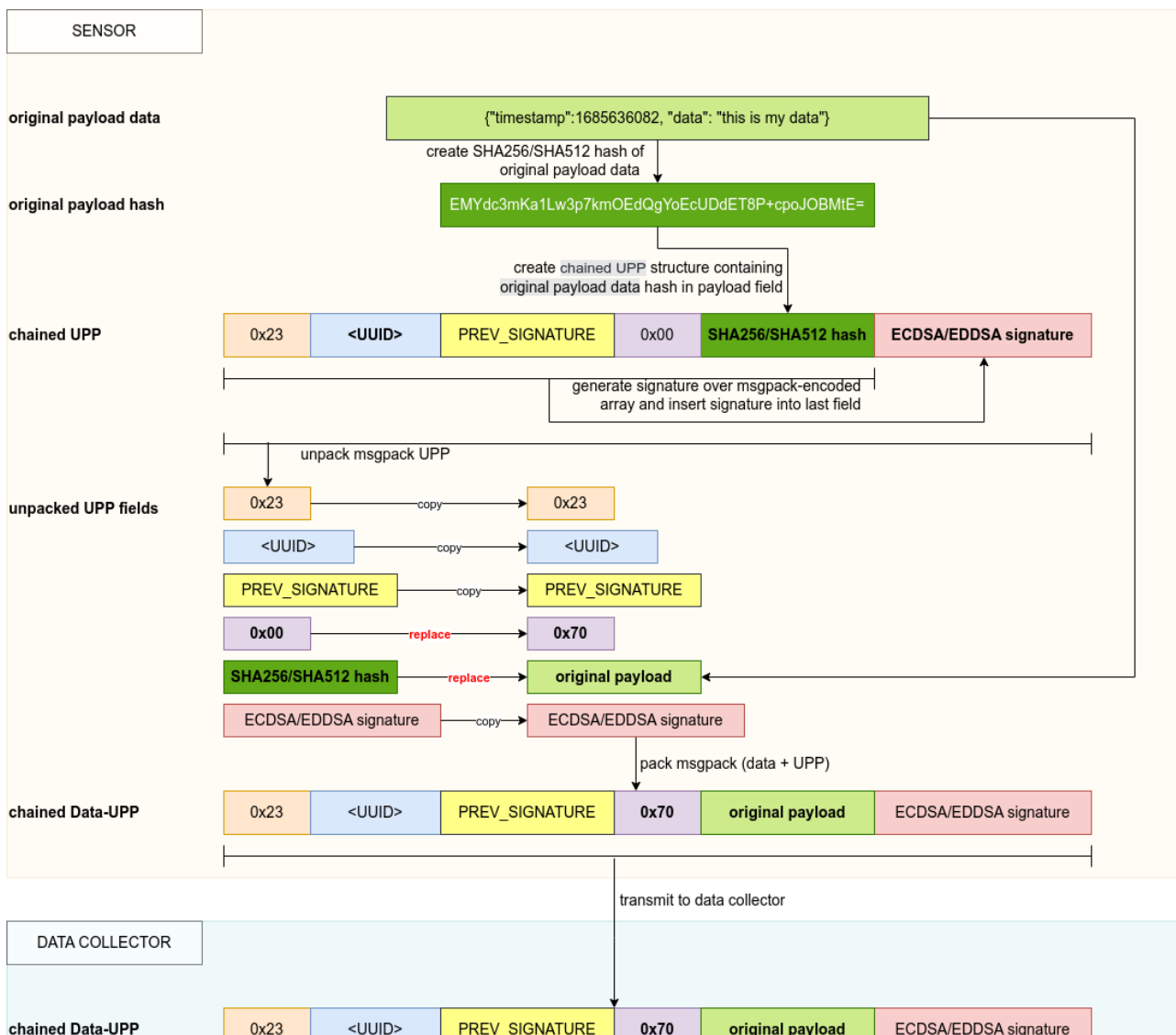
This Data-UPP can then be transmitted to the Data collector.

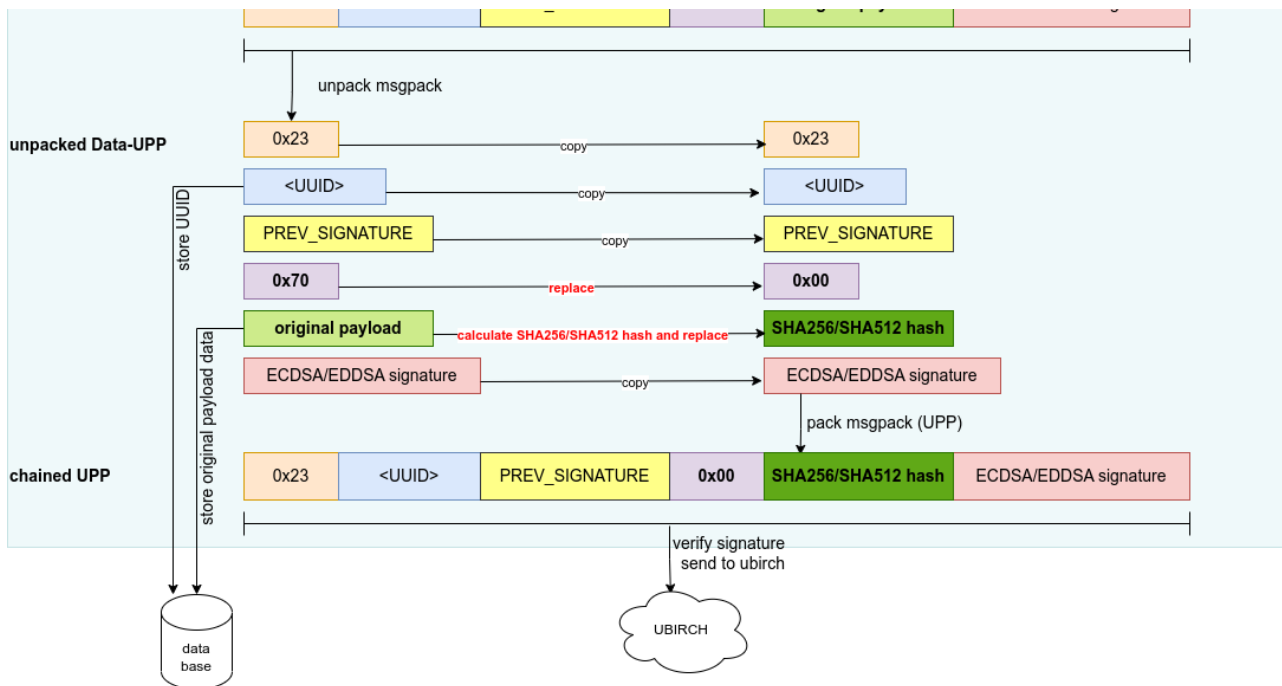
4.2 Data Collector

The data collector receives the Data-UPP and needs to make sure, that the data is valid and that it comes from the correct sensor.

- **unpack the msgpack Data-UPP**
 - this will lead to a list of fields: (`VERSION` , `UUID` , `PREV_SIGNATURE` , `TYPE` , `PAYLOAD original payload data` , `SIGNATURE`)
- check if the **public key** of the device (`UUID`) is already known
 - if not, then the **public key** can be requested from the identity Service [getV1CurrentHardwareId](#)
 - store the **UUID** and the corresponding **public key**

- copy `VERSION` , `UUID` and `PREV_SIGNATURE`
- replace the `TYPE` from `0x00` to `0x70`
- calculate the **hash** of the **original payload data** (based on the cryptographic algorithm of the device (`UUID`), this can be either **SHA256** or **SHA512**)
 - replace the **original payload data** by the calculated **hash**
- copy the `SIGNATURE`
- **pack the msgpack UPP**
- verify the UPP locally, via signature verification
 - if the `SIGNATURE` is valid, the `UUID` and the **original payload data** is correct and the UPP is ready for anchoring
- transmit the UPP to the ubirch niomon service
- store the **original payload data** in the database
- (transmit the **original payload data** to the data processor)



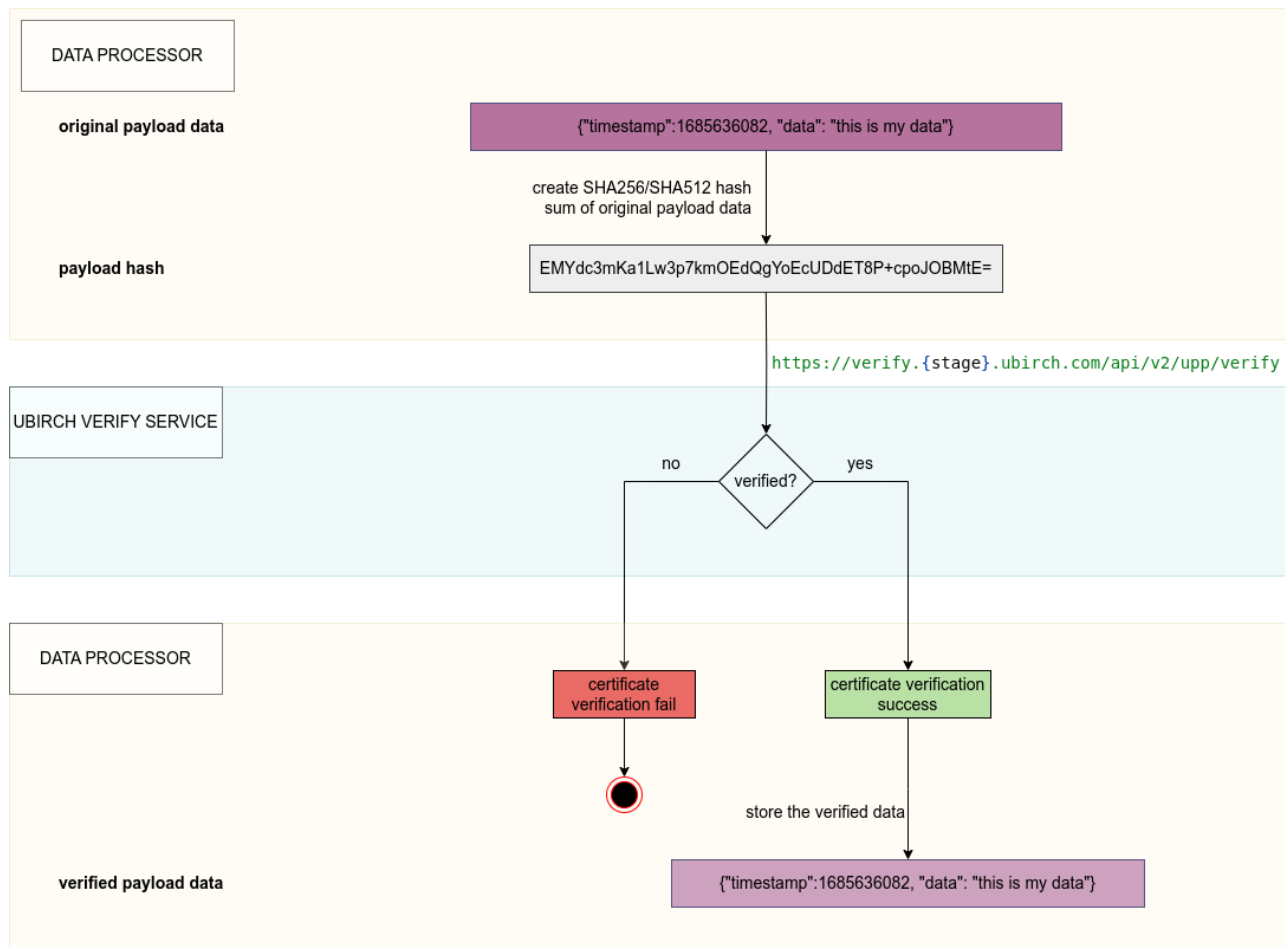


3 Certification Process

5 Verification Process

For the verification it is only necessary to calculate the hash of the original payload data and send it to the ubirch verification Service. If the hash was anchored before, the service will respond with success.

The verification can be performed by any instance holding a verification JWT and the original payload data.



4 Verification process