Quicktest DCC processing – data structures and communications

Further details can be found here: https://github.com/corona-warn-app/cwa-quicktest-onboarding/wiki/Anbindung-an-CWA-mit-Verwendung-von-DCCs

Mind the case sensitivity of all JSON attributes!

1 JSON / Test Registration / Assembled by Partner SW – QR(base64url) to CWA

EU Spec here

Defined by Partner
there will be another „testId“ in (3), do not mess it up

SHA-265 Hash of Test Registration except dgc
[dob]#[fn]#[ln]#[timestamp]#[testid]#[salt]

```
{
        "fn": "Erika",
        "ln": "Mustermann",
        "dob": "1990-12-23",
        "timestamp": 1622541600,
        "testid": "internalid01",
        "salt": "759F8FF3554F0E1BBF6EFF8DE298D9E9",
        "dgc": true,
        "hash": "a5df5308a159909e9f1e436e7fc9748d12bd8a251278b3c4a7f99b71da2bd9e8"
}
```

To CWA (QR or link)

2 JSON / Test Result / Assembled by Partner SW – POST to DCC-Service

id = hash
timestamp optional
Result 6 = negativ (no DCC for 7,8)
Defined by Partner, max 64 digits
Point of Care ID

```
{
        "testResults":
        [
                {
                "id": "a5df5308a159909e9f1e436e7fc9748d12bd8a251278b3c4a7f99b71da2bd9e8",
                "sc": 1622541600,
                "result": 6
                }
        ],
        "labId": "mylab01234"
}
```

hashed

To Result-Server (POST)
[Base-URL of Result-Server]/api/v1/quicktest/results

Hint: Test person has to open the negative test result and confirm to receive a DCC to generate a key pair and publish the public key.

3 JSON / DDC-Info from Backend for each recieved Test / Assembled by DCC-Service – GET on [Base-URL des DCC-Servers]/version/v1/publicKey/search/{labId}

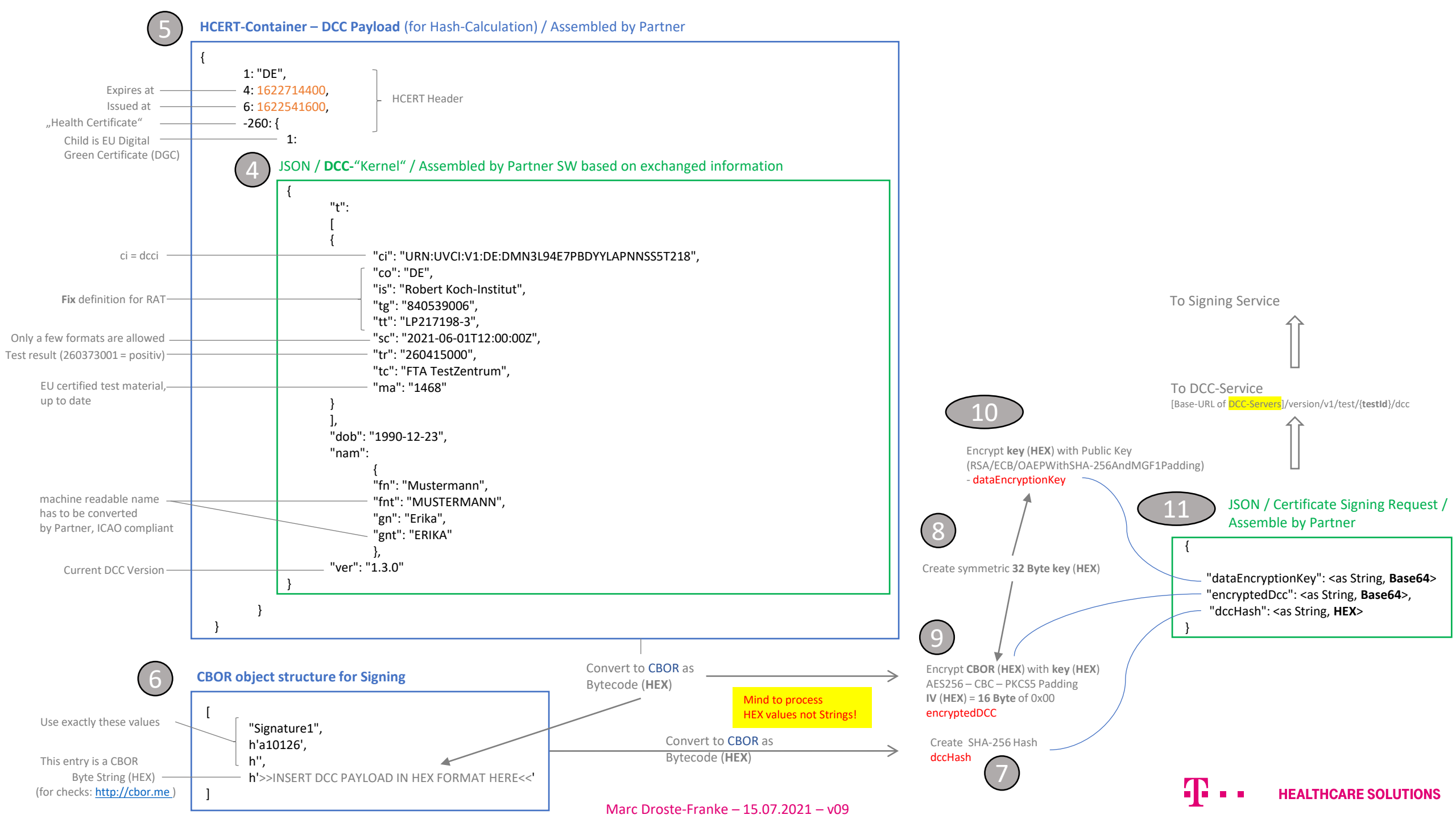
SHA-265 Hash of id, to identify test credentials
Has to be calculated by Partner

Uniqued Id of DCC that has to be assembled

pub key of key pair generated by CWA after DCC confirmation after receiving the test result
- CWA publishes pub key to DCC-Service
CWA keeps priv key
3072 bit – base64

```
[
        {
        "testId": "4fd1f6f41b510fc…",
        "dcci": "URN:UVCI:V1:DE:DMN3L94E7PBDYYLAPNNSS5T218",
        "publicKey": "MIIBojANBgkqhkiG9w0BAQEFAAOCAY8AMIIBigKC…"
        }
]
```

From DCC-Service (GET)
[Base-URL of DCC-Server]/version/v1/publicKey/search/{**labId**}


5
HCERT-Container – DCC Payload (for Hash-Calculation) / Assembled by Partner
{
1: "DE",
Expires at
4: 1622714400,
Issued at
6: 1622541600,
„Health Certificate“
-260: {
HCERT Header
Child is EU Digital Green Certificate (DGC)
1:
4
JSON / DCC-“Kernel“ / Assembled by Partner SW based on exchanged information
{
"t":
[
{
ci = dcci
"ci": "URN:UVCI:V1:DE:DMN3L94E7PBDYYLAPNNSS5T218",
"co": "DE",
"is": "Robert Koch-Institut",
Fix definition for RAT
"tg": "840539006",
"tt": "LP217198-3",
Only a few formats are allowed
"sc": "2021-06-01T12:00:00Z",
Test result (260373001 = positiv)
"tr": "260415000",
"tc": "FTA TestZentrum",
EU certified test material, up to date
"ma": "1468"
}
],
"dob": "1990-12-23",
"nam":
{
machine readable name has to be converted by Partner, ICAO compliant
"fn": "Mustermann",
"fnt": "MUSTERMANN",
"gn": "Erika",
"gnt": "ERIKA"
},
Current DCC Version
"ver": "1.3.0"
}
}
}
6
CBOR object structure for Signing
[
Use exactly these values
"Signature1",
h'a10126',
h'',
This entry is a CBOR Byte String (HEX) (for checks: http://cbor.me)
h'>>INSERT DCC PAYLOAD IN HEX FORMAT HERE<<'
]
Convert to CBOR as Bytecode (HEX)
Mind to process HEX values not Strings!
Convert to CBOR as Bytecode (HEX)
10
Encrypt key (HEX) with Public Key (RSA/ECB/OAEPWithSHA-256AndMGF1Padding)
- dataEncryptionKey
8
Create symmetric 32 Byte key (HEX)
9
Encrypt CBOR (HEX) with key (HEX)
AES256 – CBC – PKCS5 Padding
IV (HEX) = 16 Byte of 0x00
encryptedDCC
Create SHA-256 Hash
dccHash
7
To Signing Service
To DCC-Service
[Base-URL of DCC-Servers]/version/v1/test/{testId}/dcc
11
JSON / Certificate Signing Request / Assemble by Partner
{
"dataEncryptionKey": <as String, Base64>
"encryptedDcc": <as String, Base64>,
"dccHash": <as String, HEX>
}
Marc Droste-Franke – 15.07.2021 – v09
HEALTHCARE SOLUTIONS

12 JSON / Response from DCC Service with „raw“ base64 COSE Obj

```
{
        "partialDcc": " 0oRDoQEmoQRIDEsVUSvpFAFYLGdIZ2d…"
}
```

From DCC Service ⇦ From Signing Service ⇦

OPTIONAL - Create a Certificate QR-Code for RAT

13 CBOR object structure (COSE Array) / decoded Base64-CBOR from Response

```
[
    h'a10126',
    {4: h'0C4B15512BE91401'},
    h'674867674D596569795A534D33734A49564D4F454C4E797350493939551714F52526B4C794D7A35764A453D',
    h'[Signature from DCC server]'
]
```

Fixed

14 Placeholder!
Replace with DCC-Payload CBOR

15 Save as byte array and create QR, e.g. with CoseToQrCode(dccCose);