eHealth Network

DCC Anomaly Capture Process for COVID Certificate Data

Best current practice

Version: 1.01

15 September 2021

The eHealth Network is a voluntary network, set up under article 14 of Directive 2011/24/EU. It provides a platform of Member States' competent authorities dealing with eHealth. The Joint Action supporting the eHealth Network (JAseHN) provides scientific and technical support to the Network.

Adopted by consensus by the eHealth Network on 15 September 2021.

Executive Summary

At times it may be required to capture Digital COVID Certificate (DCC) data in the field for further investigation (e.g. if the DCC fails verification for no apparent reason; or if (large scale/sophisticated) digital fraud is suspected. Three levels of capture are defined; ranging from one that is fully anonymised (but still allows for verification of the digital seal) to an intermediate one (with just the UVCI, as per the recommendation) and a special level in which a one to one exact copy of the whole QR is made. Finally this document details some recommendations around the subsequent handling of that data.

Scope

This document sets out a process for the privacy-preserving handling of the in-vitro scans.

The GDPR and other legal considerations are out of scope; nor does this document make any assumption about which party should be responsible for the (initial) scan or the data. Any processing of personal data must comply with GDPR.

These considerations will be handled in a separate best-practice guideline.

Background

DCCs are rolled out in volume now; by many different countries (and sometimes even by different issuers within a country) -- each with their own issuer software. This software is generally written from scratch, by independent teams and with a highly diverse set of technologies. Likewise; most countries participating have written their own scanners; using a similarly diverse set of technologies.

Software is generally not perfect. And in this case - the standard evolved during the process.

So with many permutations of issuers, scanners (and near daily software updates), it is likely that we will increasingly need to investigate a ‘RED’ scan in the field, share these scans internationally or turn them into a format suitable for tests.

Critics of different backgrounds, be it government critics, corona critics or privacy critics publish “findings” on different media and in the press. Sometimes a problem is in fact found, sometimes it is not. These cases also have a need for investigation.

Citizens of Europe are eager to use the DCC, but sometimes encounter problems on issuance or verification. They call the Helpdesk and want to be helped. They often offer to release the QR so a solution can be found.

In all these cases (software, publications and help requests) data must be processed in another process than to grant access.

To make the bilateral (or through the eHealth Network) exchanges of this data easier - it is desirable for countries to use similar (good) practices. This makes it easier for all parties to understand what the situation is; and to share (debugging) tools.

However a DCC contains private, medical, data. Which can only be stored and exchanged with relatively high safeguard and in exceptional cases (in fact -the Regulation forbids routine capture). Experience during the first 4 weeks of operation has shown for most (technical) validations and ‘in vivo’ debugging the actual sensitive data is **not needed**. Instead - structure, checksums and digital signatures are more important to preserve.

Principles

The need to mask personal data - and in particular medical data - has long been a topic of interest in the field of medical informatics. Since with the EU Digital COVID Certificate (DCC) we are dealing with, albeit in a very small measure, medical data then we can turn to established standards for both pseudonymization and anonymization [1]:

In particular, the DICOM and HL7 international standards make provision for masking sensitive or personal data (DICOM de-identification [2], HL7 anonymization [3])

In addition, the IHE also provides a description of how to de-identify data [4] and there is an ISO standard available (ISO 25237 - [5]) which deals specifically with how to handle pseudonymization in the context of medical informatics.

Note that there are already free / open-source tools available for both DICOM de-identification (e.g. [6], [7]) and HL7 anonymization (e.g. [8]).

Best Current Practice

Since the EU DCC is neither a complete DICOM Metadata nor HL7 data record, then best current practice is to conform to ISO 25237:

"ISO 25237:2017 contains principles and requirements for privacy protection using pseudonymization services for the protection of personal health information. This document is applicable to organizations who wish to undertake pseudonymization processes for themselves or to organizations who make a claim of trustworthiness for operations engaged in pseudonymization services." [5]

For normal capture - all personal data should be masked from the record. This includes all fields in the "nam" field as well as the UVCI ('ci') field.

EU DCC fields:

- nam
- dob
- [v | t | r] /ci

To aid debugging - the masking should be done such that certain (structural) elements that may be relevant remain (both in the nam, dob and ci fields).

General field masking

In the decoded UTF8 sequence; each (unicode) glyph should be replaced according to the following schema for all fields (except the ci field) to a 7bit safe character from the ASCI 32..127 range:

Unicode 6 category	Sub-category
Letter (L) group	Ll (lowercase) by an 'x' LT (titlecase), Lu (Uppercase) by an 'X' Lm (modifier) by an M

	Lo (other) by an R
Mark (M) group	Mc by an ‘S’, Me, Mn by an ‘s’.
Number (N) group	Nd (digit) in the range U+0030-0039 to an ‘9’, all others to an 8, letter (Nl) by a 1, All others by a 2
Punctuation(P) group	‘-’ (U+002D) by a ‘-’; ‘.’ (U+002E) by a ‘.’, U+002C) by a ‘,’ remainder of Pd (Dash group): ‘=’. Pf/Ps/Pi/Pe (quotes/open) by a ‘Q’ All others by an ‘!’
Symbol (S) group (Sc, Sk, Sm, So)	By an ‘@’
Separator (Z) and Other (O) Group	Retain space: ‘ ‘(U+0020) by a ‘ ‘; all others Space (Zs) by an ‘_’. Line (Zl), Paragraph (Zp) by an N. All others by an ‘?’
Anything else	By the ‘Q’ (U+0071)

The reason for not mapping all (for example) numbers to a “9” is to distinguish between typical cases that need to be debugged. Such as the common substitution of a lowercase ‘L’(U+006C) for the digit ‘1’ (U+0031).

For this reason it is critical that no normalisation or any such changes are done to the UTF8 string prior to substitution; as to preserve things such as hidden backspaces, writing order, diacritical marks written as a Combining Character (e.g. U+0300–U+036F), hard spaces, etc.

DoB field substitution

The date of birth (‘dob’) should be reduced to just the year; the remainder of the string should be masked. The reason for preserving the year is to maintain the ability to apply special business logic (e.g. for children or young adults).

As the 0-9 digits are mapped to a ‘9’; and any alphanumeric character to an ‘X’ it becomes possible to recognise incomplete DoBs (e.g. those lacking the day or month, or using non standard values). This presents a small privacy risk (as this group is relatively small ~ 0.5% of the population).

UVCI field substitution

For the UVCI field - above defined masking should be applied after the country designator; but maintaining the length. This is to aid debugging of extreme/odd lengths (this is unlikely to be an indirect personal data issue - as countries generally issue UVCI’s of very similar and usually identical lengths).

The masking should be done with the following deviation from above table:

U+0041..005A	X
U+0061..007A	X
U+0030..0039	X

The reason for this more strict substitution is the relatively high level of entropy in some countries' UVCIs compared to their (much smaller, population sized) combinatorial space. Letting the position of digits/alphanumerics shimmer through would lead to unblinding risks.

Other residual risks

There is a potential residual risk around the time stamp in the COSE field which is not rounded or mapped out at L1. Future versions may need to mask this field to some number of equal length if actual implementation experience shows that this is an issue in practice.

Levels disclosed:

L1 (normal capture)	L2 (traceable capture)	L3 (full take)
n/a	n/a	QR code / photograph
n/a	n/a	SHA256 of the decoded QR Payload of decoded QR as base45*
CWT/COSE structure with the payload field replaced by a sequence 'X's (i.e. the byte 0x58); same length as the original binary/octet string.	As L1	SHA256 of the CWT/COSE structure CWT/COSE structure as base64
SHA256 of the actual payload sequence as a HEX sequence (as to still allow sig validation)	As L1	Base64 representation of the payload SHA256 of the decoded payload
n/a	SHA 256 of the QR	SHA256 of the QR

{
"ver": "1.3.0",
"nam": {
"fn": "Xxxxx-Xxxxx",
"fnt": "XX9XX<XXXX",
"gn": "Xxxxxxx Xxxxxx",
"gnt": "XXXXXXX<XXXXXX"
},
"dob": "1964-99-99",
"t": [
{
"tg": "840539006",
	{
	"ver": "1.3.0",
	"nam": {
	"fn": "Xxxxx-Xxxxx",
	"fnt": "XX9XX<XXXX",
	"gn": "Xxxxxxx Xxxxxx",
	"gnt": "XXXXXXX<XXXXXX"
	},
	"dob": "1964-99-99",
	"t": [
	{
	"tg": "840539006",
		{
		"ver": "1.3.0",
		"nam": {
		"fn": "Smith-Jones",
		"fnt": "SM1TH<JONES",
		"gn": "Charles Edward",
		"gnt": "CHARLES<EDWARD"
		},
		"dob": "1964-02-01",
		"t": [
		{
		"tg": "840539006",

"tt": "LP217198-3", "ma": "532", "sc": "2021-06-11T99:99:99+99", "tr": "260415000", "co": "NL", "is": "Amsterdam PHR", "ci": "URN:UVCI:01:NL:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX" }] }	"tt": "LP217198-3", "ma": "532", "sc": "2021-06-11T17:30:00+02", "tr": "260415000", "co": "NL", "is": "Amsterdam PHR", "ci": "URN:UVCI:01:NL:DADFCC47C7334E45A906DB12FD859FB2" }] }	"tt": "LP217198-3", "ma": "532", "sc": "2021-06-11T17:30:00+02", "tr": "260415000", "co": "UNHCR", "is": "Amsterdam PHR", "ci": "URN:UVCI:01:NL:DADFCC47C7334E45A906DB12FD859FB2" }] }

**) while the decoded payload of the QR is in principle an ASCII string; it may not be in case of decoding or ecc/cell errors. For this reason it must be treated as a binary octed string and encoded in base64 for transport safety.*

For L1 and higher - the data handled contains personal data (either just the UVCI in L2, or `everything' at L3). Handling and storage of these requires a set of appropriate organisational and technical measures. As a minimum the principle of four-eyes checking should be in place, with full, independent, auditable logs. In combination with encryption at rest. For L3 it is strongly advised to asymmetrically encrypt the record with controlled decryption key access (e.g. public/private key mechanism).

Note that there is a certain unblinding risk in L2 by revealing the payload SHA256 if the "nam" and "ci" fields are (relatively) short. As the permutation space of a short name and the missing DoB digits is small (10-15 characters with a lot of common names as you know the country, 3-4 digits for the DoB yields).

For L1 this is less of an issue - as the UCI should be both large and sufficiently securely random. As this is very unlikely for a real person (and likely the type of anomaly that one is trying to find) this is considered an acceptable, proportional risk.

Retention

Member States are advised to retain L1 and L2 records no longer than needed; and to consider to construct a fully anonymised replica if it is needed longer (e.g for a test case, a regression set, etc).

L3 captures should be retained no longer than needed and will need justification if kept for more than a month. The transmission of these records to a third party will need to comply with applicable procedures.

If records are shared between Member States then this will be subject to the agreed bilateral standardized bilateral assistance data sharing agreement.

All records must have a fixed (default is 10 days) and clear communicated retention time according to the national rights and anonymization level which can be confirmed by the user on the point of storing the dcc for analysis.

Encryption and Transmission

Appropriate (and often legally required) measures should be taken to ensure privacy and overall system integrity and security. Therefore the data records should be encrypted independent of the anonymization level to avoid data protection issues, if the verifier device is lost or any data is shared to the wrong destination.

To achieve this, the institution which verifies the DCC (e.g. authority) may provide a Public Key to encrypt the DCCs after anonymization and storing on the device. In this case, the X.509 certificate or public key must meet the SOGIS minimal levels as set at the date of release of the app (SOGIS Agreed Cryptographic Mechanisms, version 1.20, January 2020 [9]).

And can, at the time of writing this document, be a RSA-PSS or ECDSA 256 Key (P-256 Parameters) which is configurable in the verifier app .

The transmission of the DCCs should be done over secure channels. Whatsapp, Github, MMS or any other unsecured channel should not be used to share and/or collect data. It's recommended to delete every day the collected data from the device automatically to reduce the data collection on the device.

International exchange format (version 1.00)

For exchange purposes; it is suggested that member states package the data gathered in a ZIP file (ISO/IEC 21320-1 [10]) file that contains:

- A file called 'VERSION.txt' that contains just the 4 byte ASCII string '1.00' (semantic versioning will be used) of this international exchange format followed by a linefeed (LF, 0x0A).
- A README.txt that contains some human readable/oriented metadata on the capture process such as the application (version) used, the date, the entity responsible for the capture & contact details, issue/ticket numbers, and any other information deemed useful; such as errors/debug log information or circumstances.
- A 32 byte file 'payload-sha.bin' and a 65 byte human readable 'payload-sha.txt' that contains the SHA256 of the payload as a case insensitive HEX string terminated by a linefeed.
- A file 'QR.base64 that contains the COSE structure (with for L1/2 the payload replaced by an equal number of 0x58 bytes) as a base64 string [12].
- A file 'payload.json' that contains the decoded JSON *(with substitutions depending on the level applied).*
- **For L2 and L3:**
  - A 32 byte file 'QR-sha.bin' and a 65 byte, human readable, QR-sha.txt file that contains the SHA-256 of the QR as a case insensitive HEX string terminated by a linefeed.
- **For L3:**
  - A file 'QR.png' or 'QR.jpg' that contains the scanned QR **(L3 only)**
  - A file 'QR.txt' that contains the decoded string from the image 'as is'; so prior to HC1 stripping and base45 decoding **(L3 only)**
  - A 32 byte file cose-sha.bin' and a 65 byte, human readable, 'cose-sha.txt' that contains the SHA-256 of the payload as decoded as a case insensitive HEX string terminated by a linefeed. **(L3 only)**
  - A file `cose.base64' that contains the COSE binary ***(L3 only)***
  - A file `payload.base64' that contains the payload ***(L3 only)***

This format is subject (and likely) to change as implementation experience and feedback from the field is gathered. Optionally, ZIP files should then be digitally encrypted (and optionally signed) and packaged as a PKCS#7 CMS (RFC 3852 Cryptographic Message Syntax [10]) file (as per the suggestion in the previous section 'Transmission and Encryption)'.

International exchange process

This document does not yet propose an international process or how the relation between the controllers.

Version History

Version	Date	Changes
draft	202107XX	Various drafts for discussion; see also TSI design document
1.00	20210820	Proposed version; eHN WH meting 2021/08/27
1.01	20210910	Normative changes: **none.** Non normative/editorial changes: typos, fixed substitution, added CWT clarifications, clarify text around need to base64 protect decoded string. Technical scope added.

References

[1] https://www.johner-institute.com/articles/software-iec-62304/and-more/anonymization-and-pseudonymization/ , last accessed 2021-07-18

[2] http://dicom.nema.org/medical/dicom/current/output/html/part15.html#chapter_E , last accessed 2021-07-18

[3] http://hl7.org/fhir/secpriv-module.html#deId, last accessed 2021-07-18

[4] https://wiki.ihe.net/index.php/Healthcare_De-Identification_Handbook , last accessed 2021-07-18

[5] https://www.iso.org/standard/63553.html , last accessed 2021-07-18

[6] http://gdcm.sourceforge.net/html/gdcmanon.html , last accessed 2021-07-18

[7] https://github.com/aces/DICAT , last accessed 2021-07-18

[8] https://github.com/microsoft/FHIR-Tools-for-Anonymization , last accessed 2021-07-18

[9] https://www.sogis.eu/uk/supporting_doc_en.html (SOG IS minimal security standards), last accessed 2021-07-18

[10] https://www.iso.org/standard/60101.html (Zip Standard), last accessed 2021-07-18

[11] https://datatracker.ietf.org/doc/html/rfc3852 (Cyptographic Message Syntex (CMS) standard). last accessed 2021-07-18

[12] https://datatracker.ietf.org/doc/html/rfc4648 The Base16, Base32, and Base64 Data Encodings, Standard, last accessed 2021-07-18

Appendix A - Anonymization / Pseudonymization

Definition: Anonymization

“Anonymization is the changing of personal information so that the individual information about personal or material relationships can no longer be assigned to a certain person or determinable natural person or only with an unreasonably great expense of time, costs and effort.” *Source: FDPA*

Definition: Pseudonymization: “Pseudonymization” is the processing of personal data in such a way that the personal data or enlistment of additional information can no longer be traced to a specific person, if this additional information is to be stored separately and is subject to technical and organizational measures which ensure that the personal data cannot be assigned to an identified or identifiable natural person;” *Source: GDPR Article 4(5))*

Appendix B - Sample, none-normative, mapping

Example mapping in the python3 language; an array is used rather than a dictionary to retain order; and abort on the first match.

```
from pyslet.unicode5 import CharClass

# Use array rather than dict to preserve order.
mapping = [
        re.compile(str(CharClass(CharClass.ucd_category(u"Ll")))), 'x',
        re.compile(str(CharClass(CharClass.ucd_category(u"Lu")))), 'X',
        re.compile(str(CharClass(CharClass.ucd_category(u"Lt")))), 'T',
        re.compile(str(CharClass(CharClass.ucd_category(u"Lm")))), 'M',
        re.compile(str(CharClass(CharClass.ucd_category(u"Lo")))), 'R',

        re.compile(str(CharClass(CharClass.ucd_category(u"Mc")))), 'S',

        re.compile('[0-9]'):                                       '9',
        re.compile(str(CharClass(CharClass.ucd_category(u"Nd")))), '8',
        re.compile(str(CharClass(CharClass.ucd_category(u"Nl")))), '1',

        re.compile('-'),                                           '-',
        re.compile('\.'),                                          '.',
        re.compile(','),                                           ',',
        re.compile(str(CharClass(CharClass.ucd_category(u"Pd")))), '=',
        re.compile(str(CharClass(CharClass.ucd_category(u"Pf")))), '=',
        re.compile(str(CharClass(CharClass.ucd_category(u"Ps")))), '=',
        re.compile(str(CharClass(CharClass.ucd_category(u"Pi")))), '=',
        re.compile(str(CharClass(CharClass.ucd_category(u"Pe")))), '=',

        re.compile(' '),                                           ' ',
        re.compile(str(CharClass(CharClass.ucd_category(u"Zs")))), '_',
        re.compile(str(CharClass(CharClass.ucd_category(u"Zl")))), 'N',
        re.compile(str(CharClass(CharClass.ucd_category(u"Zp")))), 'N',
        re.compile(str(CharClass(CharClass.ucd_category(u"Z")))),  '?',

        re.compile(str(CharClass(CharClass.ucd_category(u"S")))),  '@',
        re.compile(str(CharClass(CharClass.ucd_category(u"L")))),  'R',
        re.compile(str(CharClass(CharClass.ucd_category(u"M")))),  's',
        re.compile(str(CharClass(CharClass.ucd_category(u"N")))),  '2',
        re.compile(str(CharClass(CharClass.ucd_category(u"P")))),  '!',

      re.compile(‘.’),                                           'Q',
]

urn_mapping = [
        re.compile(‘[A-Za-z0-9]’), 'X',
] + mapping
```