



# eHealth Network

Guidelines on

Technical Specifications

for EU Digital COVID Certificates

Volume 2

EU Digital COVID Certificate Gateway

Version 1.5

2022-06-15

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.

This document provides a detailed architecture of the EU Digital COVID Certificate Gateway (DCCG)—the central broker of crypto material needed to realize EU-wide validation of Digital COVID Certificates (DCC), comprising vaccination information, test results, and COVID-19 recovery details of EU citizens to facilitate safe travel, robust event management, and free movement across the EU. For more details on the DCC itself, please refer to

[https://ec.europa.eu/info/live-work-travel-eu/coronavirus-response/safe-covid-19-vaccines-europeans/eu-digital-covid-certificate\\_en](https://ec.europa.eu/info/live-work-travel-eu/coronavirus-response/safe-covid-19-vaccines-europeans/eu-digital-covid-certificate_en).

Adopted by the eHealth Network on 15.06.2022.

# TABLE OF CONTENTS

- TABLE OF CONTENTS ..... 3**
- TABLE OF TABLES ..... 5**
- TABLE OF FIGURES..... 6**
- 1 Introduction ..... 7**
  - 1.1 Context ..... 7**
  - 1.2 Scope of Document ..... 7**
- 2 Architecture Overview ..... 8**
  - 2.1 Approach..... 9**
  - 2.2 Assumptions..... 9**
  - 2.3 Verification through the backend server ..... 9**
- 3 Communication ..... 10**
  - 3.1 Triangle of Trust ..... 10**
  - 3.2 Distribution of Verification Information ..... 11**
  - 3.3 Device-to-Device Communication ..... 12**
  - 3.4 Backend-to-Backend Communication..... 12**
  - 3.5 Trust Establishment..... 13**
- 4 Data Structures ..... 14**
  - 4.1 Data Types ..... 14**
    - 4.1.1 JSON Schemas ..... 14
    - 4.1.2 Static Data Content ..... 14
    - 4.1.3 CMS Signed Data ..... 14
  - 4.2 Data Storage ..... 14**
    - 4.2.1 Database Requirements ..... 14
    - 4.2.2 Database ..... 15
    - 4.2.3 Data Format ..... 15
- 5 Building Block Schema ..... 17**
- 6 Interfaces ..... 18**
  - 6.1 Overview..... 18**
  - 6.2 MIME Type ..... 19**
  - 6.3 Upload Interface ..... 19**
    - 6.3.1 Overview ..... 19
    - 6.3.2 Parameters..... 20
    - 6.3.3 Responses..... 20
    - 6.3.4 Transmission Protocol ..... 21
  - 6.4 Signer Delete Interface ..... 21**

- 6.4.1 Overview ..... 21
- 6.4.2 Parameters ..... 21
- 6.4.3 Responses ..... 22
- 6.4.4 Transmission Protocol ..... 23
- 6.5 Trust List Interface ..... 23**
- 6.5.1 Overview ..... 23
- 6.5.2 Parameters ..... 24
- 6.5.3 Responses ..... 24
- 6.6 DSC Delta Download Extension ..... 25**
- 6.6.1 Introduction ..... 25
- 6.6.2 Assumptions ..... 25
- 6.6.3 Scope ..... 26
- 6.6.4 Solution ..... 26
- 6.6.5 Parameters ..... 27
- 6.6.6 Side Conditions/Assumptions ..... 28
- 6.6.7 Use Scenarios ..... 28
- 6.6.8 Recommendation for Verifier/Wallet App ..... 28
- 6.6.9 Reference Implementation Scope ..... 29
- 7 DSC Publication ..... 29**
- 7.1 Assumptions and constraints ..... 29**
- 7.2 Architecture ..... 29**
- 7.2.1 Solution ..... 29
- 7.2.2 Publishing Format ..... 30
- 7.2.3 Publication Process ..... 31
- 8 Technology Choice ..... 32**
- 9 Implementation Roadmap ..... 33**
- 10 Glossary ..... 34**
- APPENDIX ..... 36**

## TABLE OF TABLES

Table 1: Database Requirements ..... 15

Table 2: Trusted Party Table ..... 15

Table 3: Certificate Types ..... 15

Table 4: Signer Information Table ..... 16

Table 5: Audit Table..... 16

Table 6: Signer Information Table ..... 16

Table 8: Technology Choice..... 27

Table 9: Roadmap..... 28

## TABLE OF FIGURES

Figure 1: DCCG Overview ..... 8

Figure 2: Autonomous National Backends ..... 8

Figure 3: Triangle of Trust ..... 10

Figure 4: Trust Anchoring ..... 11

Figure 5: Distribution of Signing and Validation Information ..... 11

Figure 6: Distribution of Verification Information ..... 12

Figure 7: Indirect Backend-to-Backend Communication ..... 12

Figure 8: Trusted Data Exchange (Country 1 to Country 2) ..... 13

Figure 9: Onboarding Process ..... 13

Figure 14: Building Block Schema ..... 17

Figure 15: API Overview ..... 18

Figure 16: Interface Definition Overview ..... 19

Figure 17: Upload Interface..... 20

Figure 18: Upload Parameters ..... 20

Figure 20: Upload Responses ..... 21

Figure 21: Upload Transmission Process..... 22

Figure 22: Delete Interface..... 22

Figure 23: Delete Parameters ..... 23

Figure 24: Delete Response..... 23

Figure 25: Delete Flow..... 24

Figure 26: Trust List Interface..... 24

Figure 27: Trust List Interface..... 25

Figure 28: Trust List Responses ..... 26

## 1 Introduction

This document complements normative technical specifications adopted and published as Commission Implementing Decision (EU) 2021/1073 (with any amendments, such as Commission Implementing Decision (EU) 2021/2014). The document should be read together with the legal acts.

### 1.1 Context

The context for this architecture is the vision of the EU Trust framework to establish a secure data exchange for public key material, validation information and value sets between member states.

### 1.2 Scope of Document

The scope of the document is to specify the EU Digital COVID Certificate Gateway (DCCG) to support the EU trust framework concerning interoperability<sup>1</sup>. This consists of:

- Verification information (crypto material, references to trusted parties, etc.),
- Validation information (rules/patterns to differentiate valid and invalid certificates)

---

<sup>1</sup> [https://ec.europa.eu/health/sites/health/files/ehealth/docs/vaccination-proof\\_interoperability-guidelines\\_en.pdf](https://ec.europa.eu/health/sites/health/files/ehealth/docs/vaccination-proof_interoperability-guidelines_en.pdf)

## 2 Architecture Overview

As said before, DCCG is used to share validation and verification information across all national backend servers.

The following figure gives an overview of DCCG as specified in this document:

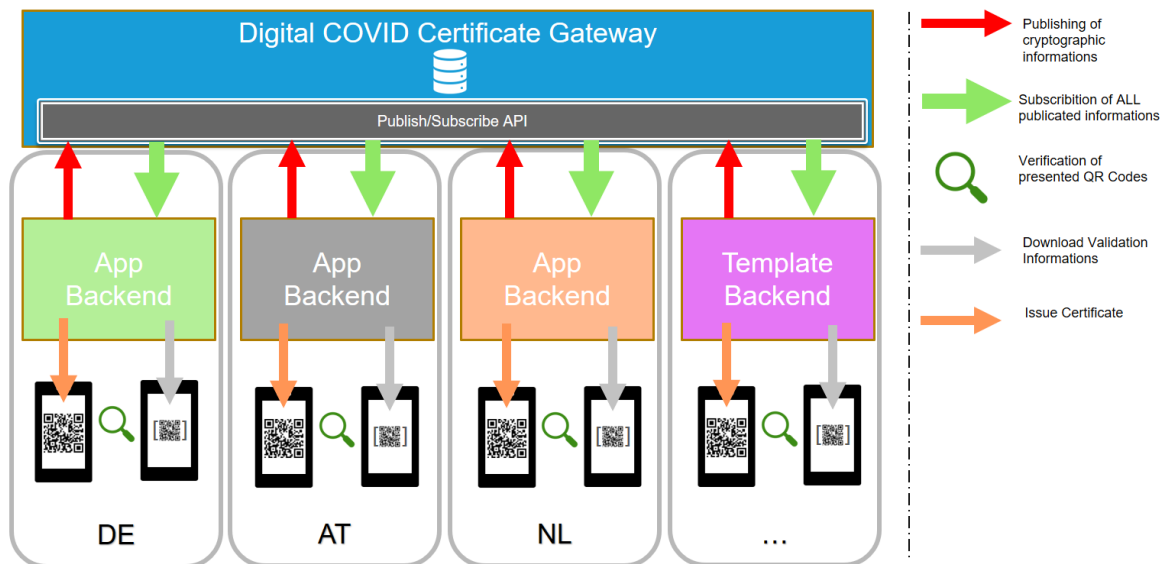


Figure 1: DCCG Overview

By using DCCG, backend-to-backend integration is facilitated, and countries can onboard incrementally, while the national backends retain flexibility and can control data processing of their users.

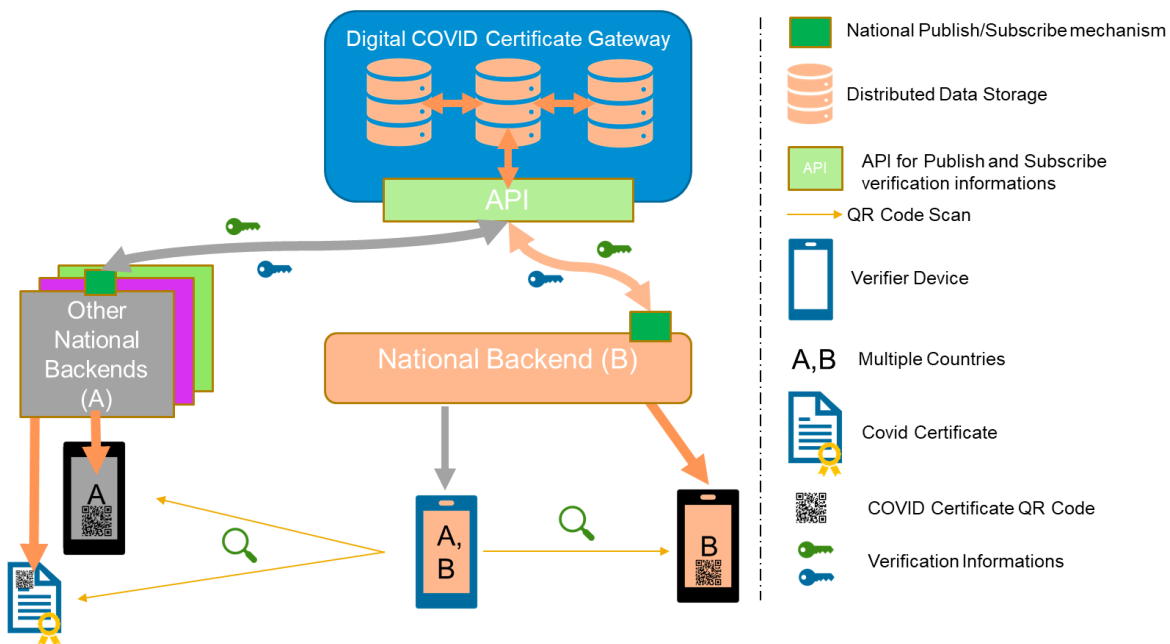


Figure 2: Autonomous National Backends

Each national backend is free to distribute the keys via any preferred technology to support the national verification devices in the best way. If the DCC is in a correctly formatted 2D code, each verifier



device can verify each code from other countries, if the verifier is connected to the backend or if it has downloaded and stored the necessary public keys beforehand.

## 2.1 Approach

The approach of this architecture is, as described before, to exchange different kinds of information to support the validation of the vaccination status, test result, or recovery status of a citizen (based on a test). For this purpose, the EU Trust Framework introduces a standardized signed CBOR data structure which is represented in a 2D Code. To validate this data structure in each country represented by different EU citizens, cryptographic public keys must be shared across the EU. The DCCG is designed to distribute such information easily to all member states and act as a Trust Anchor. This guarantees a reliable governance structure and trust into the public keys.

The entire architecture is inspired by the European Federation Gateway Service (EFGS), which is used to share GAEN diagnosis keys around the member states.

## 2.2 Assumptions

These are the main **assumptions** underlying this architecture:

- 1) For effective interoperability, each certificate must contain a string which uniquely identifies the issuing authority.
- 2) Each authority must pass an onboarding process, which identifies the authority uniquely by adding authentication, signing, issuer domain, CSCA and corresponding client certificates, as well contact persons and 24x7 support information's. Security of the onboarding process is vital for the security of the whole Digital COVID Certificate framework. The CSCA certificates should be at least 2 years valid.
- 3) Each authority (there can be several per member state) provide endpoints for revocation and optionally callbacks, which are secured by mutual authentication. The required certificates are provided during onboarding and are whitelisted by the authorities by downloading the DCCG Trust Lists. The DCCG Trust List contains the public keys of all trusted parties, i.e., authorities who successfully completed the DCCG onboarding process.
- 4) Each 2D code relies on Key identifier, which allows each member state to check a scanned DCC against the crypto material list received from the DCCG (issuer maps to COVID Certificates) The crypto material should be uniquely identifiable. (described in the 2D Code Specification, with the first 8 bytes of a SHA256 cert hash)
- 5) All Digital COVID Certificates and related crypto material are managed by the issuing authority.

## 2.3 Verification through the backend server

The national Backend servers of the Member State will provide access to the Public Keys according to national specifications, in particular to the entities that will perform verification of DCC using the relevant Public Keys (Trust lists). To this end the national backend servers will be kept up-to-date.

This will enable signature verification services and/or access to the public keys needed to perform verifications.

This feature will be implemented on the national backends level for instance through a secure API interface and SDKs. Such feature would be of particular relevance for airlines.

### 3 Communication

#### 3.1 Triangle of Trust

The triangle of trust describes the relationship between a holder, issuer, and verifier of credentials in a distributed system. This setup increases data privacy, because the holder is the only actor who has a direct connection to the verifier and the issuer, i.e., the issuer doesn't know where the holder presents their credentials and the issuer doesn't know who trusts him.

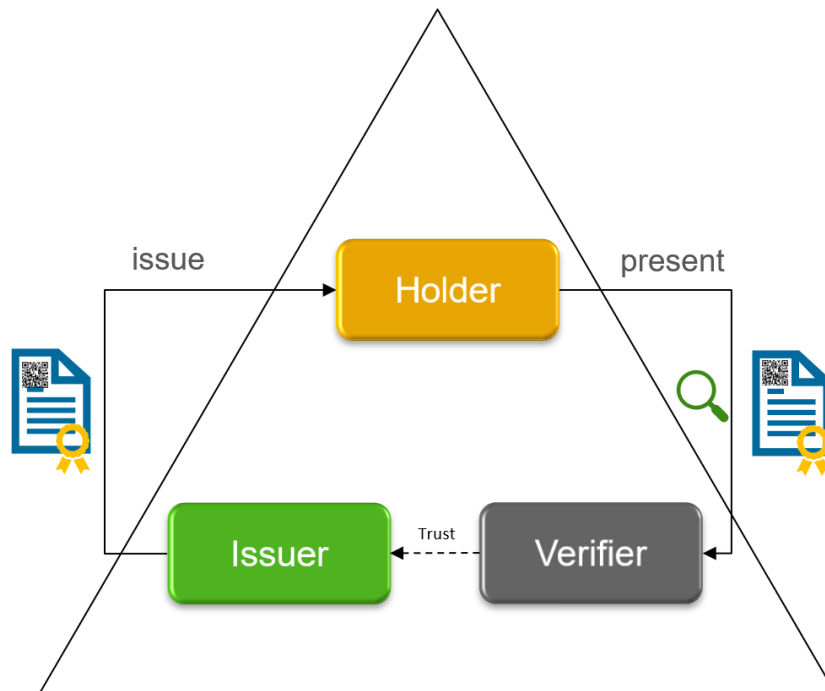


Figure 3: Triangle of Trust

The triangle of trust is the blueprint for EU Digital COVID Certificate (DCC) interoperability:

- **Holder:** A DCC owner (i.e., a citizen with a vaccination, test result, or recovery status (based on a positive test result) - note that the DCC can be held digitally within a wallet app or on paper (or both)
- **Issuer:** A national authority
- **Verifier:** A verifier (e.g., customs officers, police, or hotel staff)

But there is an important question: How does the verifier know which issuer is trustworthy? In a personal relationship, one would decide by experience. In this architecture, the DCCG tells the verifier which issuers are trustworthy by providing cryptographically anchored information.

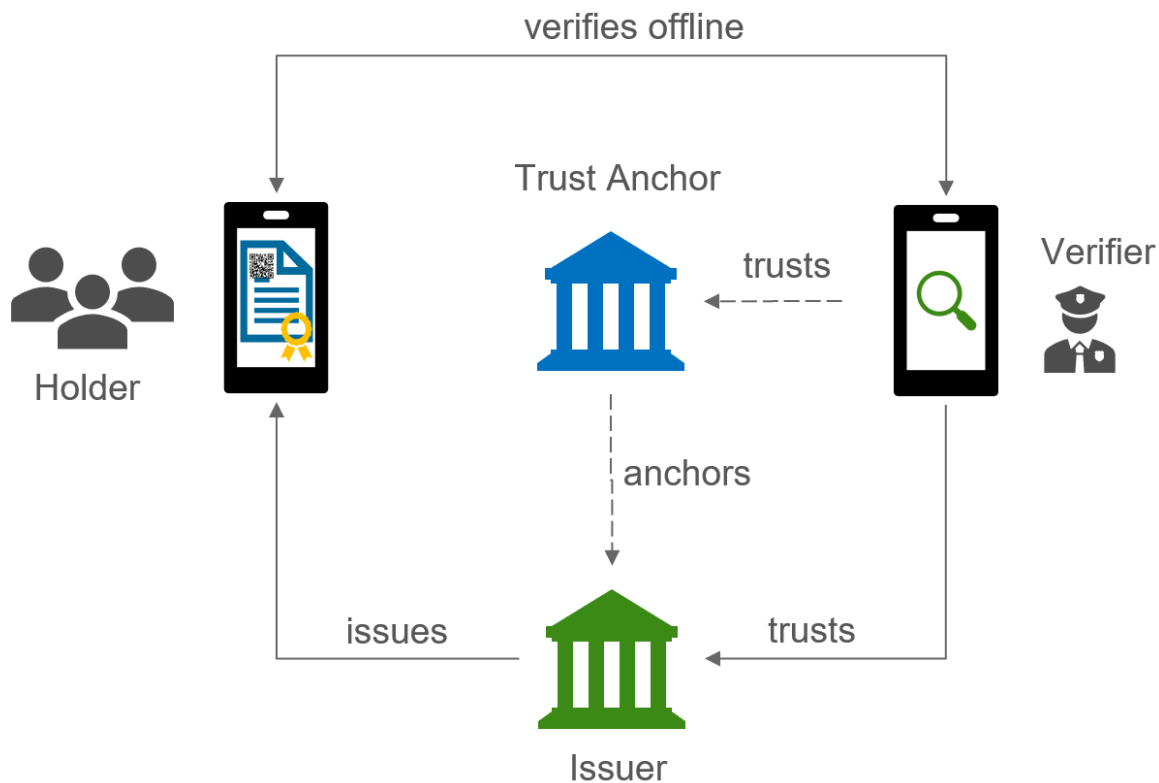


Figure 4: Trust Anchoring

### 3.2 Distribution of Verification Information

Exactly how each national app communicates with the corresponding national backend -whether via CDN, active push, or otherwise - is left to each country. Important here is the cryptographically secured E2E protection between the member states.

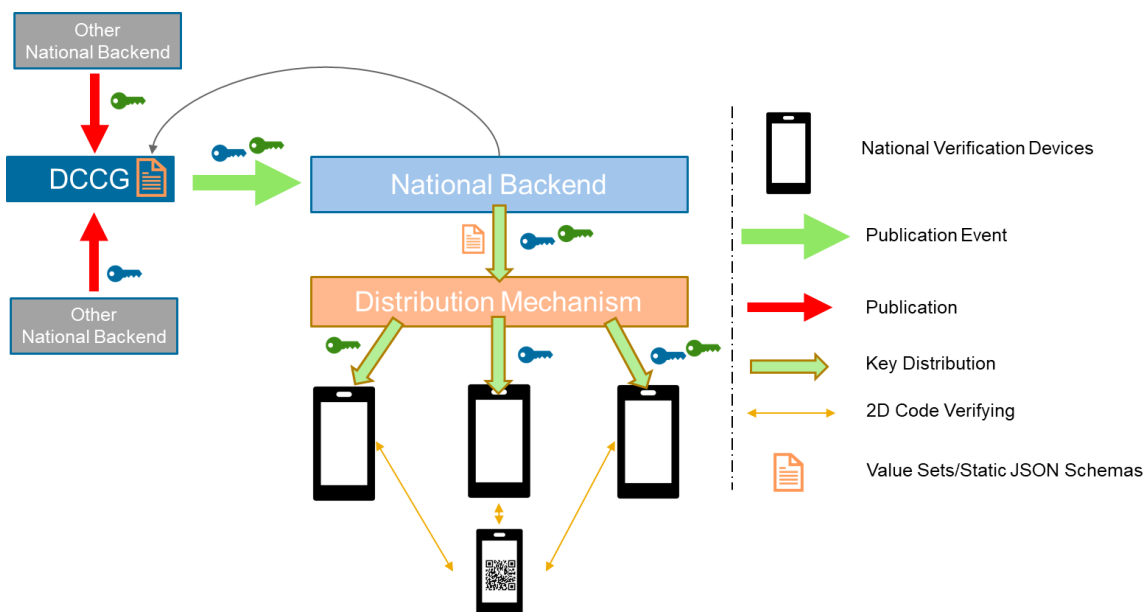


Figure 5: Distribution of Signing and Validation Information

### 3.3 Device-to-Device Communication

Device-to-device communication is built on a standardized 2D code and verifier format defined by the EU Trust Framework.

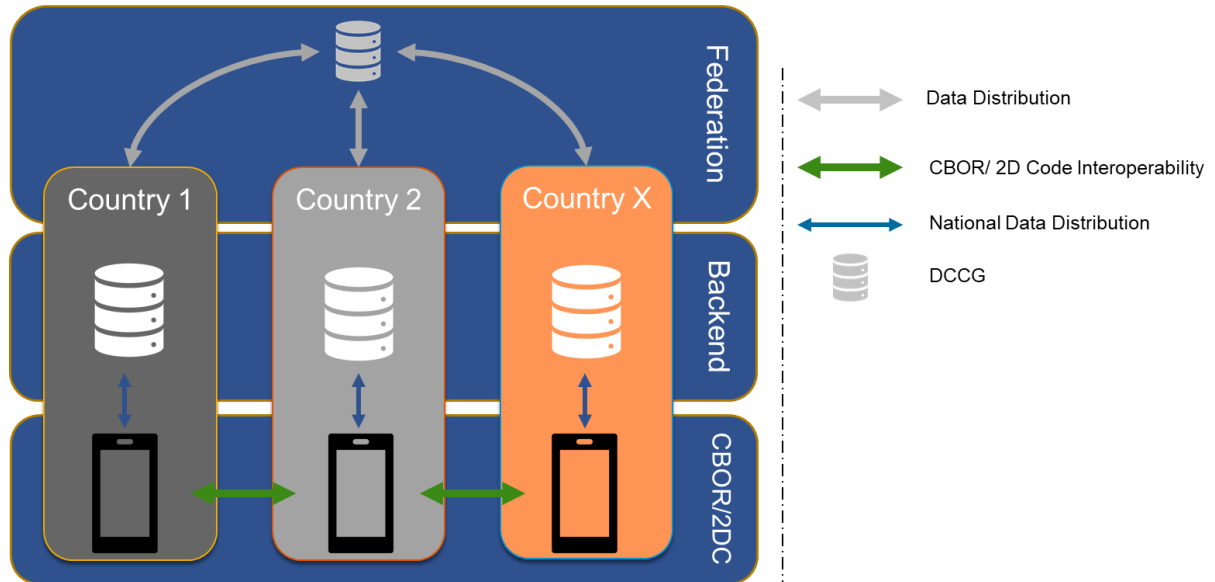


Figure 6: Distribution of Verification Information

### 3.4 Backend-to-Backend Communication

A direct backend-to-backend communication is not necessary, because the main purpose of the DCCG solution is to provide verification information. All participating national backends will provide that information to DCCG, which in turn stores the information and provides them for download. Nevertheless, bilateral communication between national backends is not categorically excluded—in a Self-Sovereign Identity scenario it is explicitly desired in the future.

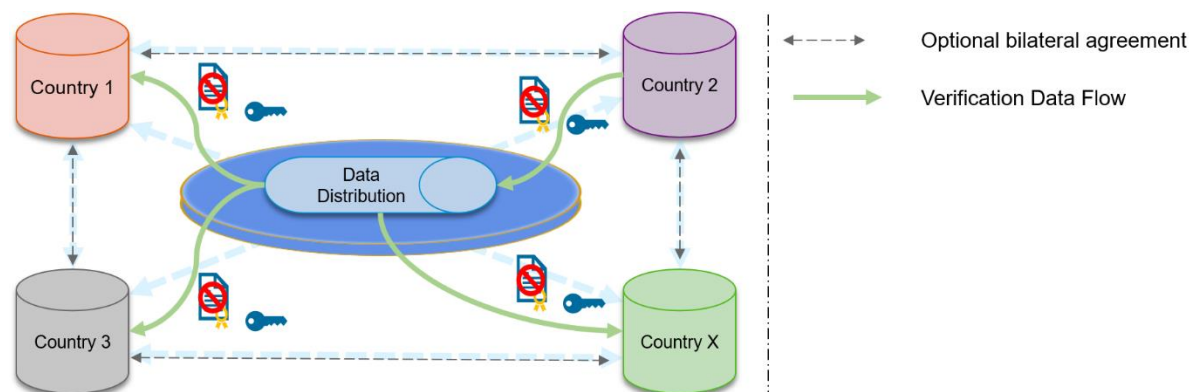


Figure 7: Indirect Backend-to-Backend Communication

### 3.5 Trust Establishment

To ensure that just data from trusted parties are accepted. The system contains a trust list which is signed entry by entry air gapped by an official signer. This signer, signs with his private key each request of onboarding and provides this signed information to the DCCG operator which can set this entry on the trust list. This guarantees that no external attacker or another party than the trusted signer can create valid records for the trust list. The public key of the trusted signer is shared out of band to the other parties, to establish an effective trust anchoring.

The trust list contains entries for authentication, signing of data packages, CSCAs etc.

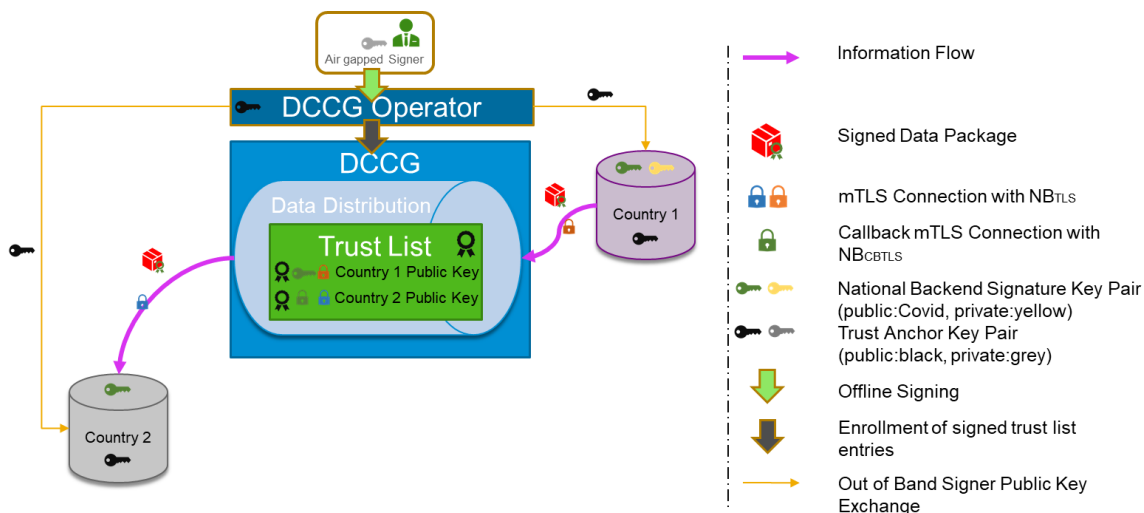


Figure 8: Trusted Data Exchange (Country 1 to Country 2)

To create the Trust List, an onboarding process is established, which onboards attendees by checking different criteria. If the process is successful, all certificates for connection and uploading DSCs (Document Signer Certificates) are anchored in the Trust List.

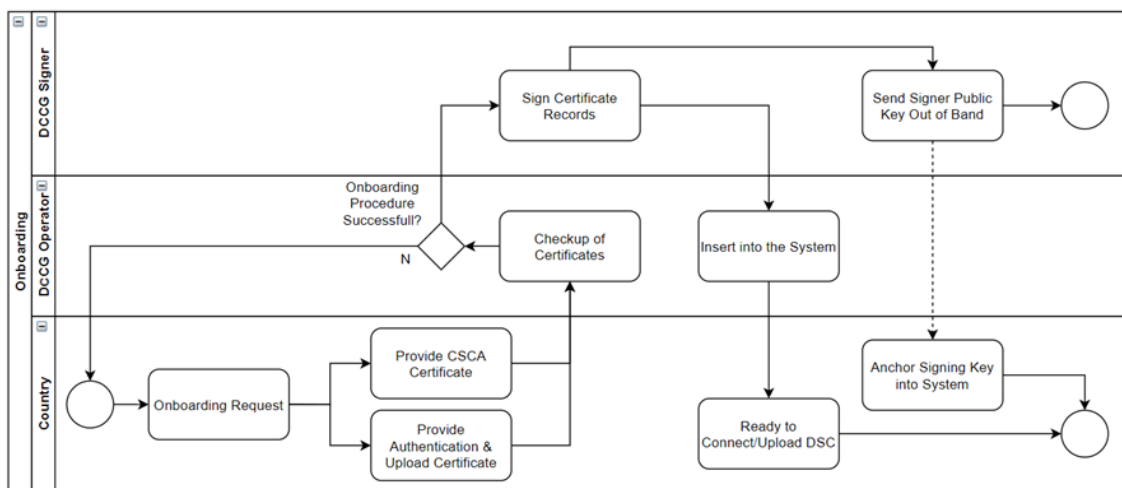


Figure 9: Onboarding Process

## 4 Data Structures

### 4.1 Data Types

#### 4.1.1 JSON Schemas

All JSON objects are based on JSON schemas following the JSON-Schema.org syntax (<https://json-schema.org/specification.html>)

#### 4.1.2 Static Data Content

The exchanged QR codes are based on a generated CBOR, which contains the minimal datasets of a certificate described by the Regulation on the Digital COVID Certificate and implementing acts adopted pursuant to it. The schema for EU DCC contents is available at <https://github.com/ehn-dcc-development/ehn-dcc-schema>.

Value set schema and implementations of the value sets are available at <https://github.com/ehn-dcc-development/ehn-dcc-valuesets>.

Value sets are hosted at the DCCG environment with the pattern /valuesets/{id}. The applicable id of each value set is defined in its field valueSetId (country-2-codes, covid-19-lab-test-manufacturer-and-name, disease-agent-targeted, etc).

#### 4.1.3 CMS Signed Data

To ensure the end-to-end secure data transfer of certificates, the cryptographic message standard is used. This Cryptographic Message Standard is defined in RFC5652<sup>2</sup>. Each information uploaded by the member states must be digital signed and embedded into this container format. This guarantees from the issuer to the receiving backend a secure data exchange. These packages are sent in Base64 format described in RFC4648<sup>3</sup>. over an HTTPS Interface which uses the media type **application/cms** described in RFC7193<sup>4</sup>.

The package content contains the signer certificates encoded as raw byte array.

### 4.2 Data Storage

Uploaded verification information is permanently stored because of the long lifetime of existing Digital COVID Certificates. The verification information must be explicitly deleted by the controlling member states if it's no longer needed.

#### 4.2.1 Database Requirements

As for database technology, we gathered the following requirements:

Requirement	Explanation
Object Storage	The database must support storage of different objects without needing schema changes
Strong Consistency	The database must support strong consistency, i.e., new data is fully replicated after each transaction


<sup>2</sup> <https://tools.ietf.org/html/rfc5652>

<sup>3</sup> <https://tools.ietf.org/html/rfc4648>

<sup>4</sup> <https://tools.ietf.org/html/rfc7193>

High Availability	We need redundant, replicated storage to avoid data loss
-------------------	--

Table 1: Database Requirements

 According to the CAP theorem for distributed data storage, only two of the three requirements *consistency*, *availability*, and *partition tolerance* can be fully met at the same time. Partition tolerance refers to resilience against message loss across the network. Since consistency and availability provide the greatest value for the national backends—and since both the number of partitions and message loss rate will be small—we focus on the first two requirements.

#### 4.2.2 Database

According to the requirements and the given environment within the DIGIT, a relational MySQL Database has been chosen.

#### 4.2.3 Data Format

##### 4.2.3.1 Trusted Party Table



Field	Description	Data Type
<b><i>Id</i></b>	Primary key	Long
<b><i>Timestamp</i></b>	Timestamp of the Record	Timestamp
<b><i>Country</i></b>	Country Code	varchar(2)
<b><i>Sha256Fingerprint</i></b>	SHA256-fingerprint of the certificate	varchar(*)
<b><i>Certificate Type</i></b>	Type of the certificate (Authentication, Signing, Issuer, Client, CSCA)	varchar(*)
<b><i>RawData</i></b>	Raw Data of the certificate	binary
<b><i>Signature</i></b>	Signature of the Trust Anchor	varchar(*)

Table 2: Trusted Party Table

Certificate Types:

Type	Description
<b>Authentication</b>	Certificate which the member state is using to authenticate at DCCG (NBTLS)
<b>Upload</b>	Certificate which the member state is using to sign the uploaded information's (NBUS)
<b>CSCA</b>	Country Signing Certificate Authority certificate (NBCSCA)

Table 3: Certificate Types

-  The DCCG will add client certificates as well (country: EU).
-  The DCCG trust anchor certificate is sent out-of-band to the member states to anchor the public key trust network.

##### 4.2.3.2 Signer Information Table

Field	Description	Data Type
<b><i>Id</i></b>	Primary key	Long

<b>Timestamp</b>	Timestamp of the Record	Timestamp
<b>Country</b>	Country Code	varchar(2)
<b>Sha256Fingerprint</b>	SHA256-fingerprint of the certificate	varchar(*)
<b>Certificate Type</b>	Type of the certificate (Authentication, Signing, Issuer, Client, CSCA)	varchar(*)
<b>RawData</b>	Raw Data of the certificate	binary
<b>Signature</b>	Signature of the Uploader	varchar(*)

Table 4: Signer Information Table

Certificate Types:

Type	Description
<b>DSC</b>	Certificate which the member state is using to sign documents (NBDSC)

#### 4.2.3.3 Audit Event Table

Field	Description	Data Type
<b>Id</b>	Primary key	Long
<b>Timestamp</b>	Timestamp of the Record	Timestamp
<b>Country</b>	Country Code	varchar(2)
<b>Uploader Sha256Fingerprint</b>	SHA256-fingerprint of the certificate	varchar(*)
<b>Authentication Sha256Fingerprint</b>	SHA256-fingerprint of the certificate	
<b>Event</b>	Event which occurs	varchar(*)
<b>Description</b>	Description of the Event	varchar(*)

Table 5: Audit Table

#### 4.2.3.4 User Rights

Role	Table	Grants	Actor
<b>Admin</b>	Signer Information	Full Access	DCCG Operator
<b>Admin</b>	Trusted Party Table	Full Access	DCCG Operator
<b>Admin</b>	Audit Table	Full Access	DCCG Operator
<b>User</b>	Signer Information	Read/Write/Delete	DCCG Application
<b>User</b>	Audit Table	Write	DCCG Application
<b>User</b>	Trusted Party Table	Read	DCCG Application

Table 6: Signer Information Table



## 5 Building Block Schema

In a rough overview, DCCG is built on top of Tomcat, which is directly linked to a F5 load balancer, combined with a reverse proxy. The main modules are for upload, download, and revoke. A trust list API provides additionally all anchored party information's.

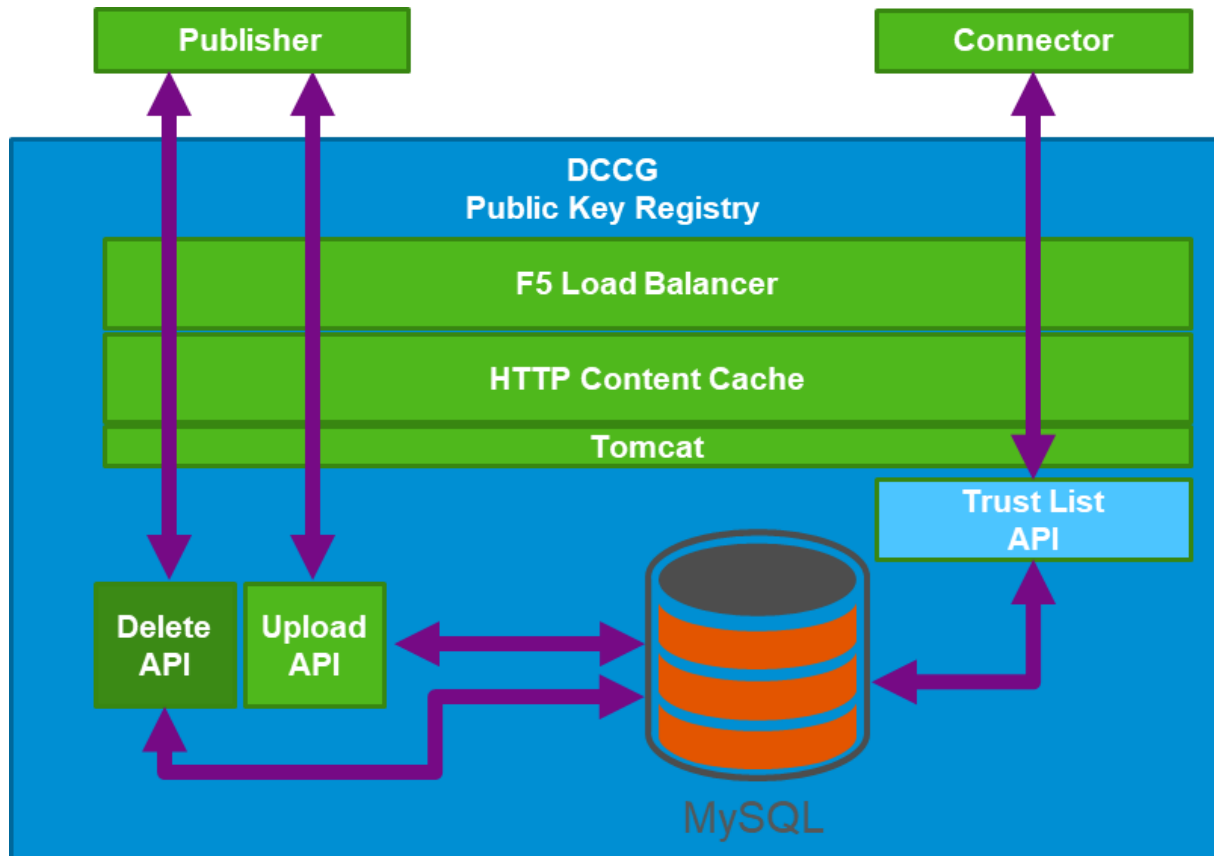


Figure 14: Building Block Schema

- ⚠ DCCG performs mutual authentication with the national backends—its API validates the provided server certificate of the national backend and proves its identity by way of a client certificate to them. Each national backend has to explicitly whitelist DCCG’s client certificate and has to provide its server certificate’s public key to the DCCG for whitelisting.

## 6 Interfaces

### 6.1 Overview

DCCG provides a simple REST API with common upload and download functionality for trusted information.

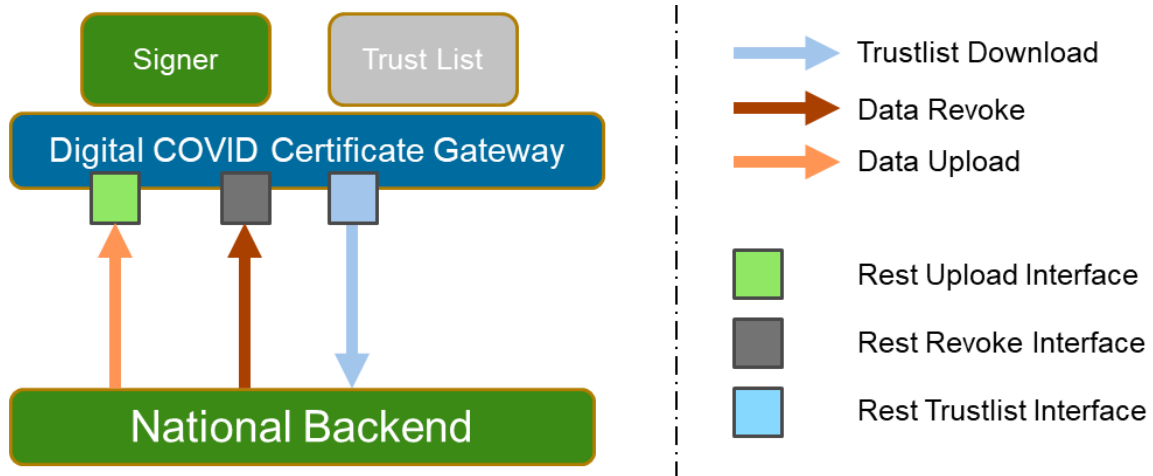


Figure 15: API Overview

The functions of the interfaces are:

- Upload/Revoke of signer information's, and
- Download Trust List information's

All these actions are represented by GET, POST, and DELETE actions.

For detailed description of REST interfaces, we rely on the Open API Specification 3.0<sup>5</sup>. This allows a comprehensive human-readable and machine-readable representation of all aspects of the defined interface.

We define access points according to the following scheme:

---

<sup>5</sup> <https://swagger.io/specification/>

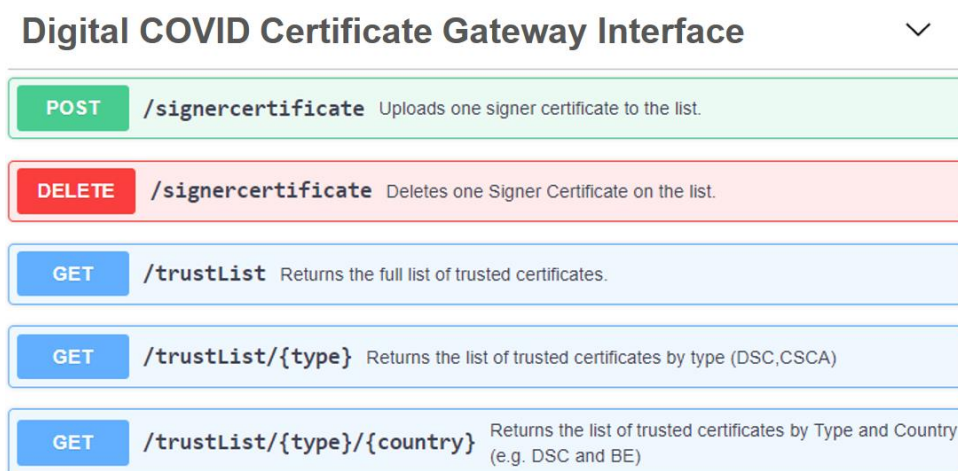



Figure 16: Interface Definition Overview

 The Digital COVID Certificate Gateway Service API performs no signing of data packages. Each national backend needs to format and sign the data by itself.

## 6.2 MIME Type

The packages for Upload and Delete interfaces are sent in Base64 format described in RFC4648<sup>6</sup>. The used MIME Type is

**application/cms**

described in RFC7193<sup>7</sup>.

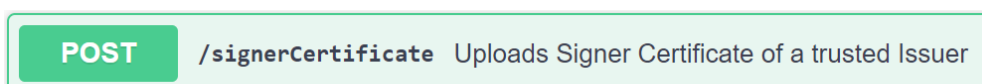
Usage:

```
Content-Type: application/cms
Content-Transfer-Encoding: base64
```

## 6.3 Upload Interface

### 6.3.1 Overview

The upload interface consists of one POST request (in version 1.0), to upload information about the verification. The registration of the certificate takes max 10 seconds. After this, the successful registration can be verified by fetching the list of valid certificates and ensuring that the uploaded certificate is on the list.



<sup>6</sup> <https://tools.ietf.org/html/rfc4648>

<sup>7</sup> <https://tools.ietf.org/html/rfc7193>

Figure 17: Upload Interface

### 6.3.2 Parameters

Name	Description
<b>Content-Type</b> * required string (header)	Example : application/cms <input type="text" value="application/cms"/>
<b>Content-Transfer-Encoding</b> * required string (header)	Example : base64 <input type="text" value="base64"/>

Figure 18: Upload Parameters

### 6.3.3 Responses

Code	Description
201	Signer Information was created successfully.
401	Unauthorized. No Access to the system. (Client Certificate not present or whitelisted)
	Media type <input type="text" value="application/json"/>
	Example Value   Schema <pre>{   "Code": "0x001",   "Problem": "[PROBLEM]",   "Sent Value": "[Sent Value]",   "Details": "..."} </pre>
403	Forbidden. Signer Information package is not accepted. (hash Value or signature wrong, client certificate matches not to the signer of the package)
	Media type <input type="text" value="application/json"/>
	Example Value   Schema <pre>{   "Code": "0x001",   "Problem": "[PROBLEM]",   "Sent Value": "[Sent Value]",   "Details": "..."} </pre>
406	Content is not acceptable. (Wrong Format, no CMS, not the correct signing etc.)
	Media type <input type="text" value="application/json"/>
	Example Value   Schema <pre>{   "Code": "0x001",   "Problem": "[PROBLEM]",   "Sent Value": "[Sent Value]",   "Details": "..."} </pre>
409	Conflict. Chosen UUID is already used. Please choose another one.

Figure 20: Upload Responses

### 6.3.4 Transmission Protocol

During the upload, the uploader identity is extracted from the client certificate. If the client certificate is valid, the submitted content is validated and stored in the database.

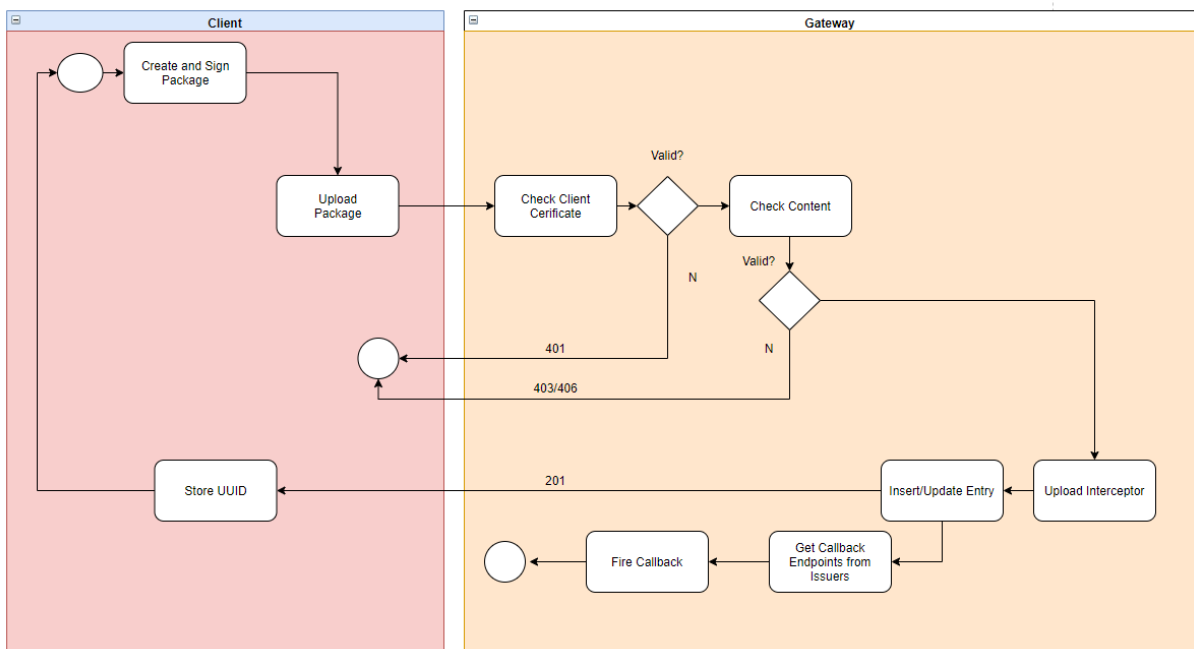


Figure 21: Upload Transmission Process

Content Checks:

Check	Description
1 Uploader Certificate	Does the used Uploader Certificate match to the registered TLS Client certificate for the Country?
2 CSCA Checkup	Does the contained certificates match to the registered CSCA(s)?
3 Already Exist Check	Is there still an existing certificate with the same SHA-256 Fingerprint? If yes, write an entry in the audit log.

## 6.4 Signer Delete Interface

### 6.4.1 Overview

The upload interface contains a DELETE request to delete signer information. The deletion of the certificate takes max 10 seconds. After this, the deletion can be verified by fetching the list of valid certificates and ensuring that the deleted certificate is not included on the list.

```

DELETE /signercertificate Deletes one Signer Certificate on the list.
    
```

Figure 22: Delete Interface

### 6.4.2 Parameters

The request requires a single header parameter, X-RECORD-UUID (SHA256-fingerprint of the cert):

Name	Description
X-RECORD-UUID string (header)	XJ32-SKFJ3-KJKFJR-KDJEJ4

Figure 23: Delete Parameters

### 6.4.3 Responses

Responses		
Code	Description	Links
200	OK.	No links
401	Unauthorized. No Access to the system. (Client Certificate not present or whitelisted)	No links
	Media type <input type="text" value="application/json"/>	
	Example Value   Schema <pre>{   "Code": "0x001",   "Problem": "[PROBLEM]",   "Sent Value": "[Sent Value]",   "Details": "..."} </pre>	
403	Forbidden call in cause of invalid client certificate for this resource.	No links
	Media type <input type="text" value="application/json"/>	
	Example Value   Schema <pre>{   "Code": "0x001",   "Problem": "[PROBLEM]",   "Sent Value": "[Sent Value]",   "Details": "..."} </pre>	
410	Object is permanently gone.	No links
	Media type <input type="text" value="application/json"/>	
	Example Value   Schema <pre>{   "Code": "0x001",   "Problem": "[PROBLEM]",   "Sent Value": "[Sent Value]",   "Details": "..."} </pre>	

Figure 24: Delete Response

### 6.4.4 Transmission Protocol

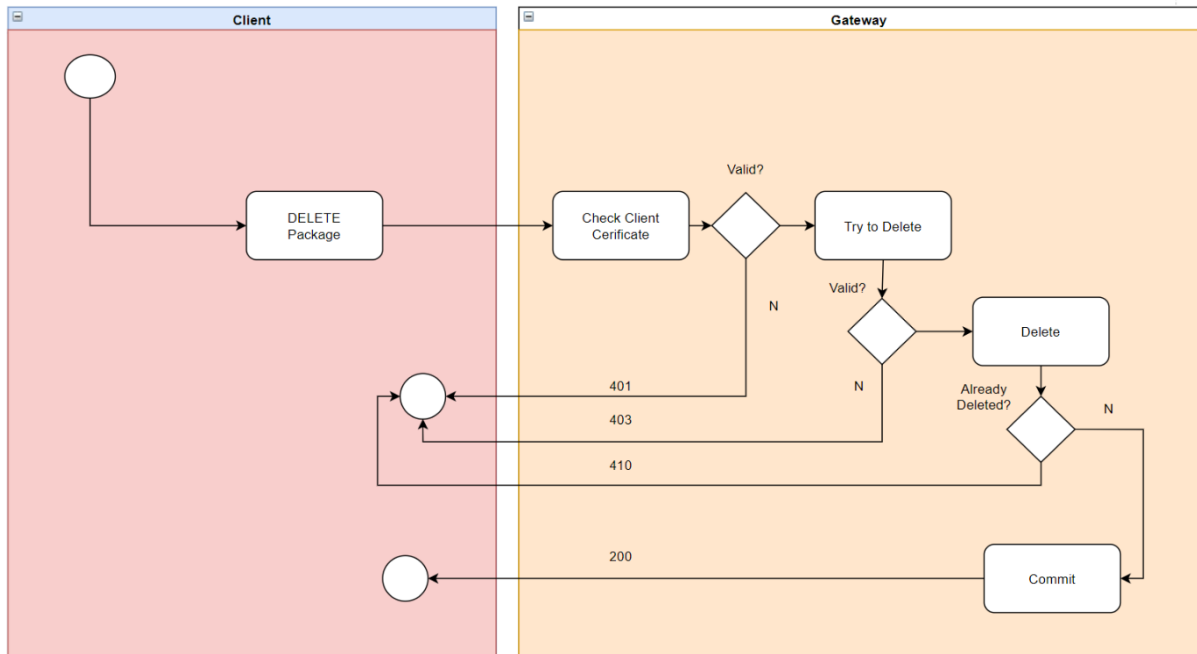


Figure 25: Delete Flow

## 6.5 Trust List Interface

### 6.5.1 Overview

The trust interface consists three GET requests to retrieve the DCCG Trust Lists.

GET	<code>/trustList</code>	Returns the list of trusted certificates
GET	<code>/trustList/{type}</code>	Returns the list of trusted certificates
GET	<code>/trustList/{type}/{country}</code>	Returns the list of trusted certificates by Type and Country

Figure 26: Trust List Interface



The trust list provides an aggregated view on a given query, not on the uploaded batch of certificates.

### 6.5.2 Parameters

Name	Description
<b>Type</b> * required string (header)	Example : CSCA,DSC <input type="text" value="CSCA,DSC"/>
<b>Country</b> * required string (header)	Example : CSCA,DSC <input type="text" value="CSCA,DSC"/>

Figure 27: Trust List Interface



Depending on the Route, the two parameters can be optional or are not necessary.

### 6.5.3 Responses

The Trust List calls return an array of trusted entries which contains the SHA256 fingerprint of the certificate and the related information about the country, certificate type and the BASE64 encoded raw data of the certificate itself.



The signature is calculated over the rawdata and can be checked against the trust list.

NOTE: The Type Authentication, Upload and CSCA can just be verified over the Public Key Certificate which was received out of band during the onboarding.

The Type DSC signature can be validated against the Upload certificate of the country and against the onboarded CSCAs on the trust list.



Code	Description
200	Returns the full list of trusted parties.



Media type

Controls Accept header:

Example Value | Schema

```
[
  {
    "ID": "32434-234234-234-2342424-23434",
    "TimeStamp": "2021-03-05",
    "Country": "DE",
    "CertificateType": "CSCA",
    "SHA256Fingerprint": "69c697c045b4cdaa441a28af0ec1bb4128153b9ddc796b66bfa04b02ea3e103e",
    "Signature": "o53CbAa77LyIMFc5Gz+82Jc2756dg/SdLayw7gx0GrTcinR95zftLr8nNHgJMYLX3rD8Y11zB/OsyT0VLCCDZr+e6gtMms8qr0qMzw1G74cSPiKCb6TEpc/pB8Gx1jtofInvksLjJzW3Pu4fbKz6KikdUjXA81xYx//aosd7qWxo2lnxbJlo1URXw/BINanoKj+R2SSCheZzi8dbUjmfOP8IZUFvtpf3isyMpaD+5+gpcGgNqNz9aUPvWk++jTlKj+e4ZtFkUh0nPR4hYsmXct9jn32lk2M3r3CcmvgvVh+VIRYRGSEmgjGy2Ewzva5nVhsaA+/udnmbYQw9LjA0Q==",
    "RawData": "MIICyDCCAbCgAwIBAgIGAXR3DUUMA0GC5qGSIb3DQEBBQUAMBwxCzAJBgNVBAYT\nAkrFMQ0wCwYDVQQDDARKZW1vMB4XDTEwMDgyNzA4MDY1M1oXDTEwMDkxMDA4MDY1\nnM1owHDELMAkGA1UEBhMCREUxDALBgNVBAMMBGR1bW8wggEiMA0GC5qGSIb3DQEB\nnAQUAA4IBDwAwggEKAoIBAQCkR0TEJ004z0ks40MAovcyxuPpeZuR1JykNNF3OR+\n\nvFWJLJtdVGRjtuqSuKcghLa/ci+0yIs30eitGtajqfIukYksvX2Lx0ZDYDUBnpGQ\n\nnDPNMVmpEavDBbvKON8C8K036pC41bNvwkTrfUyZ8iE+hV2+kj1SHUyW7jweUoIw\n\nnNmMiaXXPiMIOj7D0qnmM+iTGN9g/DrJ/IvvsGiGpK3Q1Q5pnHs2BzvrSw4LFAZ8c\n\nnSQfWkheZVhfQf26mJFdEzowrzfzForDdeFAPiIirhufe3jWfXj1thfztu+VSMj84\n\nnsDqodEt2VJOY+DvLB1Ls/26LSmFtMnCEuBAhkbQ1E0tbAgMBAAGjEDAOAwGA1Ud\n\nnEwEB/wQCMAAwDQYJKoZIhvcNAQEFBQADggEBABaMEQz4Gbj+G0SZGZaIDoUF
```

Figure 28: Trust List Responses

-  The “RawData” Field contains an X509 certificate, uploaded over signerCertificate upload.
-  The Signature is calculated over the byte presentation of the raw data.

## 6.6 DSC Delta Download Extension

### 6.6.1 Introduction

A higher amount of document signer certificates is downloaded and uploaded over the gateway and later on distributed to verifiers. With each new DSC the overall distribution volume in the network increases by around 1.5 KB multiplied with the number of verifiers. Currently most distribution systems are designed as lists which are downloaded in one piece, but this would result in a heavy increase of the download volume when a lot of more DSCs are used. To solve this problem a “pagination” and “delta” concepts are described in this document to optimize the network traffic.

### 6.6.2 Assumptions

How many certificates DCCG will store? Every national backend (one per country?) will provision their trusted certificates during an onboarding procedure. We assume it’ll be up to ~100 countries with max 500 certificates each, so the upper bound could be 100\*500\*1.5KB = 73MB entries for the whole EU certificate store, which is not too much. At the certificate revocation procedure we’ll not delete the

certificate, but will set its revocation timestamp. Later (after 1..2 months) revoked certificates can be physically deleted from DCCG DB when we're sure that information was already shared with all national backends.

### 6.6.3 Scope

The scope is the enhancement of the DCC Gateway and the template applications to support query requests with pagination and delta download for DSCs.

### 6.6.4 Solution

To display lists of data, clients do not always need to download the complete set of data from the backend. It is enough to load the data in pages - this reduces traffic and load on both client and server sides. To do this, the API will be enhanced with a URL extension.

The delta download will be enabled by the header parameter:

#### **If-Modified-Since**

In one special use case for the initial downloading (Delta=0), the response would be very big and therefore the header is extended by a paging mechanism for a optimized download:

```
Header: If-Modified-Since=Sat, 29 Oct 1994 19:43:31 GMT  
GET /URL?page=<page>&pagesize=<size>
```

If the header parameter is set, the route will return in the list deleted entries which contain just empty entries next to the KID. If such an entry is received, the entry can be removed from the local trust list.

Example:

Current Response of a existing certificate:

```
[  
  {  
    "kid": "qroU+hDDovs=",  
    "timestamp": "2022-02-21T09:54:42.802Z",  
    "country": "DE",  
    "certificateType": "DSC",  
    "thumbprint": "aaba14fa10c3a2fb441a28af0ec1bb4128153b9ddc796b66bfa04b02ea3e103e",  
    "signature": "o53CbAa77LyIMFc5Gz+B2Jc275Gdg/SdLayw7gx0GrTcinR95zFTLr8nNHgJMY1X3rD8Y11zB/0syth  
... W+VIrYRGSEmgjGy2EwzvA5nVhsaA+/udnmyQw9LjAQQ==",  
    "rawData": "MIICyDCCAbCgAwIBAgIGAXR3DZUUMA0GCSqGSIb3DQEBBQUAMBwxCzAJB ...  
Jpux30QRhsNZwkmEYSbRv+vp5/obgH1mL5ouoV5I="  }  
]
```

Future Response of a deleted Certificate, when it's existing on the list:

```
[
  {
    "kid": "qroU+hDDovs=",
    "timestamp": "2022-02-21T09:54:42.802Z",
    "country": "EU",
    "certificateType": "DSC",
    "thumbprint": null,
    "signature": null,
    "rawData": null
  }
]
```

### 6.6.5 Parameters

The [pagination](#) /delta parameters are:

Parameter	Description	Default value
page	Number of the page requested.	0
pagesize	a page size – how many records should be loaded from server	100

All pagination parameters are optional, when any of them are not passed, the backend will return the default response.

In addition to the new pagination parameters, the HTTP Header “If-Modified-Since” can be used to highlight what data shall be skipped.

To load trustLists the request might look like this now:

```
GET /trustLists?page=0 & pagesize=100
GET /trustLists/DSC?page=0 & pagesize=100
GET /trustLists/DSC/DE?page=0 & pagesize=100
```

The certificate delta download can be established by using the pagination query approach with new modifiedSince parameter in the trustedLists request:

```
Header: If-Modified-Since=20211201
GET /trustLists
```

or, with additional pagination parameters:

```
Header: If-Modified-Since=20211201  
GET /trustLists?page=1&pagesize=2
```

### 6.6.6 Side Conditions/Assumptions

For the usage of this feature the following things must be considered:

- The certificates will not appear endless in the system. After a grace period of 2 weeks, the deleted entries will be purged from the system
- The backend does from time to time a full sync

### 6.6.7 Use Scenarios

Scenario 1 is the download of the entire list. This can be done either by using the default route:

```
GET /trustLists
```

or by using the paging:

```
GET /trustLists?page=0 & pagesize=100
```

If the sync shall include the delta, enable the if modified since header, either by using paging or without:

```
Header: If-Modified-Since=20211201  
GET /trustLists?page=1&pagesize=2
```

### 6.6.8 Recommendation for Verifier/Wallet App

With an increasing amount of DSCs in the system, it should be considered to implement the Delta Download Approach or similar approaches as well for Verifier and Wallet Apps to reduce the network traffic. This can be realized over an seperated delta by KID and the certificates it self with the If-Modified-Since Header:

```
Header: If-Modified-Since=Sat, 29 Oct 1994 19:43:31 GMT  
GET /signerCertificateStatus/delta
```

The route returns then a list of KID which express the status of each signer certificate in the defined time:

```
{  
  "update": ["33333d=", "333311=", "55554=],  
  "deleted": ["3115adf=]
```

```
}
```

To pick exactly the updated certificates from the trustlist, the app should use the following route to receive the concrete entries:

```
POST /signerCertificateUpdate
```

```
[“33333d=”,“333311=”,“55554=”]
```

The response contains a list with the new certificates sorted by country .

```
{  
  “DE”: [“MII...”,“MII...”],  
  “NL”: [“MII...”,“MII...”]  
}
```

### 6.6.9 Reference Implementation Scope

The DCCG Reference Implementation will be enhanced with the following points:

- The gateway trustlist will be enhanced with the described mechanism
- The verifier distribution service will be modified :
  - signerCertificateUpdate Route will be updated with new route
  - signerCertificateStatus Route will be updated with new route
- DCCG Verifier App is enhanced with the delta download

## 7 DSC Publication

### 7.1 Assumptions and constraints

The current module architecture has been made with next assumptions:

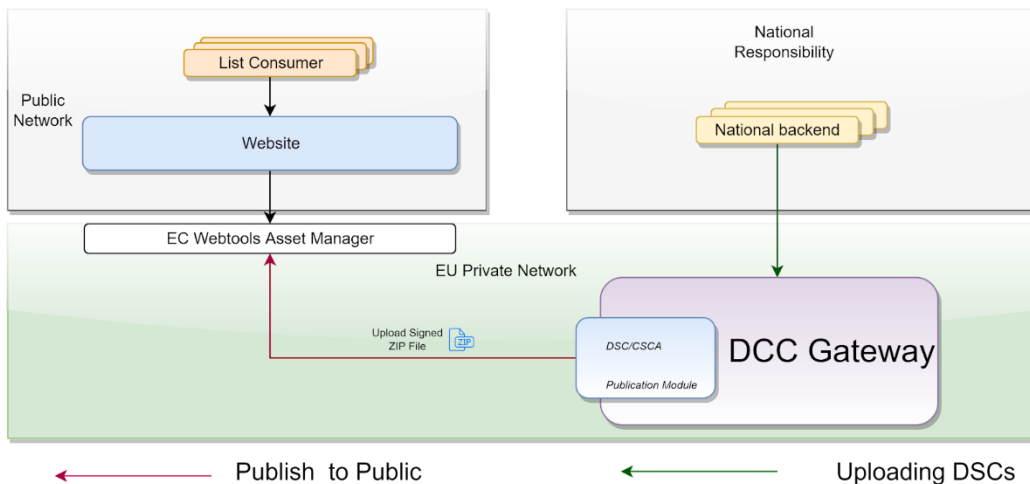
- The DCC Gateway does not communicate with DSC Publication in any way and is not public accessible
- The DSC Publication module can provide information in different formats, if necessary
- DSC Publication is open for any anonymous users as a public resource and over an EU website

### 7.2 Architecture

#### 7.2.1 Solution

Using the Spring boot as the main framework, the gateway is enhanced by a publication module. This module can be activated by a Spring Boot Profile. The DSC Publication module provides the functionality to push a signed zip file periodically to the EU Asset Manager which stores all uploaded content for later publishing.

## eHealth Network



### 7.2.2 Publishing Format

The data container contains a list of DSCs or CSCAs which are trusted by the EU DCC Gateway. Each file contains as well a README and a License File for usage which must contain disclaimers, purpose descriptions etc.

The format of the container is ZIP compressed with the “deflate algorithm” which must be signed as CMS to ensure the Package integrity and authenticity.

The public key for the verification is placed on the publishing website in PEM format.

#### Data Structure:

The main data structure contains a readme text file, a license text file, a version text file and a folder structure which is sorted by DSC/CSCA, the domain (e.g. DCC), the country and all certificates in PEM format.

- Readme.txt*
- Licence.txt*
- Version.txt*
- DSC**
  - **DCC**
    - **DE**
      - *DSC1.pem*
      - *DSC2.pem*
- CSCA**
  - **DCC**
    - **DE**
      - *CSCA1.pem*
      - *CSCA2.pem*

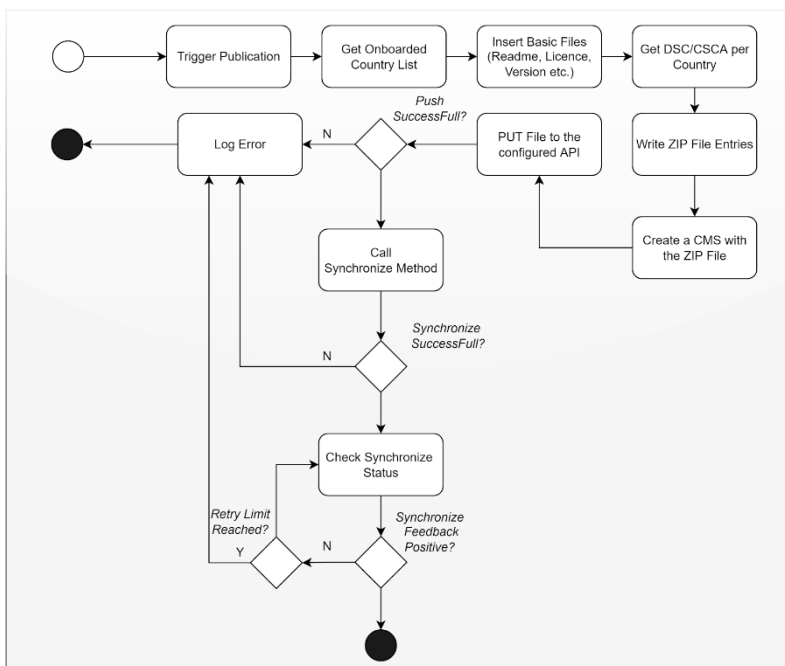
The public key has a PEM representation contained in the ZIP file as signer information and can be verified by a PEM thumbprint stored on the website encoded in hexadecimal format, for instance:

65:B0:9A:81:6F:F5:C7:67:3D:6E:F1:3F:03:5D:ED:4A:8A:C4:B9:61:B7:59:55:F1:D5:92:6D:C3:C6:3E:CB:5C

This thumbprint must be published on the website to give verifiers the chance to validate the integrity of the used certificate for the CMS signature.

### 7.2.3 Publication Process

The zip file and the public key will be pushed automatically over an HTTP request. This process is defined as follows:



All calls are REST calls to a configured API which is in this case the EU Asset Manager API.

Within the gateway, three endpoints including it's headers and access credentials will be configured:

URL	Method	Mandatory Headers	Payload
synchronize	POST	Authentication, ContentType	Static Configurable Payload
upload	PUT	Authentication, ContentType	CMS Message as binary
synchronizeStatus	POST	Authentication, ContentType	Static Configurable Payload

## 8 Technology Choice

Note: We selected the technologies below from a restricted set of alternatives provided by the designated platform operator (DIGIT<sup>8</sup> in Luxemburg).

Component	Technology	Core Features
<b>REST API</b>	Java / Spring Boot	Powerful, versatile web framework
<b>Database</b>	MySQL	Supports JSON documents
<b>Load Balancer</b>	F5	Reverse proxy, load balancing, detailed traffic metrics, SSL offloading, Client Auth
<b>Web Server</b>	Tomcat	

Table 8: Technology Choice

---

<sup>8</sup> [https://ec.europa.eu/info/departments/informatics\\_en](https://ec.europa.eu/info/departments/informatics_en)



## 9 Implementation Roadmap

Feature	Expected Version
Onboarding	1.0
mTLS/Signed Data Upload	1.0
Download/Upload of Signer Certificates	1.0
Revoke of uploaded Signer Certificate	1.0
Static Data Content	1.0
Onboarding of CSCA Certificates	1.0
Check of Signer Uploads against CSCA Certificates	1.0
Trust lists	1.0
Callbacks for Changes	TBD (candidate for 1.0)
Upload of Validation Schemas	TBD
Validation and Revocation Rules	TBD

Table 9: Roadmap

## 10 Glossary

Term	Description
<b>2D code / QR code</b>	Two-dimensional barcode
<b>Certificate</b>	Technical Certificate like X509.
<b>Certificate issuance</b>	The act of creating a Digital COVID Certificate
<b>Civil identity</b>	Defined by the eHN Minimal Data Set: Person Name (The legal name of the vaccinated person (surname(s) and forename(s) in that order), Person Date of birth
<b>CMS</b>	Cryptographic Message Syntax, RFC 5652
<b>Crypto Material / Cryptographic material</b>	All material, including documents, devices, or equipment that contains cryptographic information and is essential to the encryption, decryption, or authentication of telecommunications
<b>DCC</b>	Digital COVID Certificate. Includes also the machine processable part of the according paper document. Currently, it can be differentiated between three different types: vaccination, test result, recovery certificate (based on a positive test result)
<b>DCC Gateway / DCCG</b>	Exchange of public keys, certificates, and other crypto material between national backends
<b>DCCI</b>	Digital COVID Certificate Identifier. Universally unique ID assigned to the DCC when issued by the issuer app
<b>Holder / DCC Owner</b>	Person in possession of a Digital COVID Certificate
<b>Holder verification / verification</b>	The process of verification to answer the question, if a person who states to be the legal holder of a Digital COVID Certificate is the same person who holds the certificate
<b>ICAO</b>	International Civil Aviation Organization is a specialized agency of the United Nations
<b>Issuer</b>	A person or system that works in behalf of an Issuing (health) authority to issue Digital COVID Certificates
<b>Issuer App / Issuer application frontend</b>	The application that is used by the issuer to issue Digital COVID Certificates. The issuer application frontend provides a user interface that is used by the issuer to enter the necessary data. The application will communicate with the backend / dcca-issuer-service to implement the process of digital COVID certificates signing. <i>Component: dcca-issuer-web</i>
<b>Issuing authority</b>	Institutions named by a member state to act in its will for issuing digital COVID certificates
<b>JSON</b>	JavaScript Object Notation is an open standard file format, and data interchange format, that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and array data types (or any other serializable value)
<b>Key pair</b>	Public and private key in the context of asymmetric cryptography
<b>Member states / MS</b>	Member state of the European Union (currently 27) or a third country in scope of the system (e.g. based on an adequacy decision)
<b>National certificate backend / backend</b>	The issuer backend is accessed by the issuer frontend application and the respective wallet apps. The backend itself

	publishes its public keys to the DCCG where they can be distributed to other MS. Each MS hosts its own issuer backend. DCC signing, TAN generation, TAN validation, public key publication, and Digital COVID Certificate validation/revocation inf, etc.
<b>NTP server</b>	The Network Time Protocol (NTP) is a networking protocol for clock synchronization between computer systems over packet-switched, variable-latency data networks.
<b>Offline verification</b>	Process of verification without the need of an active internet connection during the time of verification. Attention: offline verification will make use of an internet connection that downloads necessary crypto material in advance.
<b>Onboarding</b>	The structured process of including and gaining interoperability on a technical and organizational level of a member state to issue and verify Digital COVID Certificates
<b>Second factor / 2FA</b>	Authentication method in which a computer user is granted access only after successfully presenting two pieces of evidence (or factors) to an authentication mechanism. E.g., a transaction authentication number.
<b>Service X</b>	Sample service, e.g., in a booking process
<b>SMS</b>	Short Message Service. Text messaging service component
<b>TAN</b>	Transaction authentication number
<b>Trust Anchor</b>	An authoritative entity for which trust is assumed and not derived
<b>Trust Lists / CTL</b>	A predefined list of items signed by a trusted entity
<b>Verifier</b>	Person that verifies DCCs
<b>Verifier App</b>	verifies Digital COVID Certificates with the help of the dcca-verifier-service. Consists of a frontend (to be used by the verifier) and a backend (providing trusted key for a member state). <i>Component: dcca-verifier-app</i>
<b>Wallet App</b>	Application that holds a Digital COVID Certificate and provides a frontend to be used by the holder. <i>Component: dcca-wallet-app</i>

## APPENDIX

### **(A) Authentication:**

- Connections to DCCG will be over HTTPS using TLS with cryptographic settings that meet or exceed the relevant ENISA recommendations on algorithms, key sizes, and parameters.
- There will be mutually authenticated TLS (mTLS) connections between DCCG and each national backend.
- Trust validation happens by means of certificate validation based on TLS client, TLS server certificate, and server name (CN/subjectOtherName, according the CAB forum standard).
- This is combined with pinning by both parties on explicit certificate, a dedicated CA in the chain, or issuing CA by DCCG.
- For this reason, each national backend will inform DCCG of the certificate in the chain below which they consider any client certificate as being authorized by the national backend to connect to DCCG on their behalf.
- In the most extreme case, this may be just the actual leaf certificate or a self-signed certificate. In general, implementers are urged to provide a (dedicated) CA certificate as to minimize operational logistics (from the perspective of the DCCG operator) around key rollover, revocation, and general long-term certificate management.
- The operator of DCCG will communicate the certificate in the chain below which they consider any server certificate as being appropriate for DCCG.

### **(B) Operational and Runtime Considerations:**

- The national backends will communicate a contact point for operational matters if such is not readily evident from the certificate.
- The operator of DCCG will communicate the certificates used by each national backend to all other national backends.