



eHealth Network

Guidelines
on the use of Digital Covid Certificates
in traveller and online booking scenarios

V1.2.0

2021-10-20

eHealth Network

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.

These specifications have no binding character; they have been created to support MS and other parties in providing extra services to the EU Digital Covid Certificate framework.

Adopted by consensus by the eHealth Network on 20 October 2021

CONTENTS

1 Introduction	1
2 Use Cases	1
2.1 Integration of DCCs in Booking Processes	1
2.2 DCC Backup	1
2.3 DCC Exchange	2
2.4 Document Support	2
2.5 DCC Emergency Mode	2
3 Booking and Ticketing Integration	2
3.1 Overview	2
3.2 Principles	3
3.3 Logical Model	4
3.4 Technical Model	4
3.5 Building Blocks	5
3.5.1 Overview	5
3.5.2 Protocol Handler	6
3.5.3 Validation Decorator Structure	6
3.5.4 Validation Service Structure	7
3.6 DCC Exchange Protocol Overview	8
3.7 Validation Modes	8
3.8 Data Structures	9
3.8.1 Initialization QR Code Content	9
3.8.2 Identity Document	11
3.8.3 Validation Initialisation	14
3.8.4 Validation Access Token	15
3.8.5 Result Token	19
3.9 Communication Flow	21
3.9.1 Validation Flow	21
3.9.2 Backchannel Upload	22
3.9.3 Frontend Upload	22
3.9.4 Wallet to Service Provider App Communication	24
3.9.5 Cancel Process	24
3.10 Interfaces	25
3.10.1 Validation Decorator	25

3.10.2 Secure Validation Service	31
4 DCC Export/Import	36
5 DCC Exchange	36
6 Enhancement of Document Support	37
6.1. Overview.....	37
6.2 Enhanced Document Validation	37
7 Traveller Acceptance	38
7.1 Acceptance and Restrictions.....	38
7.2 Categorization of Rule Evaluations	40
8 Emergency Mode	40
8.1 Overview.....	40
8.2 Details View	41
8.3 Anonymous DCC Export.....	41
Appendix A – Example Wallet UX.....	43
Appendix B – UX Example	49
Appendix C – Example Restriction Rule	50

1 Introduction

This document specifies the Digital Covid Certificate (DCC) features applicable to traveller & online booking scenarios (Version 1.2). It provides specifications for the support functions and services applicable to travel booking/check-in scenarios as well as some additional features enhancing the EU DCC digital wallet. DCC Gateway core functionalities and central interfaces remain untouched.

This document is intended to support Member States and other parties in considering *inter alia* how to organise DCC verification services for travellers to demonstrate their compliance with DCC requirements during online check-in.

For travel purposes the use of the previously introduced Businesses Rules¹ adds significant value to the features described in this document. All the Member States and third countries that are the subject of an equivalence decision should be encouraged to prepare and submit their applicable rules to the EU DCC Gateway.

This document sets out the overall technical solution, and its underlying principles and concepts. The detailed technical specifications and implementations related to these guidelines are maintained in GitHub.

The new features described in these guidelines meet the same high levels of security and data privacy as applied in all other guidelines and specifications adopted within the DCC framework.

Please note that these specifications have no binding character, rather they have been created to support MS and other parties in providing extra services to the EU Digital Covid Certificate framework.

2 Use Cases

Version 1.2 of the specifications cover the following use cases:

2.1 Integration of DCCs in Booking Processes

A web service (for instance an airline) wants to validate a DCC during the booking process to evaluate the status of a traveller and possible entry restrictions. For this purpose, the service wants to integrate a validation service which is able to securely process a customer's DCC to return a status of the DCC's validity/acceptance based on the acceptance rules of the country of destination and the time of arrival.

2.2 DCC Backup

During travel the paper or the digital version of a DCC can be lost or damaged. To avoid this, it should be possible to generate a PDF/PNG to backup or print the DCC. A traveller is then able to generate a PDF in the wallet which can be printed or stored as a picture of the DCC.

¹ EU DCC Validation Rules V1.00, 9 June 2021: https://ec.europa.eu/health/sites/default/files/ehealth/docs/eu-dcc_validation-rules_en.pdf

2.3 DCC Exchange

A user wants to present the DCC QR code to NFC readers, to enable future use cases like faster and less error prone DCC verification and semi-automated entry control systems.

2.4 Document Support

Different countries may require different or extra documents in paper, PDF or QR code format which are necessary to enter a country. To simplify the handling of these documents, the wallet shall support the storage of such information, in order to avoid the need for multiple apps during the check-up at the border (for instance, dPLF² + DCC + ticket code).

2.5 DCC Emergency Mode

Occasionally, some QR code anomalies occur during the verification process. To understand these anomalies better and to improve the whole DCC framework, the verifier and the holder of a DCC need to understand precisely what is wrong with the DCC. Additionally, verifiers may want to share anonymized DCCs with the developers for analysis. This feature allows to verify certain attributes of DCCs visually to address technical issues in the verification process. In addition, it enables the authorized user to generate an anonymous copy of the DCC causing problems, so it can be forwarded for further detailed analysis.

3 Booking and Ticketing Integration

3.1 Overview

To validate a DCC during an online booking or check-in process, the service provider must be connected to a trusted validation service. This validation service can receive and validate the DCC from the wallet app, or from the service frontend (eg. the ticketing website). The DCCs are encrypted by a public key of the validation service, and only then transmitted during the validation process. After validation, the service provider's backend gets feedback about validation success or failure. The service provider can then decide whether the check process may continue or not.

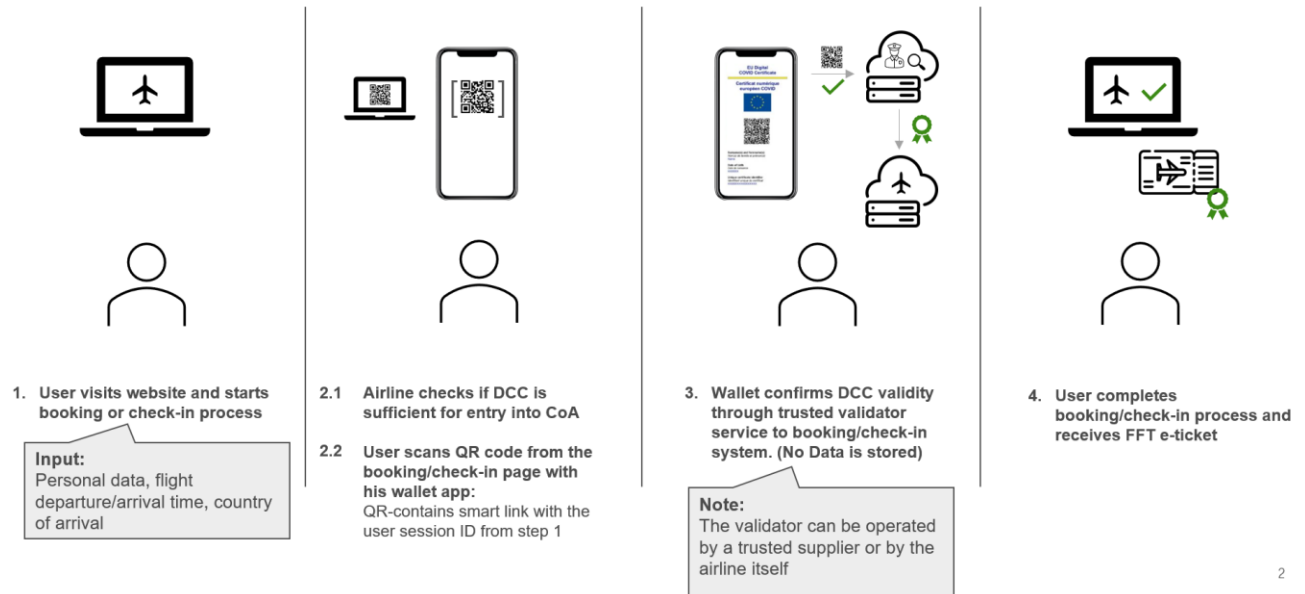
The core of the booking use case is the validation service. This service will require the minimum amount of information in order to validate the authenticity, integrity and validity of the certificate in question.

The transport operators / service providers (e.g., airline, travel company, ticketing agency, football club) operate the validation decorator. The decorator collects all of the information available about the booking process – including the DCC itself – and makes the call to the validation service and handles the response.

The validation service should be operated by an authorised (commercial or public-sector) provider (see also 3.2). Each operator uses one or more dedicated validation services, which can be also configured to check additional rules. How this is organised in each Member State is left to the

² <https://app.euplf.eu/#/>

decision of individual Member States' authorities in accordance with the applicable legal and regulatory framework for accessing and processing the data in question.



2

Notes:

1. The validation service must work in an environment specifically tailored to ensure confidentiality, integrity, availability, and GDPR compliance.
2. Details for the technical implementation and the security of the validation service are available on GitHub.

3.2 Principles

To use the secure validation service there are some principles which must be considered before this validation service is offered or integrated in any booking system:

- DCC information may not be stored (persisted). DCC must be processed within the course of one request, and the data must be discarded after it has been processed.
- A wallet app/client must request the user consent before any calls or transmissions
- The booking/ticketing provider is not allowed to transmit the DCCs over its own servers (except for encrypted routing, e.g. if cross domain sharing is not possible)
- All offered Validation Services to the end user must be monitored by the booking/ticketing provider to avoid data misuse
- If data misuse is detected, the Validation Service from that service provider must be discontinued, and the incident must be immediately reported to the competent authorities
- The validation service only validates the information within the DCCs for the country of arrival, in combination with the Country of Arrival and the date of departure. Any other travel and health related topics are out of scope of the DCC validation service and are the responsibility of the booking/ticketing provider (such as Passenger Locator Forms etc.)
- Any validation service must be covered by current regulations (national and/or EU)

3.3 Logical Model

The logic behind the flow is to enable the wallet app as the central holder of DCCs to “speak” a protocol which can exchange the DCCs securely to a validation service. This can be achieved by a special “DCC Exchange Protocol”, which enables the wallet app to share the DCCs with consent of the user. This protocol is described in the picture below by an example of a boarding pass issuing:

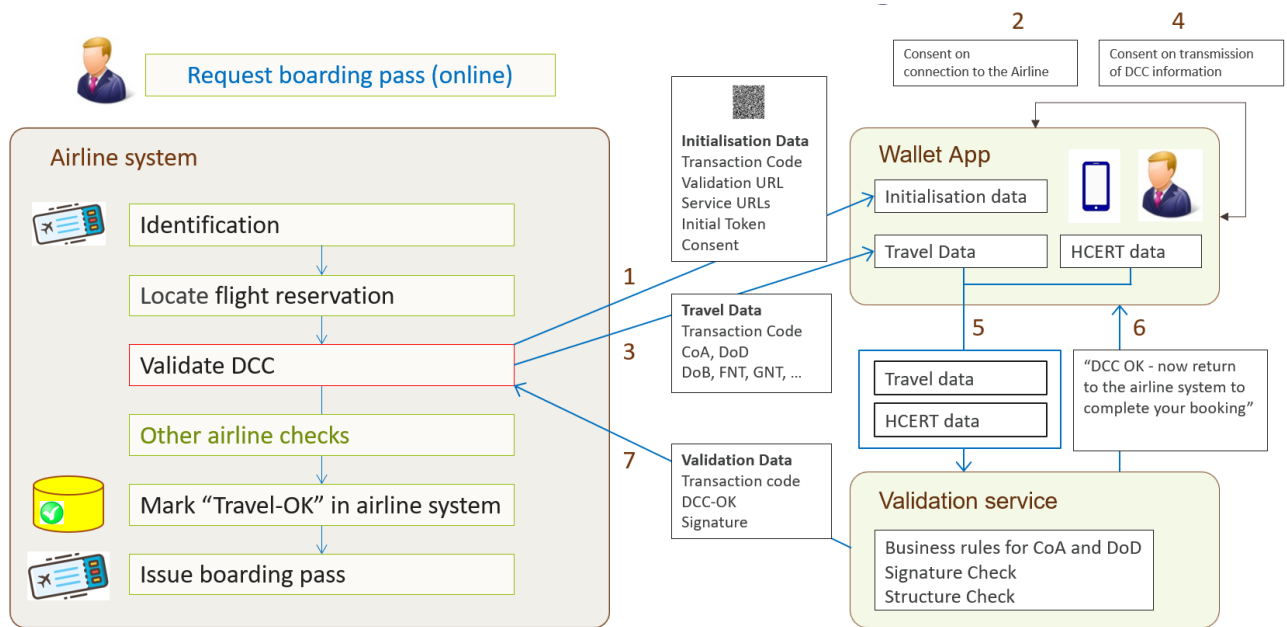


Figure 1 - basic process flow (functional)

First, the user is requesting a boarding pass and does the normal identification for the flight reservation. After the identification, the ticketing system provides a special QR code which contains initial information about the session and the validation process. The user (passenger/DCC holder, etc.) scans this QR with the wallet app [1] (alternatively the web frontend keeps the initialisation private) and exchanges this initialisation information against an access token for the validation service which contains the session information signed by the ticketing system [3]. With this access token, the user can upload his DCC to the validation service [5] and the validation service can compare the information signed by the ticketing system against the provided DCC information [6,7]. The DCC is directly encrypted and transmitted from the user to the validation service, without interference from the ticketing system. If the DCC validation is done, the validation service returns the result of the validation to both Wallet App and ticketing system.

Note: Wallet App means in the figure above can be an App or a Frontend which can handle DCCs. This can be an official Wallet App, a QR Code Reader, a Web Frontend of a Service Provider, or any other app which is able to use the protocol.

3.4 Technical Model

The technical model of the validation process is based on JSON and JWT inspired by OAuth2. JWT was chosen, because it’s a commonly known format to transport and sign JSON data.

The entire flow is triggered by a QR Code which contains Initialisation Information described further in chapter 3.8 (Data Structures). After the initialisation and user consent (user must consent that a connection to the ticketing system is ok), the public key of the user is shared to the ticketing system which uses the key to initialize the validation session at the validation service over a secured mTLS connection. If the initialisation is successful, the ticketing system responds to this with an access token which contains all information regarding the validation, and which is signed by the ticketing system. This token is later sent to the validation service together with the encrypted DCC (encrypted by the public key of the validation service). The validation service can then decrypt the DCC by using its own private key.

After encryption, the plain DCC content is checked against the user signature by using the public key received by the ticketing system. This cross check guarantees that the session of the ticketing system and the validation, was triggered by the same holder of the key pair. The validation result is directly responded to the wallet app, but the ticketing system gets a response later by call back or polling. In an airline system, process flow may look like the figure below:

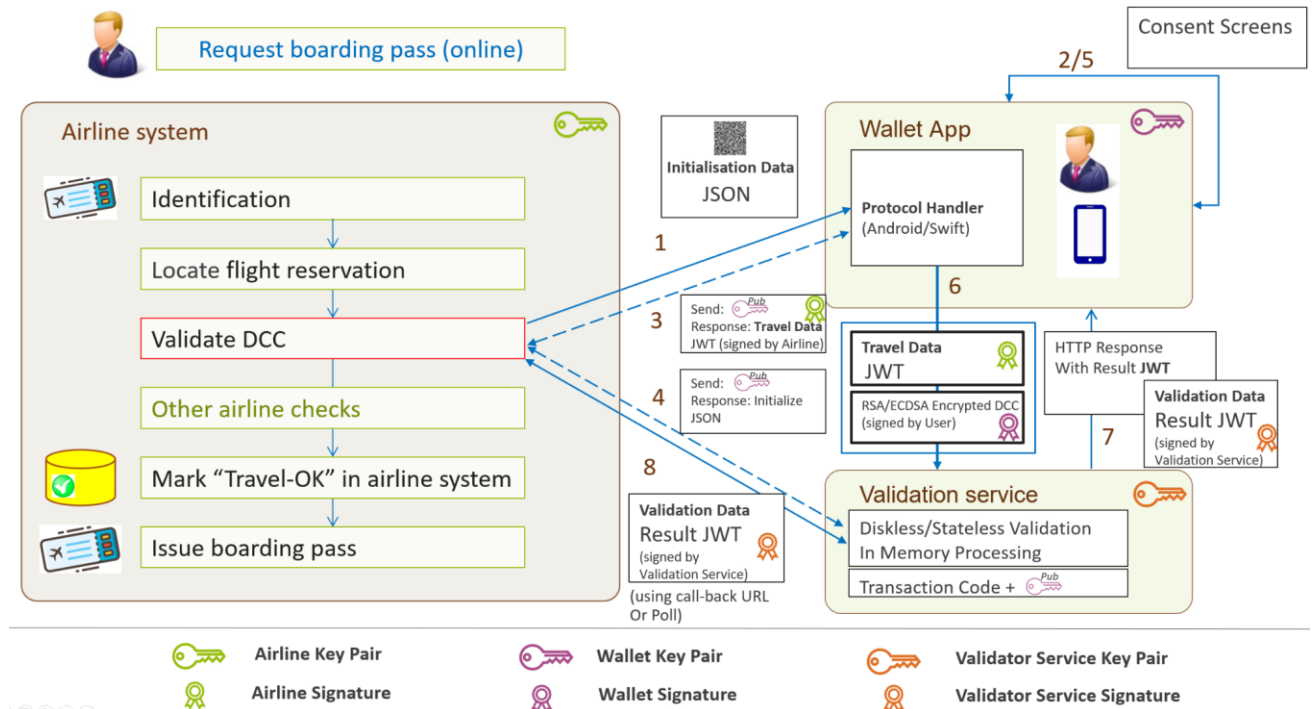


Figure 2- basic process flow (technical)

3.5 Building Blocks

3.5.1 Overview

The main building blocks of the DCC Validation is the Protocol Handler for the Wallet App (or similar), the Validation Decorator, the Validation Service, and the Protocol itself.

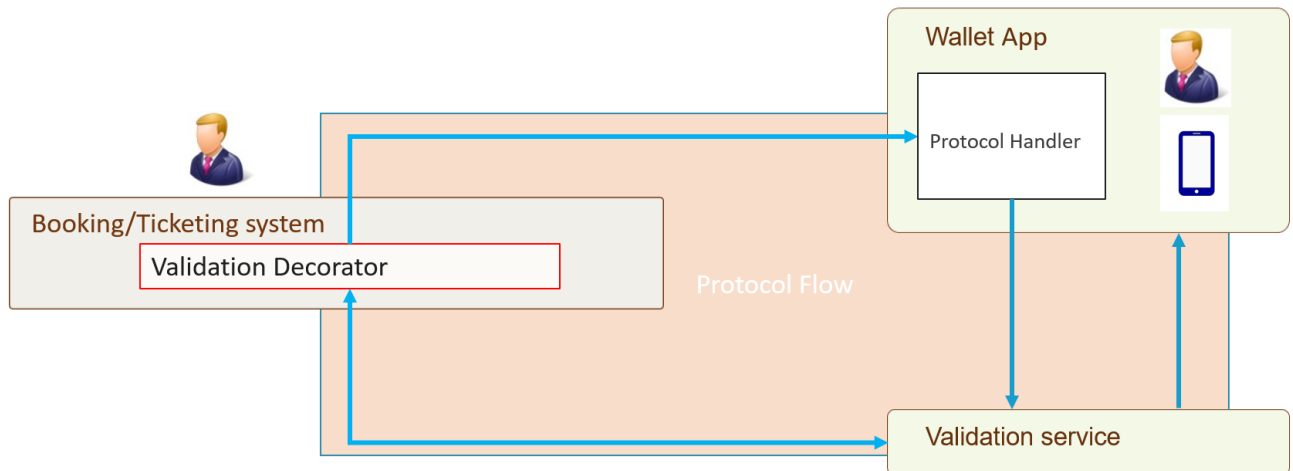


Figure 3- design patterns

The main tasks of the Decorator are to generate access credentials on behalf of the Validation Service, extract information about the current booking process for the wallet app and collect feedback on the validation process. This Decorator is securely connected to the Validation Service and has exclusive access to the booking process (e.g. by sharing the same session cache). Decorator and the secure Validation Service should run under the same domain, but on different servers to encapsulate the DCC validation in a protected environment. The secure Validation Service itself contains the entire verification logic for DCCs and is connected to the national DCC backend.

NOTE: The service provider should have its own instance of the secure validation service to integrate it in the best way in the booking process (one domain, one process flow), but the service itself should be operated by a party which is authorised to process medical data. This can be a central service or authorised hosting suppliers. The setup can vary. However, it must ensure that the medical data is not visible to the service provider during the booking process. Therefore, all key material should be separately maintained to avoid that a ticketing system has access to the medical data.

3.5.2 Protocol Handler

The Protocol Handler as part of the wallet app analyses the scanned input, validates the DCC format and triggers the consent screens. After consent is given, the Protocol Handler provides the DCC to the secure validation service.

3.5.3 Validation Decorator Structure

The Validation Decorator is a service which runs separately from the booking/ticketing system with exclusive access to the booking data. The booking system generates for each passenger and ticket/check-in process a subject, which maps to the communication and to the validation access token within the decorator. Overall, the Validation Decorator handles the generation of validation access tokens, the communication to the validation service instance(s), and the exchange of relevant booking/ticketing data between wallet and the docked system.

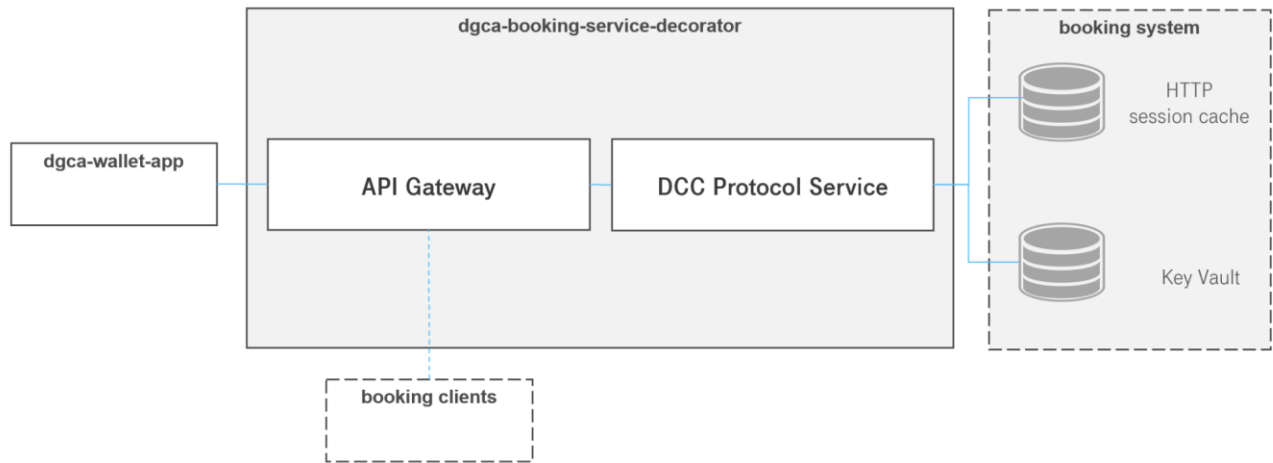


Figure 4 – Validation decorator

3.5.4 Validation Service Structure

The Validation Service has the same logic for the DCC validation as the verifier apps. In addition, it communicates the result of the validation to both ticketing system and DCC wallet holder. The public keys and business rules are necessary for validation; they can be downloaded from the verifier and business rules service (e.g. the MS backend service).

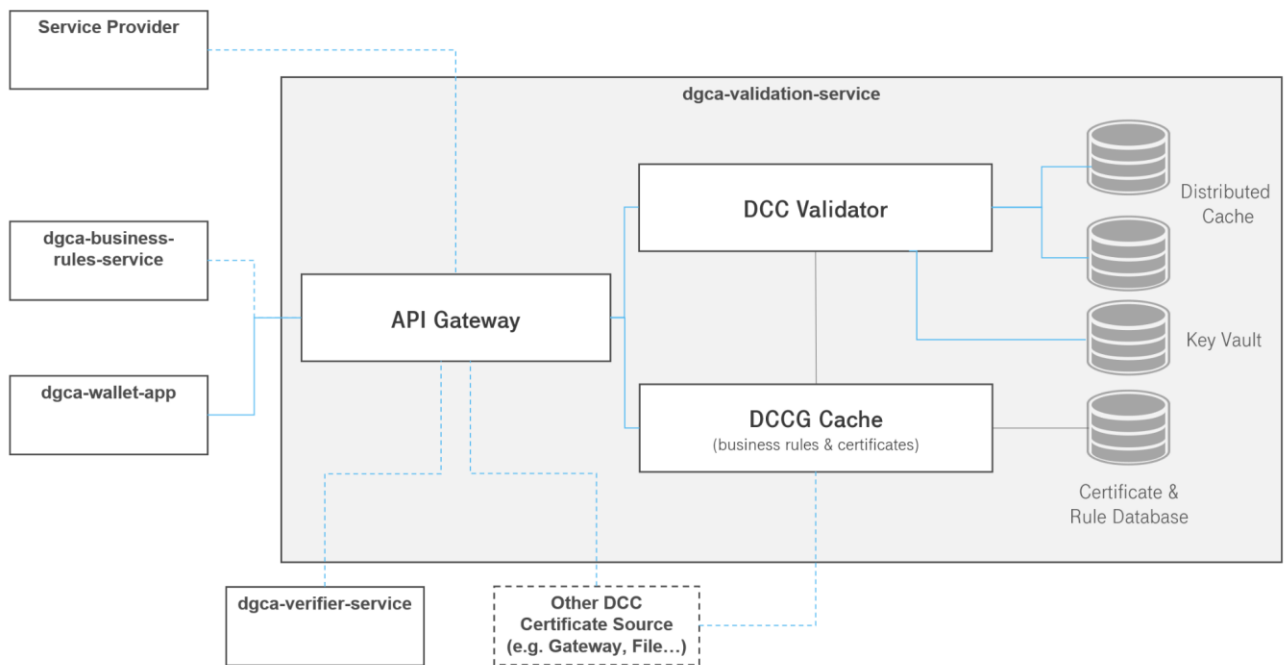
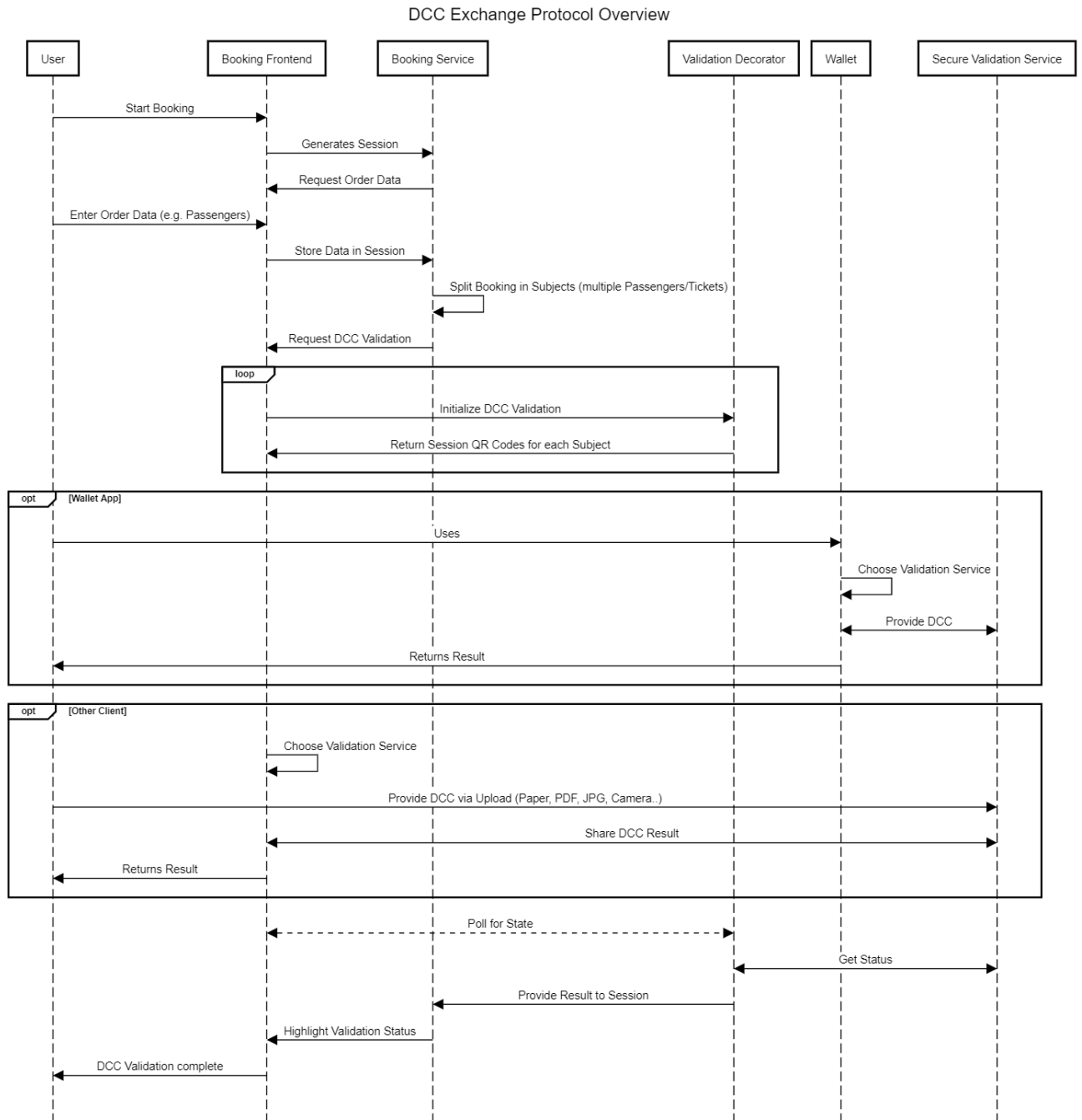


Figure 5 – Validation service

3.6 DCC Exchange Protocol Overview



3.7 Validation Modes

The validation service provides three modes of operation to support multiple users, Structural, Basic and Full.

Structural

The structural mode of the validation service provides the validation of JSON Schema, CBOR structures, data format check-up, value sets and other syntactically/semantically checks to support DCC issuers by checking the generated structure. This mode compares the calculated hash as well to ensure that the signature on the issuer side is calculated in the right way.

Cryptographic

This mode checks a provided DCC by validating the structure and the cryptographic signature against the public keys of the connected countries.

Full

This mode checks a provided DCC structure, cryptographically and against all business rules of the connected parties to given parameters. This mode is the full verification cycle which is also made by the verifier apps.

3.8 Data Structures

3.8.1 Initialization QR Code Content

The session QR code content is returned to establish a connection between the user and the service provider to exchange the DCC. For this purpose, the service provider returns a subject which references the current occurrence/user and the URL where the wallet app can find the associated endpoints, public keys, etc. Before the process is started, the user must trust this information explicitly. The structure is a JSON document rendered as QR code in the frontend.

FIELD	VALUE	DATE TYPE	DESCRIPTION
protocol	“DCCVALIDATION”	String	Type of the requested protocol
protocolVersion	1.0.0	String	SemVer version number
serviceIdentity	e.g., https://serviceprovider/document	String	URL to the service provider identity document
token	e.g. ey....	String	See table below
consent	e.g. “Service provider wants to check your DCC with the purpose of...”	String	Consent text which is shown to the user by the wallet app
privacyUrl	e.g. https://myprivacy	String	A separate privacy statement url with additional data processing information.
subject	e.g. “Booking Nr.1234”	String	Subject of the Request
serviceProvider	e.g. “MyCompany”	String	Company Name

Token

This token is generated to access the Validation Decorator endpoints and contains the information about the booking.

FIELD	VALUE	DATE TYPE	DESCRIPTION
issuer	https://serviceprovider	String	Service provider
iat	1629231259	Int	Seconds since January 1, 1970, "issued at" timestamp
sub	ADEDDDDDDDDDDDD DDD DD	String	Random value which references the service provider's occurrence
exp	27374734	Int	Expiration date of the token in Numbers of seconds since epoch

Example



FIELD	VALUE	DATE TYPE	DESCRIPTION
Id	see table below	String	Id of the Verification Material
type	JsonWebKey2020	String	Type of the Material
controller	URL of service provider	String	URL
publicKeyJwk	see below	JSON Object	JWK Representation as per RFC7517 (Mandatory only for asymmetric encryption/signing, otherwise optional)
verificationMethods	ID of verification Method	Array of String	Optional Verification IDs Array which can be used for referencing other Keys.

JWK Definition:

The JWK must contain a x509 certificate with the public key for signing/encryption in PEM encoding without markers, using ECDSA (P-256 parameters, ES256) or RSA-PSS (PS256) or RSA (RS256) algorithm following the cryptographic requirements of the DCC certificate governance. The used algorithm and the intention of usages must be set in the JWK.

```
{
  "x5c": "MI...", // (PEM)
  "alg": "ES256/PS256/RS256",
  "kid": "...",
  "use": "enc/sig"
}
```

Kid Calculation:

The kid should be calculated as the first 8 bytes of the SHA256 Fingerprint of the X5C content.

ID Values:

All ID values must be prefixed by {serviceproviderurl} and a fragment value, e.g., https://serviceprovider/verificationmethod#AccessTokenSigning-1. If this URL is called, it must return the appropriate section of the identity document.

VALUE	DEFINED IN DOCUMENT OF	DESCRIPTION
AccessTokenSignKey-X	Validation Decorator	Public key of the key pair of the service provider to sign the access token
AccessTokenServiceKey-X	Validation Decorator	Public key of the access token service URL
ValidationServiceKey-X	Validation Decorator	Public key of the used certificate for the validation service URL
ValidationServiceEncKey-X	Validation Service	Public key for encrypting the content send to the validation service
ValidationServiceSignKey X	Validation Service	Public key of the key pair of the validation provider to sign the result token
ValidationServiceEncSchemeKey- {EncryptionScheme}	Validation Service	Verification Method definition of available encryption schemes. Contains no public key. The Encryption Scheme is later used in the Validation Request.
ServiceProviderKey-X	Validation Decorator	Public key of the used certificate of the service provider URL
CancellationServiceKey-X	Validation Decorator	Public key of the used certificate of the cancellation URL
StatusServiceKey-X	Validation Decorator	Public key of the used certificate of the status URL

3.8.2.3 Service

The service section of the document contains endpoints and associated public keys which are used to process a decentralized workflow.

FIELD	VALUE	DATE TYPE	DESCRIPTION
Id	see below	String	ID of the Endpoint (must resolve to a Identity Document)
type	ID Value of table below	String	Type of the Endpoint

serviceEndpoint	https://validationprovider	String	URL of the Endpoint
name	Human Readable Value	String	Human readable Value

ID Values:

All id values have to be prefixed by {serviceproviderurl} and a fragment value, e.g. https://serviceprovider/service#AccessCredentialService. If this URL is called, it must return the appropriate section of the identity document.

VALUE	DEFINED IN DOCUMENT OF	DESCRIPTION
AccessTokenService	Validation Decorator	URL of the Access Token Service Endpoint protected by "AccessTokenServiceKey-x"
ValidationService	Validation Service	URL of the Validation Service protected by "ValidationServiceKey-x"
ServiceProvider	Validation Decorator	URL of the Service Provider protected by "ServiceProviderKey-x"
CancellationService	Validation Decorator	URL of the Cancellation Endpoint protected by "CancellationServiceKey-x"
StatusService	Validation Decorator	URL of the Status Endpoint protected by "StatusServiceKey-x"

Note: If multiple validation services should be available, the ID Value should be enhanced by any kind of Unique identifier. E.g. ValidationService123, ValidationService1 etc. Ensure that the public keys are matching to that pattern. The "type" of the Service must still be "ValidationService".

3.8.3 Validation Initialisation

The validation initialisation response delivers a JSON with a unique subject ID, which identifies the occurrence in the validation service. Next to the unique subject, a public key and an expiration date for the occurrence are attached. The public key must be unique for each subject for maximum data privacy.

FIELD	EXAMPLE VALUE	DATE TYPE	DESCRIPTION
sub	ADEDDDDDDDDDDDDDDDD	String	Hexadecimal-encoded value

exp	1629231259	Int	Number of seconds since Epoch. Expiration time of the subject in the validation service.
aud	https://validationprovider/validate/{subject}	String	Validation URL.
encKey	{"x5c":...MI... }	JSON Object(JWK)	Optional Public Key for Encryption of Validation Service. NOTE: This key should only be used in a Backend-Backend Validation. A wallet should not use the given key provided by the service provider backend, because this one can be intercepted.
signKey	{"x5c":...MI... }	JSON Object(JWK)	Optional Public Key for Signing of the Validation Service Tokens. NOTE: This key should only be used in a Backend-Backend Validation. A wallet should not use the given key provided by the service provider backend, because this one can be

3.8.4 Validation Access Token

3.8.4.1 Overview

To allow a validation process, the validation access token is generated by the service provider. The token follows the format of the JWT definition (RFC7519) and contains details about the necessary validation. The token is signed with the service provider's private key. The associated public key is known by the validation provider and can be selected by the kid.

3.8.4.2 Header

FIELD	VALUE	DATE TYPE	DESCRIPTION
typ	JWT	String	-
alg	ES256/PS256 /RS256	String	-

kid	AccessTokenSigningKey-X KID	String	Used key for signing. This key must be presented in the downloaded identity document.
------------	-----------------------------	--------	---

3.8.4.3 Payload

FIELD	EXAMPLE VALUE	DATE TYPE	DESCRIPTION
jti	23fd34f8dfff	String	Token Identifier.
iss	https://serviceprovider	String	Service Provider (id of identity document)
iat	1232344	Int	Seconds since epoch
sub	ADEDDDDDDDDDDDDDDDDDD	String	Value of the Initialisation
aud	https://validationprovider/validate/{subject}	String	Value of the Initialisation (must match to service endpoint "ValidationService")
exp	1475878357	Int	Seconds since epoch
t	See table below	int	Type of Validation
v	1.0	String	semVer Version of Validation
vc	See Table below	JSON	Validation Conditions (optional, depending on Type = Full)

Type Values

TYPE	VALUE	DESCRIPTION
Structure	0	Validates just the Content of the DCC (Schema, Values, CBOR Structure)
Cryptographic	1	Structure Validation + Signature Validation
Full	2	Structure Validation + Cryptographic + Business Rule Check (condition structure necessary necessary)

Condition Structure

The validation condition structure is embedded in the validation access token to fulfil two things:

- The validation service knows the selected conditions by the service provider/validation service user
- The wallet app can select an appropriate certificate for the user with reference to the conditions

For DCC validation the list contains the following items:

FIELD	EXAMPLE VALUE	MANDATORY FOR TYPE	DATE TYPE	DESCRIPTION
hash	SHA256	0	String	Hash of the DCC Not applicable for Type 1,2
lang	en-en	0,1,2	String	Selected language
fnt	DCERVENKOVA<PANKLOV A	1,2	String	ICAO 9303 transliterated Surname
gnt	JIRINA<MARIA<ALENA	1,2	String	ICAO 9303 transliterated given name
dob	1979-04-14 1901-08 1950	1,2	String	Date of birth
coa	NL	2	String	Country of Arrival

FIELD	EXAMPLE VALUE	MANDATORY FOR TYPE	DATE TYPE	DESCRIPTION
cod	DE	2	String	Country of Departure
roa	AW	2	String	Region of Arrival (ISO 3166-2 without Country)
rod	BW	2	String	Region of Departure (ISO 3166-2 without Country)
type	[v,t,r,tp,tr]	0,1,2	Array of String	Acceptable Type of DCC v=vaccination t=test
				r=recovery tr=RAT Test tp=PCR Test
category	e.g. "Inter-Flight", "Concert", "Domestic", "Massevent>1000" etc. Default: "Standard"	2	Array String	Optional Category which shall be reflected in the Validation by additional rules/logic. If null, Standard Business Rule Check will apply.
validationClock	2021-01-29T12:00:00+01:00	1,2	String	Date where the DCC must be validatable.
validfrom	2021-01-29T12:00:00+01:00	0,1,2	String	DCC must be valid from this date. (ISO8601 with offset)
validTo	2021-01-30T12:00:00+08:00	0,1,2	String	DCC must be valid minimum to this date. (ISO8601 with offset)

Token Replay

The token should not be used multiple times. The API should blacklist the token to the point of the expiration timestamp.

3.8.5 Result Token

3.8.5.1 Overview

The result token is generated from the validation service to acknowledge the validation associated with one subject. The booking system can use it to validate/present the result to the frontend. Within the result token is a confirmation token which can be used for archiving the validation result or for linking the validation result to the issued ticket by using the included unique jti identifier (see 3.8.4.3).

3.8.5.2 Header

FIELD	VALUE	DATE TYPE	DESCRIPTION
typ	JWT	String	-
alg	ES256/PS256/RS256	String	-
kid	ValidationServiceSignKey-X KID	String	Used key for signing

3.8.5.3 Payload

The result contains a table with all evaluated business rules, checks and their results. For data privacy reasons, the expected and current values are not included. The result contains a signed token as well of the DCC validation occurrence which can be used by the service provider to evaluate if a valid process for the criteria was done.

FIELD	EXAMPLE VALUE	DATE TYPE	DESCRIPTION
iss	https://serviceprovider	string	Issuer URL
iat	1232344334	int	Seconds since epoch
exp	1232344443	int	Seconds since epoch
category	e.g. Standard	Array of String	Category of Confirmation
sub	ADEDDDDDDDDDDDDDDDDDD	string	Value of the access token
result	CHK NOK OK	string	Final result of the evaluation. OK=Passed, NOK=Fail, CHK=Cross Check (OPEN)
results	See "Results" below	Array of Objects	Table of validation results

FIELD	EXAMPLE VALUE	DATE TYPE	DESCRIPTION
confirmation	See "Confirmation" below	string	JWT string

Confirmation

The confirmation has the same header as the result, is signed with the same key, but contains just limited information about the original request.

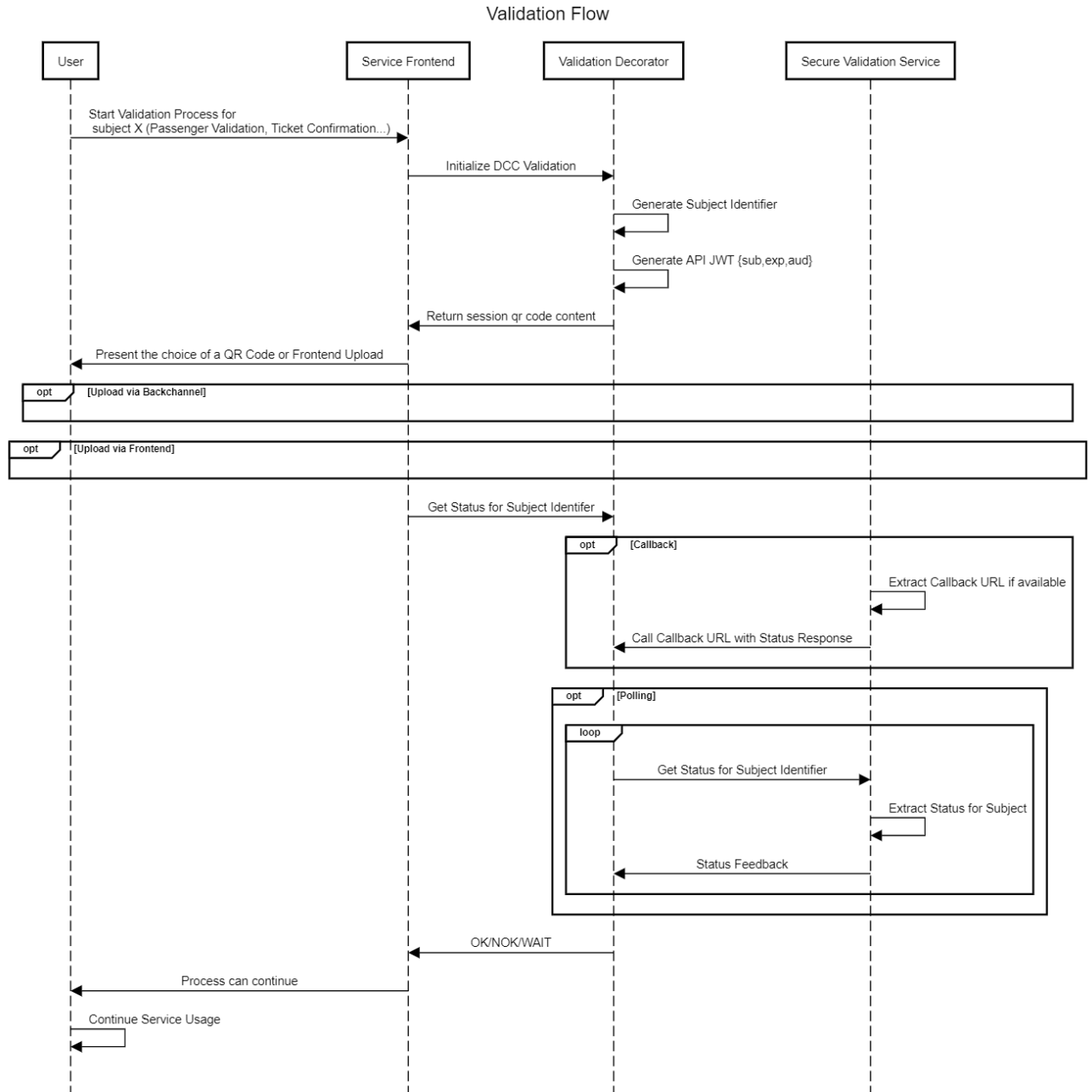
FIELD	VALUE	DATE TYPE	DESCRIPTION
jti	GUID	String	Unique Identifier of the confirmation token
sub	ADEDDDDDDDDDDDDDDDD	string	Value of the access token
iat	1475878357	int	Seconds since epoch
exp	1475878357	int	Seconds since epoch
category	e.g. Standard	String	Category of Confirmation
result	CHK NOK OK	string	Final result of the evaluation. OK=Passed, NOK=Fail, CHK=Cross Check (OPEN)

Results

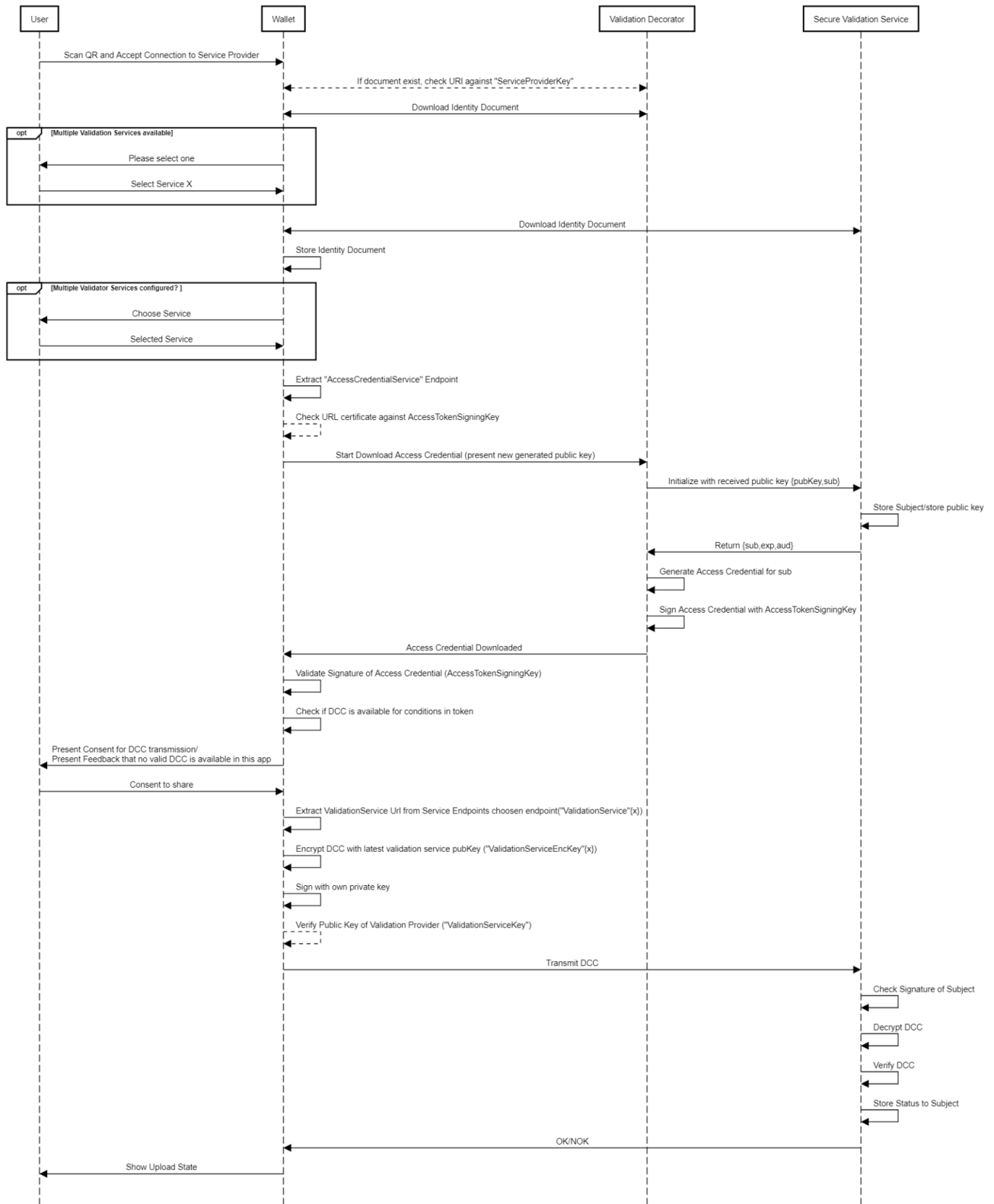
FIELD	VALUE	DATE TYPE	DESCRIPTION
identifier	e.g. VR-0001, CBOR, SIGNATURE etc.	String	Identifier of the check
result	OPEN (CHK) FAILED (NOK) PASSED (OK)	String	Result of the check
type	Technical Check Issuer Invalidation Destination Acceptance Traveller Acceptance	String	Type of the check
details	Any string	String	Description of the checkup

3.9 Communication Flow

3.9.1 Validation Flow

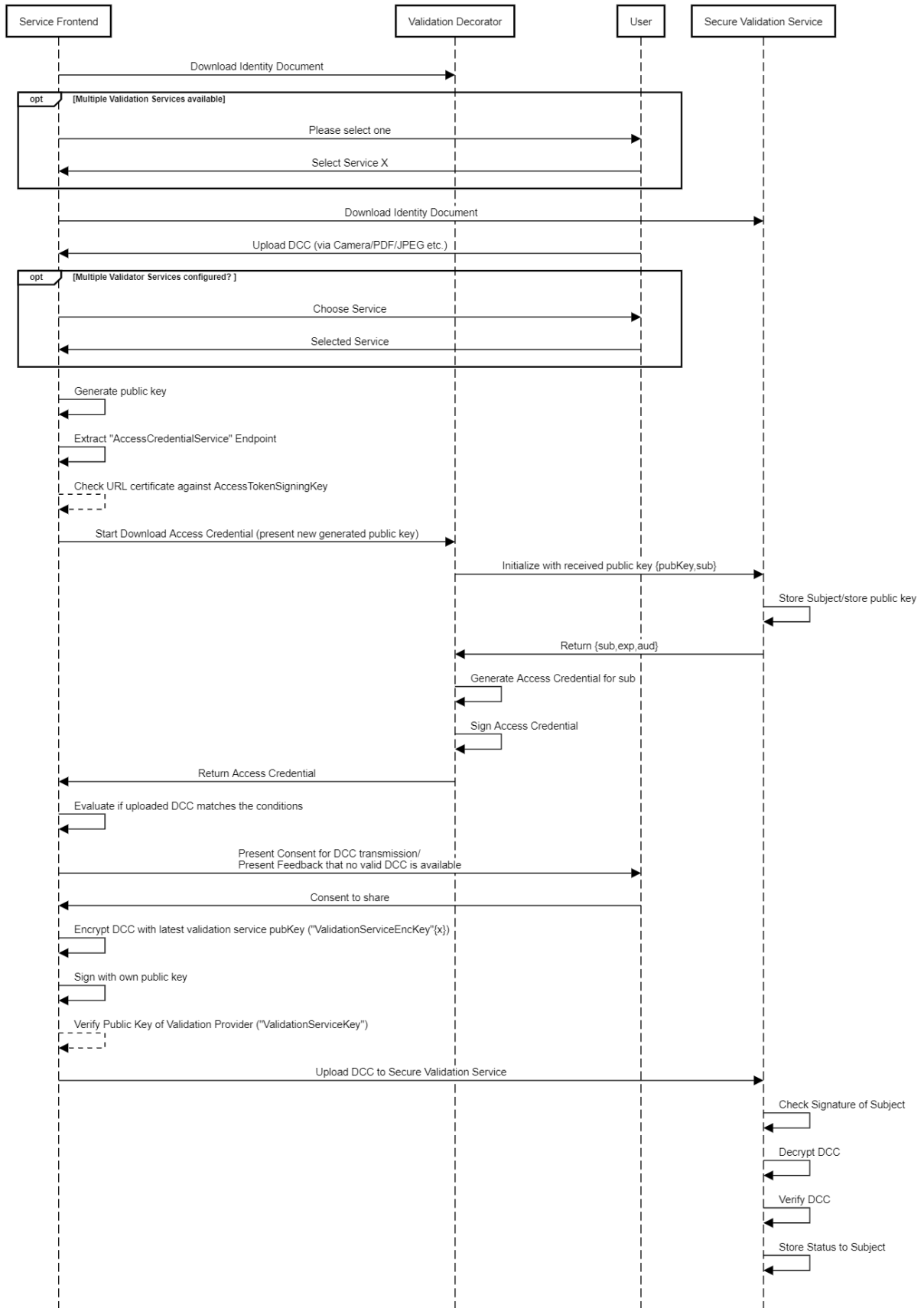


3.9.2 Backchannel Upload



3.9.3 Frontend Upload

The frontend upload flow supports the usage of a dcc in a paper or any other digital version without a wallet app. This flow can be realized in a mobile client or in the web browser.



3.9.4 Wallet to Service Provider App Communication

To share DCCs between the service provider app and the wallet app, the service provider app should use the files which are exported by the wallet app. There can be other mechanisms used like the Intent Mechanism on Android or Deep/Universal links, but these mechanisms are different from device to device. Independent of the choice of the technique, there are two main modes which can be integrated in the service provider app:

- DCC Sharing between Apps
- Sharing of the Flow Initialisation

The sharing of the flow initialization can be realized over sharing the initialization URL including the subject to trigger the wallet app over a custom URL scheme or by sharing the initialization JSON base64 encoded. If the url variant is used, a key should be provided for the Authorization Header otherwise the call is made without it. Before the call, the wallet app should ask for user consent.

Pattern:

URLScheme?initialize={url}&auth={key}

Or

URLScheme?initialize={base64EncodedJson}

Example:

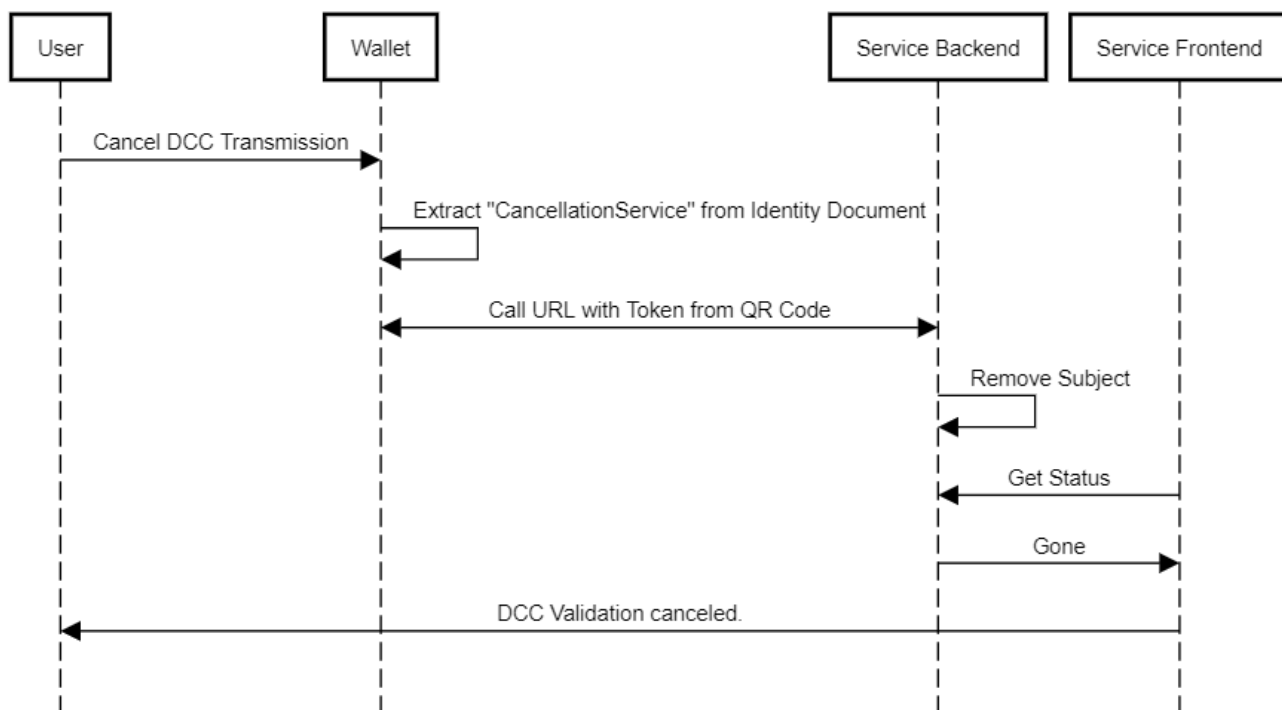
A URL scheme for the wallet app is registered as `dcc://validate?initialize={URL}&auth={key}`. The Service provider app receive a url + access key from the backend and calls on the device:

```
dcc://validate?initialize=https%3A%2F%2Fserviceprovider%2Finitialize%3Fsubject%3D12334&auth=3jffff3493403
```

The wallet app receives the initialization QR and can start with the flow.

3.9.5 Cancel Process

To reject a DCC transmission request, the wallet app extracts “CancellationService” from the Identity Document and calls the link with the provided token. The transmission is then aborted.



3.10 Interfaces

3.10.1 Validation Decorator

3.10.1.1 QR Code delivery Endpoint

3.10.1.1.1 Endpoint Structure

The endpoint for receiving the initialisation QR Code, should provide a simple protected endpoint which is called by the service providers app/frontend:

GET /initialize/{subject}

To protect the endpoint an existing http session cookie can be used, client authentication, access token, or any other kind of security mechanism of the booking/ticketing system. The call itself executes in the validation decorator the logic for extracting the current occurrence information.

3.10.1.1.2 Return Codes

CODE	DESCRIPTION	RESPONSE BODY
200	OK	See Response Body
401	Unauthorized	-

3.10.1.1.3 Request Header

HEADER	VALUE	DATE TYPE	DESCRIPTION
Accept	application/json	String	Accept Header
Content-Type	application/json	String	Content Type
X-Version	1.0	String	Version of Content (Default)

3.10.1.1.4 Path Variables

HEADER	VALUE	DATE TYPE	DESCRIPTION
subject	subject ID	String	subject ID of the current subject for extracting the data of the session

3.10.1.1.5 Response Body

The response body contains a Session QR Code structure as described in the Data Structure section 3.8.1

3.10.1.2 Identity Document Endpoint

3.10.1.2.1 Endpoint Structure

The identity document endpoint delivers a json description of public keys and service endpoints. This endpoint should be publicly available.

GET /identity

The endpoint must support variables to get sub sections of the document. The root elements of the document are path elements, and the IDs of the subsections are used in the fragment:

GET /identity/{rootElement}/{type}#{id}

The fragments itself are just interpretable by clients, but all other path variables should deliver just a subset of the identity document. For instance, /service should deliver an identity document where the service array is just filled.

Examples:

GET

https://serviceprovider/identity/service/AccessCredentialService#AccessTokenService-1

GET

https://serviceprovider/identity/verificationMethod/ValidationServiceSignKey#ValidationServiceSignKey-1

3.10.1.2.2 Return Codes

CODE	DESCRIPTION	RESPONSE BODY
200	OK	See Response Body
404	Not found	-

3.10.1.2.3 Request Header

HEADER	VALUE	MANDATORY	DATE TYPE	DESCRIPTION
Accept	application/json	Yes	String	Accept Header
X-Version	1.0	Yes	String	Version of Content (Default)

3.10.1.2.4 Response Body

The identity document endpoint provides a json in the structure defined in the data structures section of Identity Document.

3.10.1.3 Access Token Endpoint3.10.1.3.1 Endpoint Structure

The access token endpoint is triggered from the user after the consent to the transfer process with a public key of the user and the access token which was received by the qr code session content.

POST /token

The endpoint returns an access token for the validation service which contains the information of the booking session.

3.10.1.3.2 Return Codes

CODE	DESCRIPTION	RESPONSE BODY
200	OK	See Response Body
401	Unauthorized, if no access token was provided	-
410	Systems found no data to the related Subject.	-

3.10.1.3.3 Request Header

HEADER	VALUE	MANDATORY	DATE TYPE	DESCRIPTION
Authorization	JWT	Yes	String	Token from the QR Session Content
Accept	application/ jwt	Yes	String	Accept Header
Content-Type	application/ json	Yes	String	Content Type
X-Version	1.0	Yes	String	Version of Content (Default)

3.10.1.3.4 Request Body

FIELD	VALUE	MANDATORY	DATE TYPE	DESCRIPTION
service	e.g. Validator Service	Yes	String	chosen Validator Service from Identity Document
pubKey	e.g. base64 string	Yes	String	The request body contains the public ECDSA (P-256)/RSA-PSS key generated by the service frontend in plain text (x.509 PEM).

3.10.1.3.5 Response Body

See data structure section 3.8.4

3.10.1.3.6 Response Header

HEADER	VALUE	MANDATORY	DATE TYPE
X-Nonce	16 byte Nonce	Yes	String

3.10.1.4 Reject Callback Endpoint3.10.1.4.1 Endpoint Structure

The reject callback endpoint cancels a dcc validation.

GET /reject3.10.1.4.2 Return Codes

CODE	DESCRIPTION	RESPONSE BODY
200	OK	-
401	Unauthorized, if no access token is provided	-

3.10.1.4.3 Request Header

HEADER	VALUE	MANDATORY	DATE TYPE	DESCRIPTION
Authorization	JWT	Yes	String	Token from the QR Session Content
Accept	application /json	Yes	String	Accept Header
X-Version	1.0	Yes	String	Version of Content (Default)

3.10.1.5 Validation Status Endpoint3.10.1.5.1 Endpoint Structure

The validation status endpoint provides the validation result of a subject.

GET /status

Note: This endpoint is just reachable if an access token of the initialisation is available.

3.10.1.5.2 Return Codes

CODE	DESCRIPTION	RESPONSE BODY
200	OK	See Response Body
204	No content, wait for status	-
401	Unauthorized, if no access token was provided	-
410	Gone. Subject does not exist anymore (TTL expired).	-

3.10.1.5.3 Request Header

HEADER	VALUE	MANDATORY	DATE TYPE	DESCRIPTION
Authorization	JWT	Yes	String	Token from the QR Session Content
Accept	application/json	Yes	String	Accept Header
X-Version	1.0	Yes	String	Version of Content (Default)

3.10.1.5.4 Response Body

The request body contains the structure described in data structures section 3.8.5

3.10.1.6 Callback Status Endpoint3.10.1.6.1 Endpoint Structure

The optional callback endpoint receives the validation result to a subject

PUT /callback/{subject}3.10.1.6.2 Return Codes

CODE	DESCRIPTION	RESPONSE BODY
200	OK	-
401	Unauthorized, if Result Token was not correctly signed by the validation service.	-
410	Gone. Subject does not exist anymore.	-

3.10.1.6.3 Request Header

HEADER	VALUE	MANDATORY	DATE TYPE	DESCRIPTION
Content-Type	application/jwt	Yes	String	Content Type
X-Version	1.0	Yes	String	Version of Content (Default)

3.10.1.6.4 Request Body

The request body is a JWT described in 3.8.5

3.10.2 Secure Validation Service

3.10.2.1 Identity Document Endpoint

3.10.2.1.1 Endpoint Structure

The identity document endpoint delivers a json description of public keys and service endpoints. This endpoint should be publicly available.

GET /identity

The endpoint must support variables to get sub sections of the document. The root elements of the document are path elements, and the IDs of the subsections are used in the fragment:

GET /identity/{rootElement}/{type}#{id}

The fragments itself are just interpretable by clients, but all other path variables should deliver just a subset of the identity document. For instance, /service should deliver an identity document where the service array is just filled.

Examples:

GET

https://serviceprovider/identity/service/AccessCredentialService#AccessTokenService-1

GET https://serviceprovider/identity/verificationMethod/ValidationServiceSignKey#ValidationServiceSignKey-1

3.10.2.1.2 Return Codes

CODE	DESCRIPTION	RESPONSE BODY
200	OK	See Response Body
404	Not found	-

3.10.2.1.3 Request Header

HEADER	VALUE	MANDATORY	DATE TYPE	DESCRIPTION
Accept	application/json	Yes	String	Accept Header
X-Version	1.0	Yes	String	Version of Content (Default)

3.10.2.1.4 Response Body

The identity document endpoint provides a json in the structure defined in data structures section 3.8.2

3.10.2.2 DCC Validation Initialisation Endpoint

3.10.2.2.1 Endpoint Structure

The validation initialisation endpoint delivers to a subject and a public key, an initialisation information to indicate to which audience the DCC has to be delivered and when this audience expires. Within this lifetime the validation is able to receive and validate a DCC for this subject.

PUT /initialize/{subject}

3.10.2.2.2 Return Codes

CODE	DESCRIPTION	RESPONSE BODY
201	Created	See Response Body
401	Unauthorized	-
400	Bad Request	-

3.10.2.2.3 Request Body

The request body is a JSON structure which defines the user generated pubkey, the used key type and the subject.

FIELD	VALUE	DATE TYPE	DESCRIPTION
pubKey	ECDSA/RSA-PSS Key	String	Public Key generated by the user/wallet (x.509 PEM)
alg	ES256/PS256 /RS256	String	Use Crypto Method according to JWT definitions in RFC7518 ²
callback	e.g. https://...	String	Optional Callback URL
nonce	Random Byte Array	String	Base64 encoded, random 16 bytes long array

3.10.2.2.4 Request Header

HEADER	VALUE	MANDATORY	DATE TYPE	DESCRIPTION
Accept	application/json	Yes	String	Accept Header
Content-Type	application/json	Yes	String	Content Type
X-Version	1.0	Yes	String	Version of Content (Default)
X-Crypto-Enc	true/false	No	Boolean	Delivers an Encryption JWK in Response (Optional)
X-Crypto-Sign	true/false	No	Boolean	Delivers a Signing JWK in Response (Optional)

3.10.2.2.5 Response Body

The request body contains the structure described in data structures section 3.8.3

3.10.2.3 DCC Validation Status Endpoint

3.10.2.3.1 Endpoint Structure

The validation status endpoint provides the validation result of a subject.

GET /status/{subject}

Note: This endpoint is just reachable over a private connection of the Validation Decorator³.

³ <https://datatracker.ietf.org/doc/html/rfc7518#section-3.1>

3.10.2.3.2 Return Codes

CODE	DESCRIPTION	RESPONSE BODY
200	OK	See Response Body
204	No content, wait for status	-
401	Unauthorized	-
410	Gone. Subject does not exist anymore.	-

3.10.2.3.3 Request Header

HEADER	VALUE	MANDATORY	DATE TYPE	DESCRIPTION
Accept	application/json	Yes	String	Accept Header
X-Version	1.0	Yes	String	Version of Content (Default)

3.10.2.3.4 Response Body

The request body contains the structure described in data structures section 3.8.5

3.10.2.4 DCC Provision Endpoint3.10.2.4.1 Endpoint Structure

The provision endpoint is the public endpoint where DCCs can be provided for a subject. The endpoint receives the encrypted DCC, validates it and creates the result for the subject.

POST /validate/{subject}

Note: The endpoint accepts no validations without a valid access token generated by the Validation Decorator.

3.10.2.4.2 Return Codes

CODE	DESCRIPTION	RESPONSE BODY
200	OK	See Response Body
400	Bad Request, content of the provided data is malformed.	-
401	Unauthorized	-
410	Gone. Subject does not exist anymore.	-
422	Unprocessable Entity. Wrong Signature of the Subject, Wrong Encryption or any other problem with the encoding.	-

3.10.2.4.3 Request Header

HEADER	VALUE	MANDATORY	DATE TYPE	DESCRIPTION
Authorization	JWT	Yes	String	Validation Access Token (see Data Structures)
Accept	application/json	Yes	String	Accept Header
X-Version	1.0	Yes	String	Version

3.10.2.4.4 Request Body

The request body is a JSON contains an encrypted structure and a signature of the uploader.

FIELD	VALUE	DATE TYPE	DESCRIPTION
kid	e.g. 239348fdfff	String	Used kid for encryption.
dcc	Encrypted string in base64 encoding	String	Encrypted DCC according to encScheme. Input is the HCert base45 string.

FIELD	VALUE	DATE TYPE	DESCRIPTION
sig	ECDSA/RSA signature	String	User Signature of the unencrypted dcc content. The Validation Service checks it against the public key transmitted during the initialisation. Proofs that the sender which initializes the booking knows the transmitted content.
sigAlg	e.g. SHA256withECDSA	String	Used Signature Algorithm
encScheme	e.g. CMS, Custom Scheme	String	Used scheme of encryption.
encKey	e.g. Encrypted AES256 Key	String	Optional, if no CMS or no Encryption Scheme with built in AES is used(e.g. X963SHA256AESGCM). The key must be randomly created and encrypted by ECDSA/RSA OEAP. Keylength must be minimum 32 bytes.

NOTE: The detailed list of supported signature Algorithms and Encryption schemes must be collected on github to align the cross-platform schemes in the most effective way.

3.10.2.4.5 Response Body

The request body contains the structure described in data structures section 3.8.5.

4 DCC Export/Import

During travel, the paper version of the dcc can be lost or damaged, or the smartphone can be broken. To avoid problems during the travel, the wallet app should be enhanced with an export/import function to create and load PDF/JPEGs of DCCs. This allows the user to backup and restore the DCC.

5 DCC Exchange

To exchange data between devices for the purpose of verification and sharing, NFC will be integrated into the wallet and the verifier app. This should allow us to verify DCCs directly, Smartphone to Smartphone, Smartphone to NFC Reader or exchange wallet items between smartphones. For this purpose, the host card emulation technology is used.

Note: The Host Card Emulation technology is currently not available for iOS Devices. iOS supports only reading NFC Tags.

6 Enhancement of Document Support

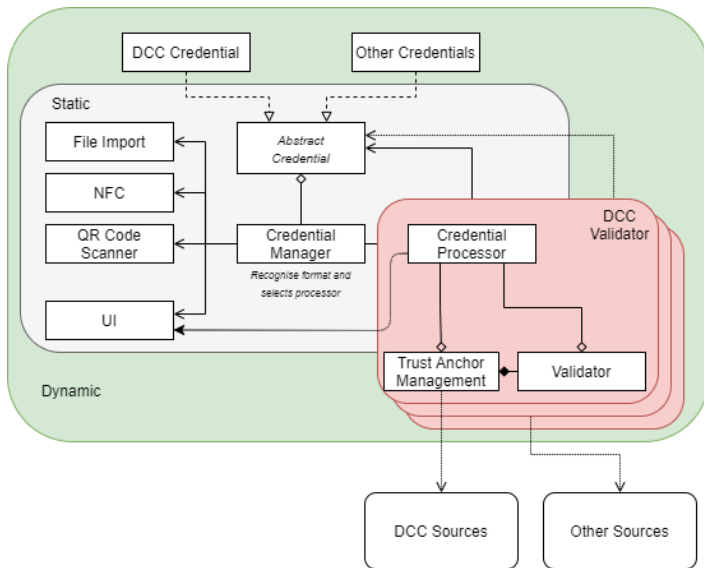
6.1. Overview

The wallet app should support other QR Codes/Documents than the DCC to improve the booking processes. This could be other QR Codes (e.g. Codes from other National Authorities) or PDF files (e.g. Passenger Locator Forms). The wallet should allow the user to manually create entries where this information can be stored. All this information must be shareable as well over the sharing interfaces (e.g. NFC or Import/Export functions)



6.2 Enhanced Document Validation

To support in a later version of the wallet/verifier apps the validation of different formats itself, the app should be prepared for different “Trust Anchors”. This trust anchor support should be realized in the future over a common software pattern within the app architecture, called “Visitor” Pattern. The visitor pattern prepares the apps for “Visitors” in the sense of validations and different formats by providing an interface which can accept an abstracted credential class. This ensures that the app contains a fixed process, but a flexible validation part. An abstract software pattern can be declared as in the following picture:



7 Traveller Acceptance

7.1 Acceptance and Restrictions

Currently business rules have the limited scope of checking whether the (medical) content of a DCC complies with certain requirements and evaluate “DCC Acceptance” in a CoA. To give the user more useful information regarding “Traveller Acceptance” in a CoA, answers to the following two questions need to be encoded within business rules provided by a CoA::

1. Is a valid/accepted DCC sufficient for the CoA to accept a traveller?
2. If yes(1), should the traveller anticipate any sort of restrictions in the CoA (for example quarantine)?

To answer this question in an automatic way is essential to give the traveller a clearer expectation of the destination country.

Example:

	DCC Vaccination	DCC Recovery	DCC NAAT/PCR	DCC RAT
Accept a traveller with valid DCC?	Yes	Yes	Yes	No
Possible restrictions within the country?	Yes	Yes	Yes	

The acceptance above can be expressed by general rules in CertLogic to check for a vaccination entry:

```
{
  "if": [
    {
      "var": "payload.v.0"
    },
    true,
    false
  ]
}
```

This rule can be created with a description: "Travellers are not allowed to enter with a vaccination certificate".

To express the restrictions the logic can be expressed like this:

```
{
  "if": [
    {
      "var": "payload.v.0"
    },
    {
      "+": ["+", "+"]
    },
    true
  ]
}
```

The rule will be failing if the vaccination entry exists and the verifier of the BR gets highlighted that there is a restriction.

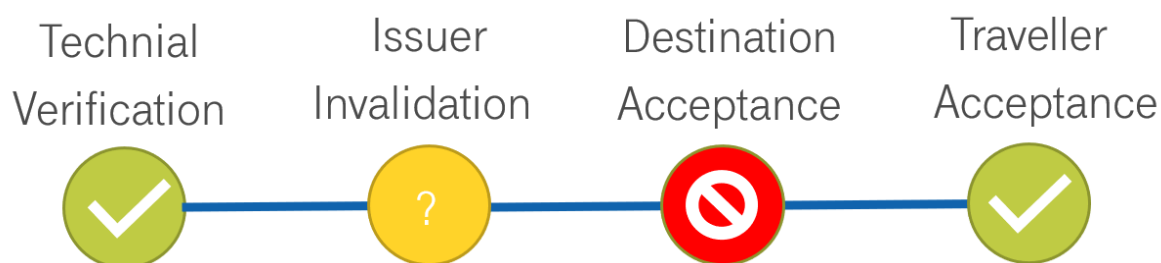
A complete example of the rule can be found in the appendix.

7.2 Categorization of Rule Evaluations

For a better visualization and highlighting of verification results, the verifier app will be enhanced by a step by step visualization of the verification process. The process will be highlighted for the following phases:

PHASE	RESULT OPTIONS	DESCRIPTION
Technical Verification	G/R	CBOR, Signature, Expiration etc.
Issuer Invalidation	G/R/Y	Rule Type "Invalidation" (for all types)
Destination Acceptance (V/T/R)	G/R/Y	All rules with type "Acceptance" and without "certificate type" general.
Traveller Acceptance	G/R/Y	Just Acceptance Rules starting with certificate type "General" respected in the result

The results are displayed in the app in a phase view as overview and grouped in table:



The final result of each validation phase follows the priority of Red, Orange or Green as defined in the business rules definition (excepting technical checks). A visualization of these results should follow the same rules of success, failed or open state (green, yellow, red). Within a phase result, one "OPEN" rule results in yellow and one "FAILED" rule in red as defined by the business rule definitions. If the invalidation result is "FAILED" the app should show red. The destination and traveller acceptance rules should be highlighted as yellow in the case of fail or open.

8 Emergency Mode

8.1 Overview

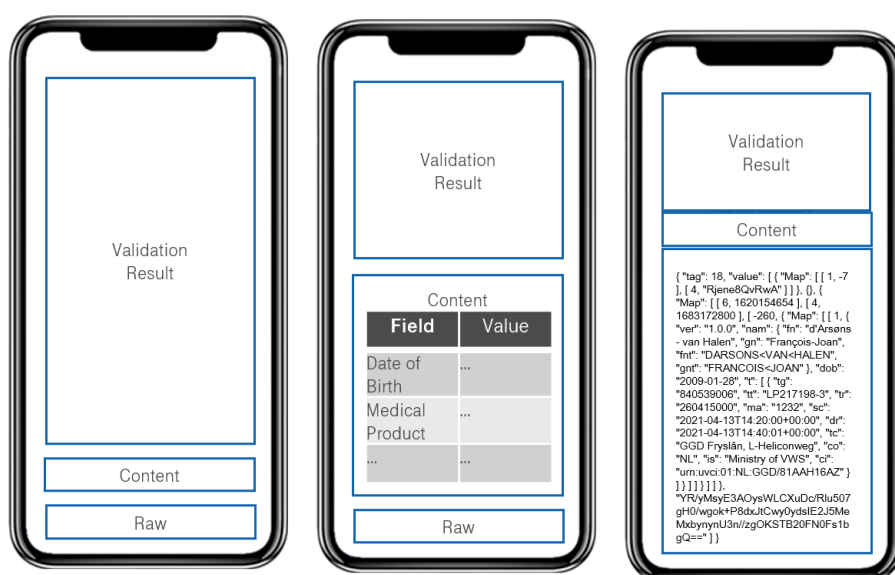
The DCC validation between various states can have from time to time some side effects which result in QR Codes anomalies and trouble at the verification process. The analysis of these anomalies is very hard for the person who is verifying and as well for the person presenting the QR code. To improve the process, a sharing and debug interface for data analysis should be implemented which respects the data protection of the holder. For this reason, the sharing interface must be restricted to authorized users and the holder must have the choice which data of the QR code shall be available for the analysis AND if he wants to share data or not. This can be realized by integrating a sharing function in the wallet and/or verifier app which can be authorized by the verifying institution.

8.2 Details View

To support the verifier more in his decisions, the debug mode should contain detailed information of all data which are scanned and the results. This feature should be enabled only in emergency cases. The details view for showing this information should show all DCC contents in human readable format. To support the verifier in the best way the verifier app should present:

1. Normal Validation Result
2. Entry of the QR (V/R/T) in a table format (all fields visible)
3. Complete Certificate formatted as JSON

UI Example:



The details view should be manually activatable over the settings for one or more country codes. If the DCC of this country is scanned, the details view is enabled by default for these countries in the scan result, otherwise the default view is shown without detailed information.

NOTE: The national app can be enhanced by authorization mechanisms, if required. For instance, to authorize a single person by sending a Push Notification or SMS to the app which contains signed authorization keys (public key can be embedded in the app). But this requires a clear and stable registration process of the users. In any case, the validation is offline on the device and it cannot be avoided that the official app is replaced by a modified one.

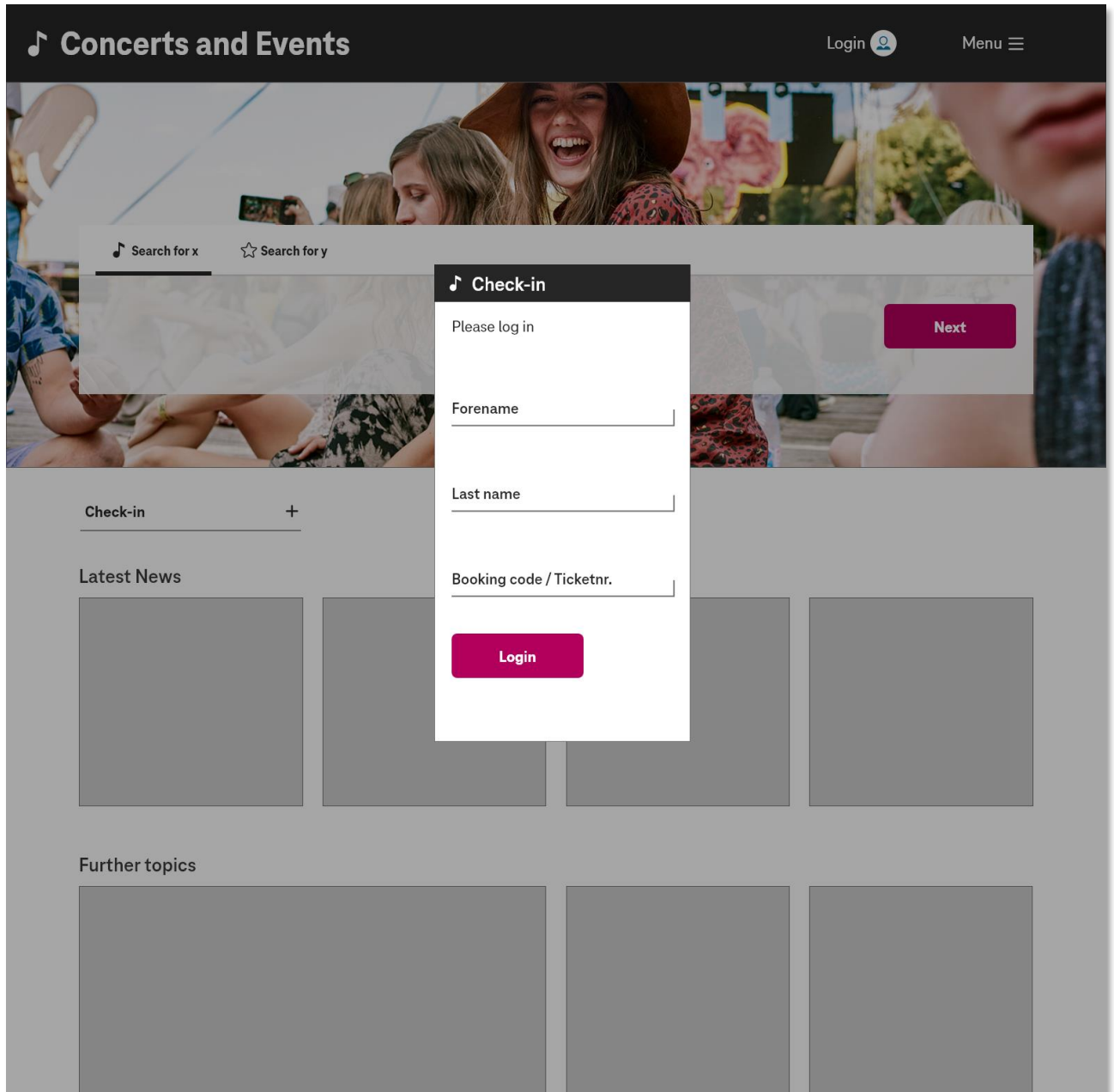
8.3 Anonymous DCC Export

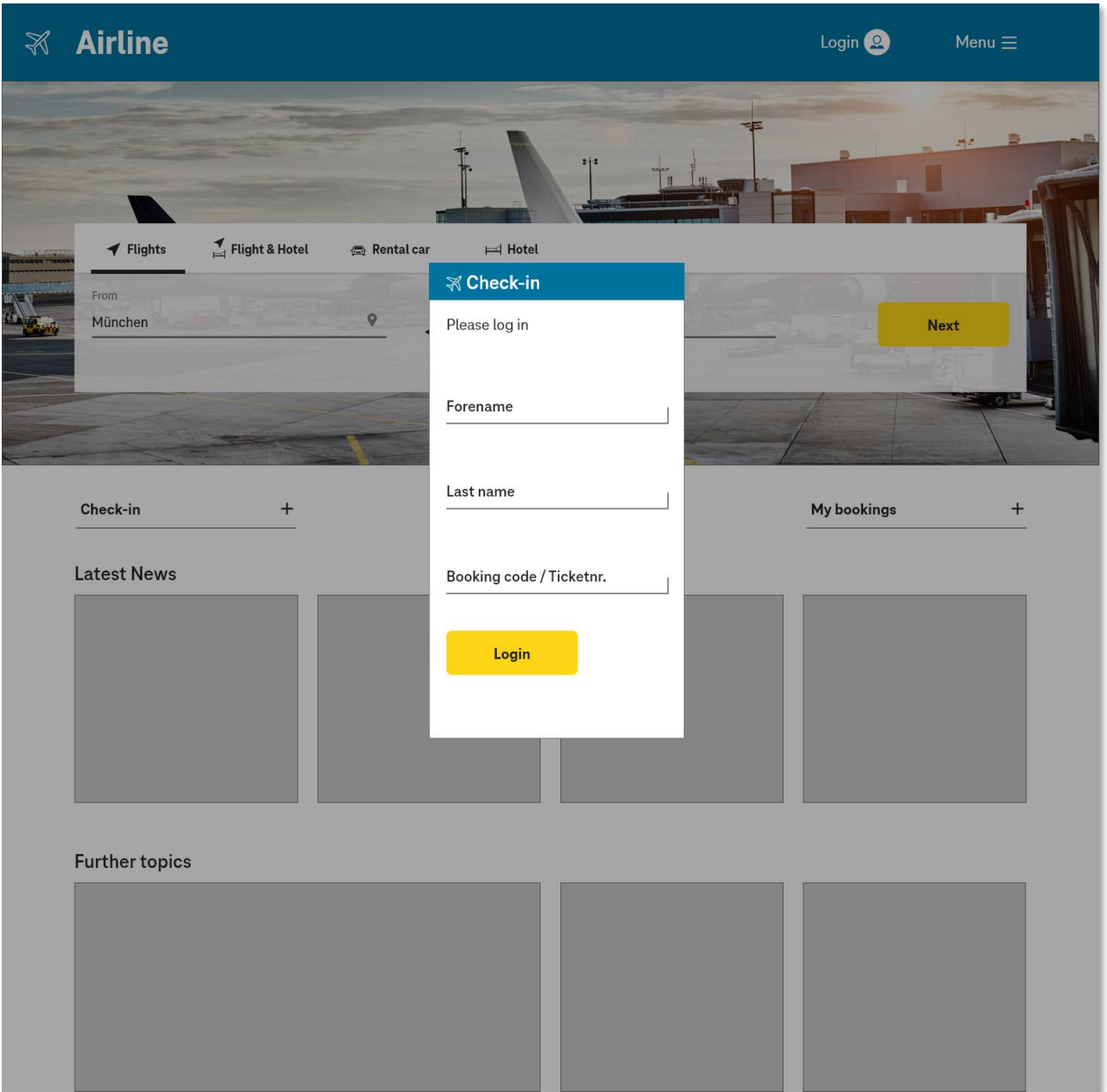
The anonymization of the personal data in the qr code should be selectable by the user of the app and must follow the rules of the DCC Anomaly Capture Process. (see https://ec.europa.eu/health/sites/default/files/ehealth/docs/covid-certificate_dcc_anomaly-capture-process_en.pdf)

Appendix A – Example Wallet UX




The screen below shows how the process from the wallet app perspective could look like. First, the code in the service provider web app is scanned and then the flow is started.

User tries to check in for any kind of service:





Service provider requests DCCs before the ticket issuance:

 **Airline**
Login 
Menu 


Check-in

In order to receive your ticket, you have to upload a test result / vaccination ticket or scan the QR code with your wallet app (please consider the validity till the time of arrival).
Click [here](#) for a tutorial.




As soon as all results are available, you have access to your e-ticket.

If not all certificates for all persons are available, this is no problem.
You can, without any problems, do a quick test directly at the airport. You're welcome to use the service at the counter to finish the check-in procedure.

Flight AIR094, Group booking
 Booking code 2DDBSL
 Munich (MUC)
 Dresden (DRE)
 Economy Classic
3 Passenger(s)
Tuesday, 17.08.2021 09:15

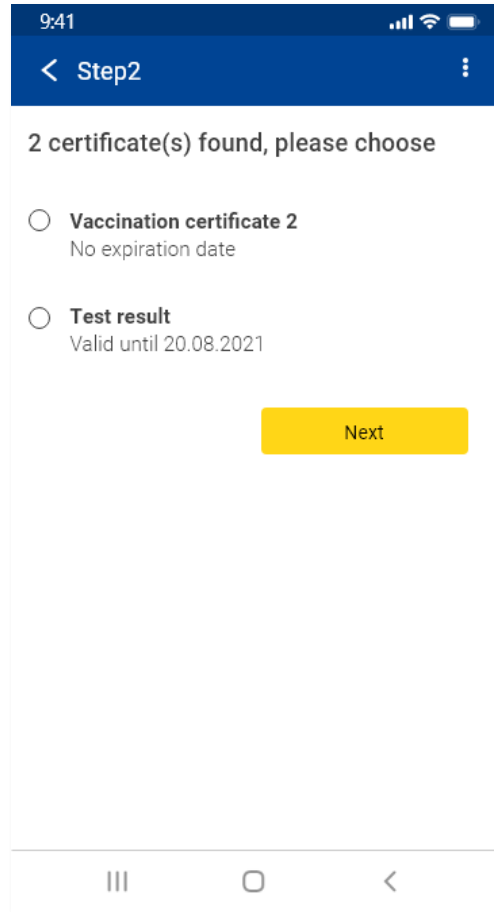
 **DCC (digital vaccination certificate)**

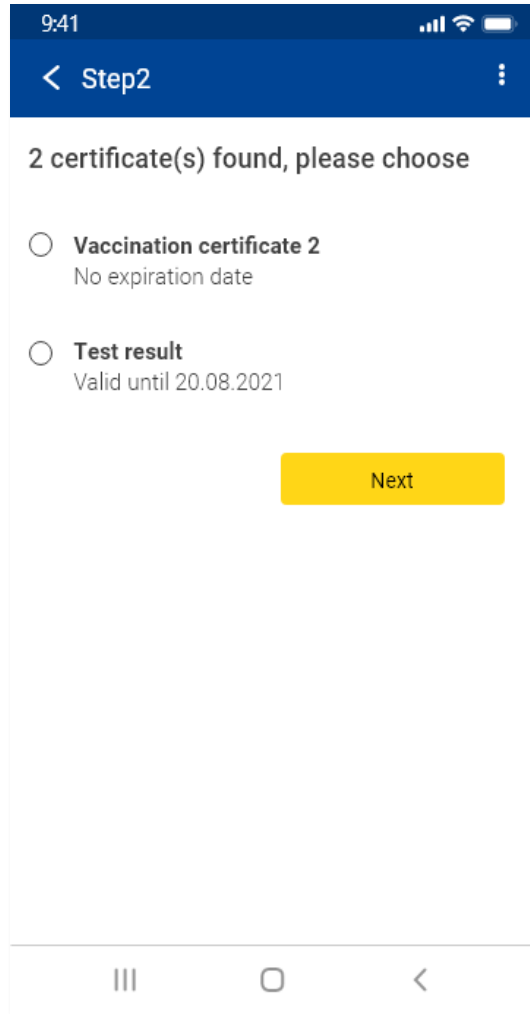
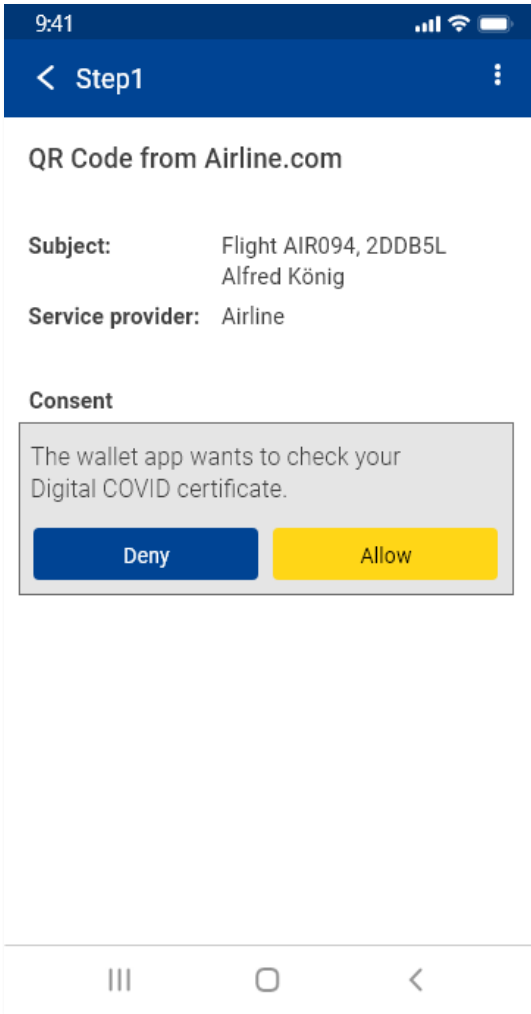
Upload a valid test/ vaccination certificate from your harddrive OR Scan the code with your [wallet app](#)

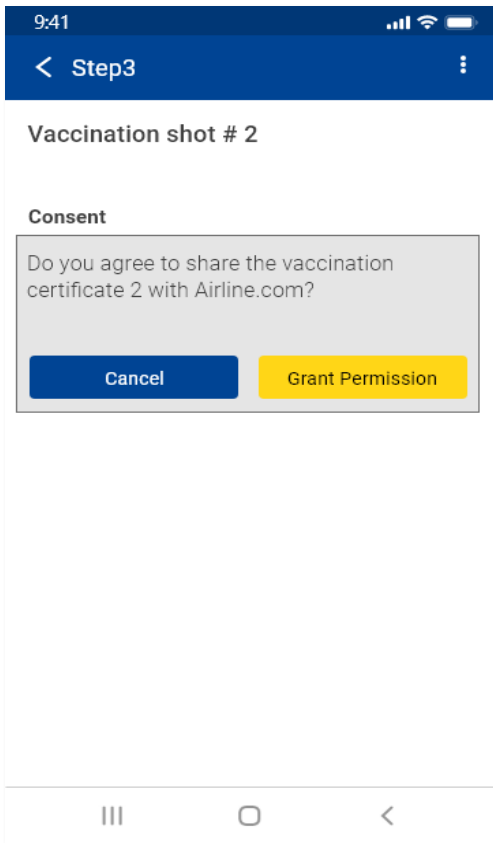
Condition fulfilled	Name		
<input type="radio"/>	Alfred König	<input type="text" value="Upload"/>	
<input type="radio"/>	Manuela König	<input type="text" value="Upload"/>	
<input type="radio"/>	Jonas König Age 3 years	<input type="text" value="Upload"/>	

Cancel
Submit Check-in

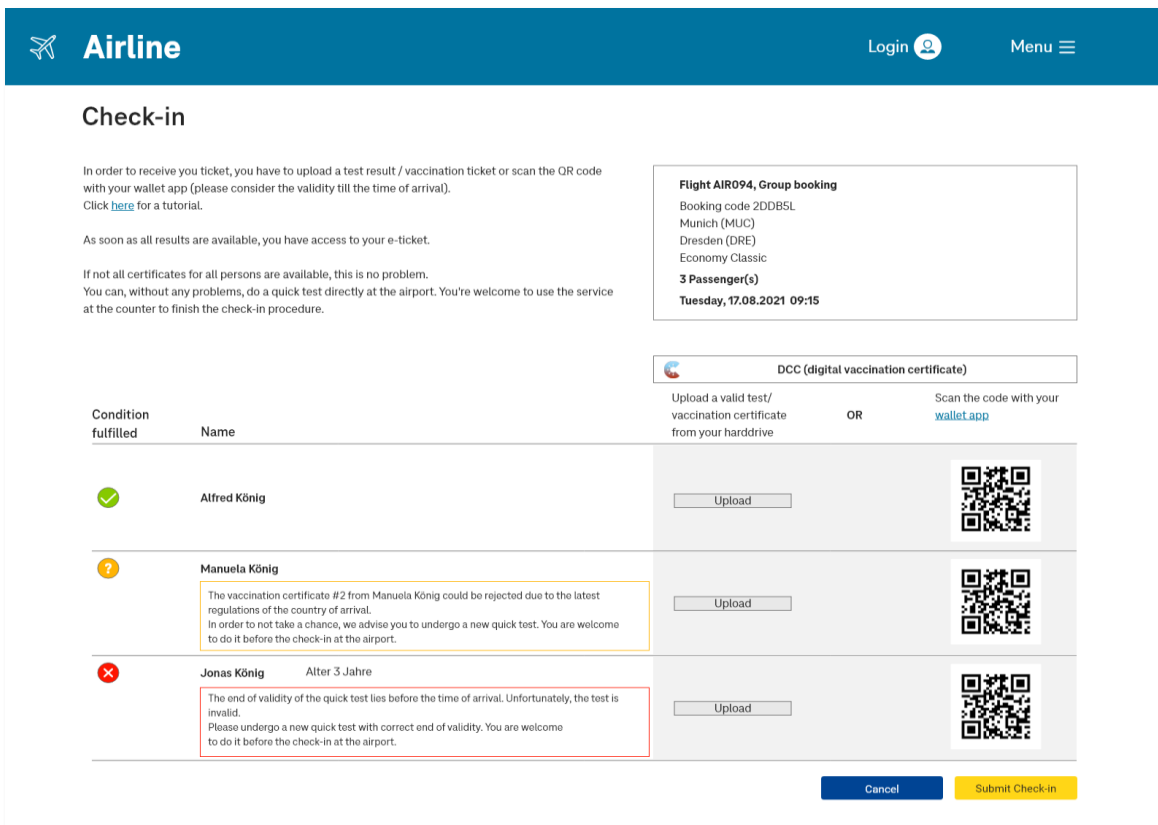
Wallet app is used to start the flow:



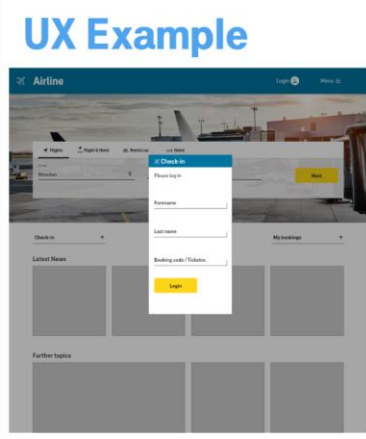
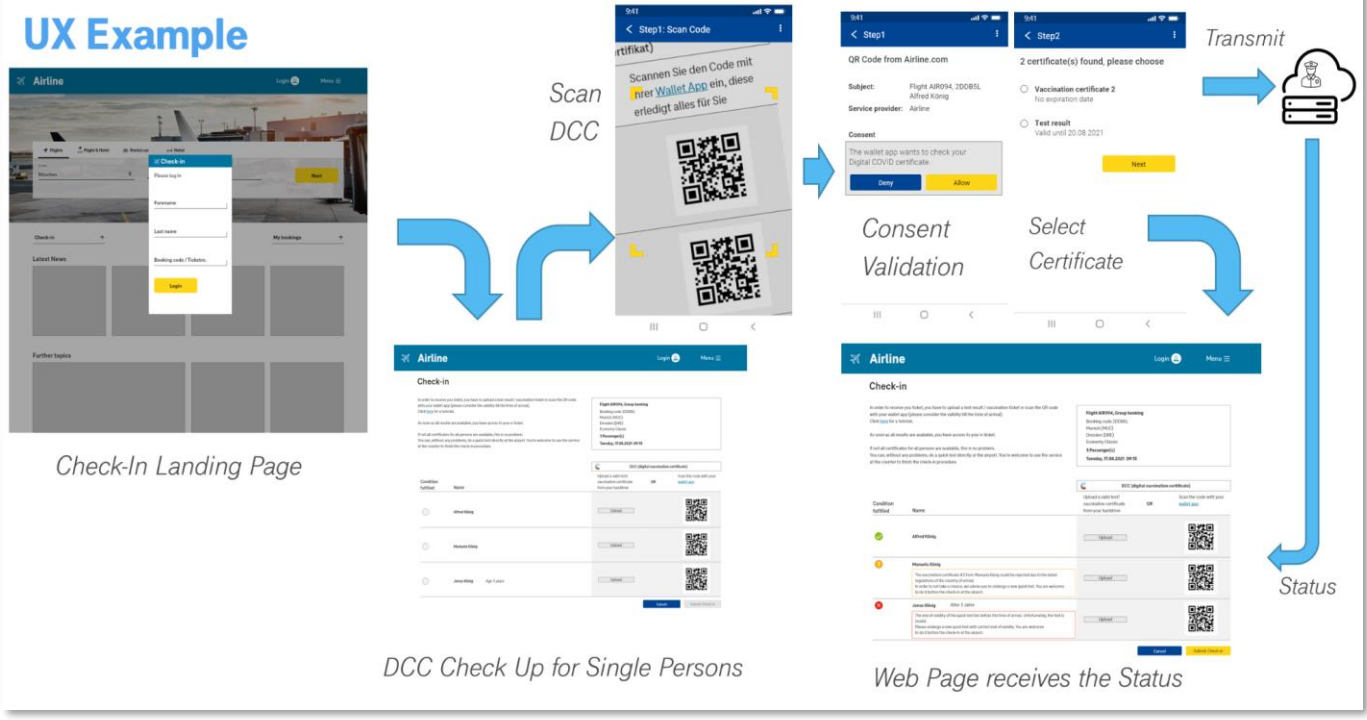




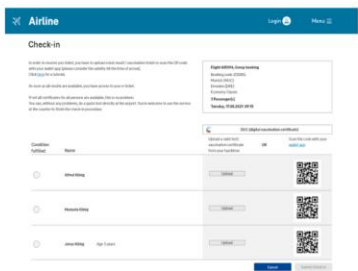
DCC was validated and the website returns the result (in this example one valid, one open and one failed):



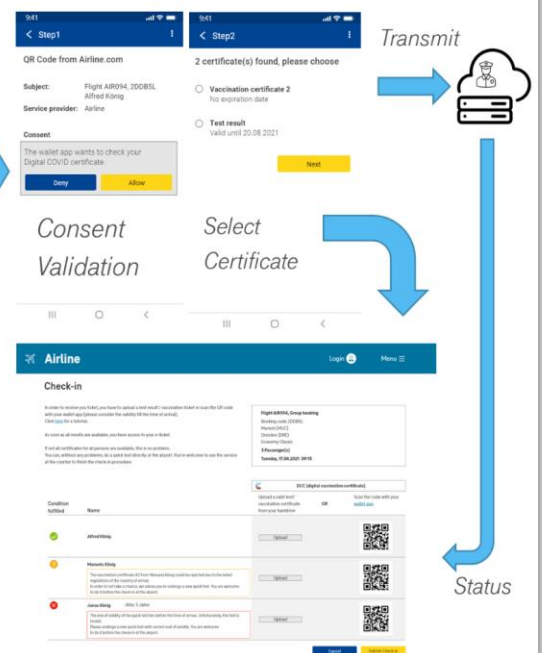
Appendix B – UX Example



Check-In Landing Page



DCC Check Up for Single Persons



Web Page receives the Status

Appendix C – Example Restriction Rule

```
{
  "Identifier": "GR-DE-0001",
  "Type": "Acceptance",
  "Country": "DE",
  "Version": "1.0.0",
  "SchemaVersion": "1.0.0",
  "Engine": "CERTLOGIC",
  "EngineVersion": "0.7.5",
  "CertificateType": "General",
  "Description": [{
    "lang": "en",
    "desc": "Restrictions are to be expected with the given certificate type.
    The restrictions can be checked at https://reopen.europa.eu/de."
  },{
    "lang": "de",
    "desc": "Mit dem gegebenen Zertifikatstyp sind Restriktionen zu erwarten.
    Auf https://reopen.europa.eu/de können die Restriktionen überprüft werden."
  },{
    "lang": "fr",
    "desc": "Restrictions are to be expected with the given certificate type.
    The restrictions can be checked at https://reopen.europa.eu/de."
  },{
    "lang": "es",
    "desc": "Restrictions are to be expected with the given certificate type.
    The restrictions can be checked at https://reopen.europa.eu/de."
  },{
    "lang": "it",
    "desc": "Restrictions are to be expected with the given certificate type.
    The restrictions can be checked at https://reopen.europa.eu/de."
  }
  ],
  "ValidFrom": "2021-08-18T14:00:00Z",
  "ValidTo": "2030-06-01T00:00:00Z",
  "AffectedFields": [
```

```
"t.0"  
],  
"Logic": {  
  "if": [  
    {  
      "var": "payload.t.0"  
    }, {  
      "+": ["+", "+"]  
    },  
    true  
  ]  
}  
}
```