

Language engineering for the Digital COVID-19 Certificate

Strumenta Virtual Meetup, April 14th 2022

Links in chat!

The EU DCC

Quick facts

What & Why

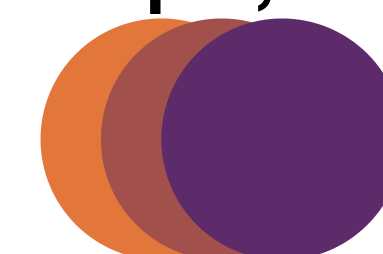
A system provided through the EU eHealth Network (eHN) to issue and verify digital proofs of vaccination, test, or recovery, to facilitate freedom of movement during the COVID-19 pandemic in a GDPR-compliant way.

Introduced July 1st 2021

Countries participating ~50 (EU MS + EFTA + “Third Countries”)

Number issued ~3 billion

Bigger than, or compatible with other standards: WHO/ICAO, resp., DIVOC

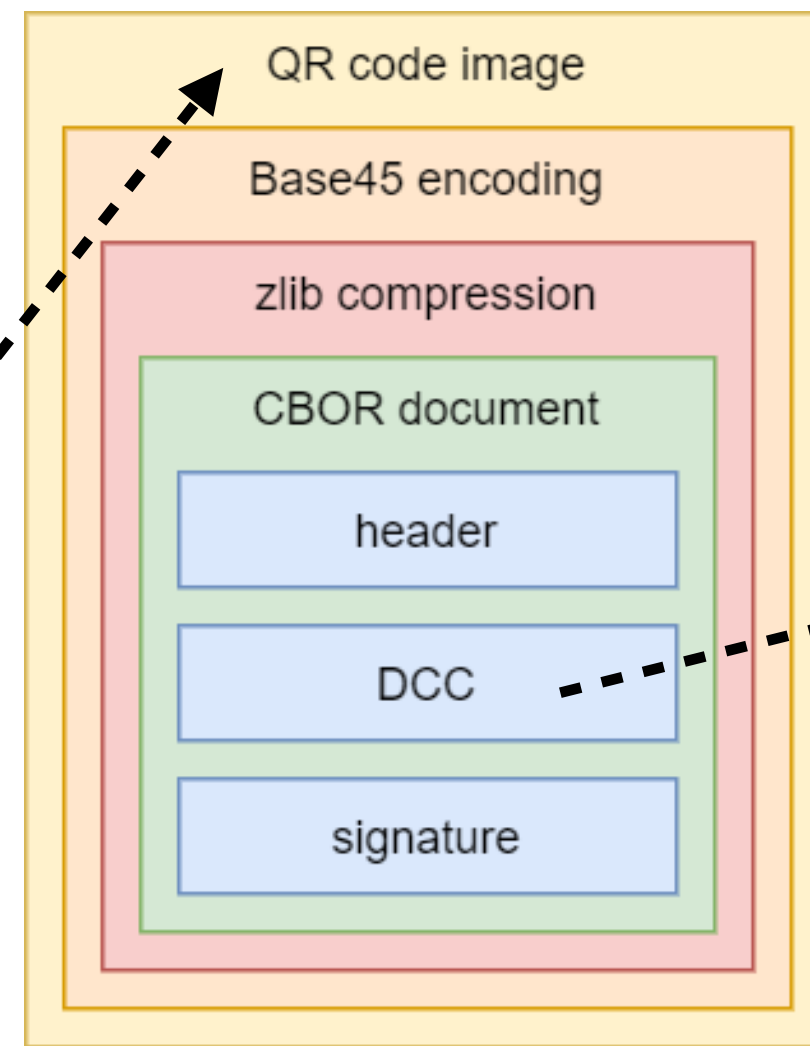


The EU DCC

What's in them



QR code

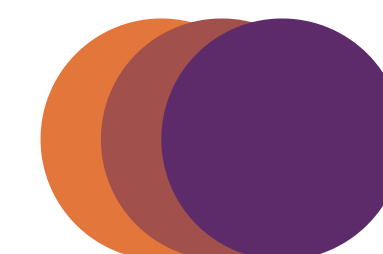


structure

```
{
  "ver": "1.3.0",
  "nam": {
    "fn": "Achternaam",
    "fnt": "ACHTERNAAM",
    "gn": "Voornaam",
    "gnt": "VOORNAAM"
  },
  "dob": "1963",
  "v": [
    {
      "tg": "840539006",
      "vp": "1119305005",
      "mp": "CVnCoV",
      "ma": "ORG-100032020",
      "dn": 1,
      "sd": 6,
      "dt": "2021-02-18",
      "co": "GR",
      "is": "Ministry of Health Welfare and Sport",
      "ci": "urn:uvci:01:NL:74827831729545bba1c279f592f2488a"
    }
  ]
}
```

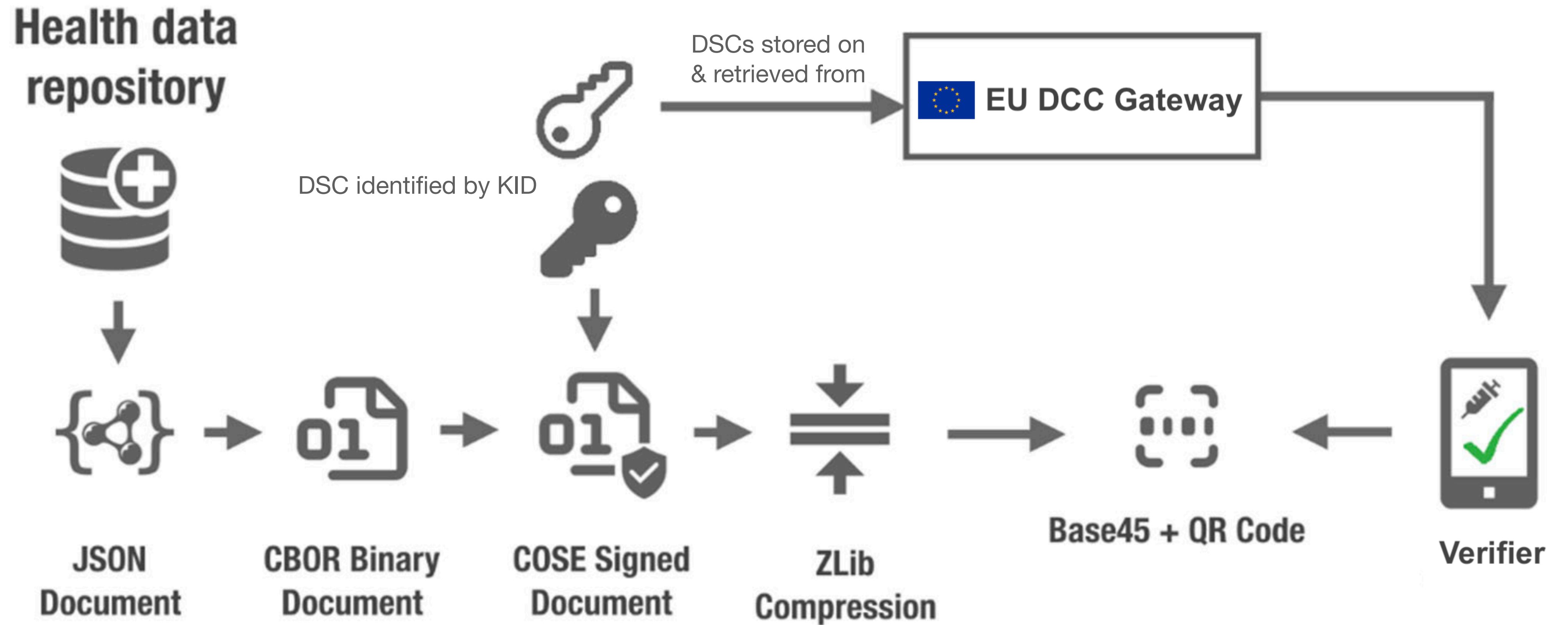
DCC JSON payload

Decode a DCC with e.g. <https://floysh.github.io/DCC-green-pass-decoder/>
(3rd party; demo)



The EU DCC

How they're made



The EU DCC

Software systems involved

EU provides **EU DCC Gateway** to publish DSCs to verify signatures:
the trust framework

Every participating country is responsible for building their own:

- 1) Verifiers (apps) - open-source reference implementations are available
- 2) Issuance infrastructure
- 3) National Backend in-between verifiers apps and Gateway

The EU DCC

How to verify them

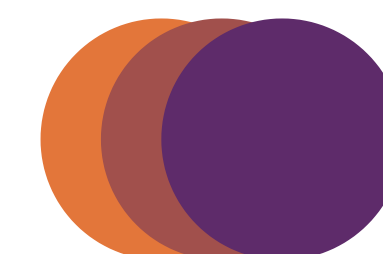
Technically valid Signature OK + JSON up-to-spec

Fit-for-entry Is the DCC acceptable for its holder to enter a Country of Arrival (CoA) regarding its entry regulations?

...or validation rules, or conditions, or constraints...

Examples of business logic as *business rules*:

- The result of a test certificate must be negative.
- A first vaccination with Janssen must be administered at least 28 days ago.
- A second vaccination with Pfizer must be at most 270 days old
...but minors are exempted!



The EU DCC

How to determine fit-for-entry

Sovereignty implies:

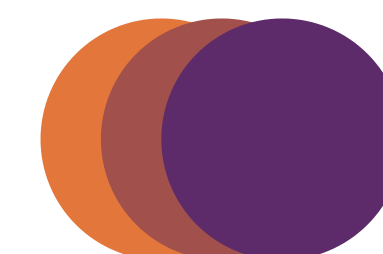
Every participating country can have their own entry regulations \Leftrightarrow business rules

Problem: Determine fit-for-entry *upfront*

Did someone say
“DSL”?!

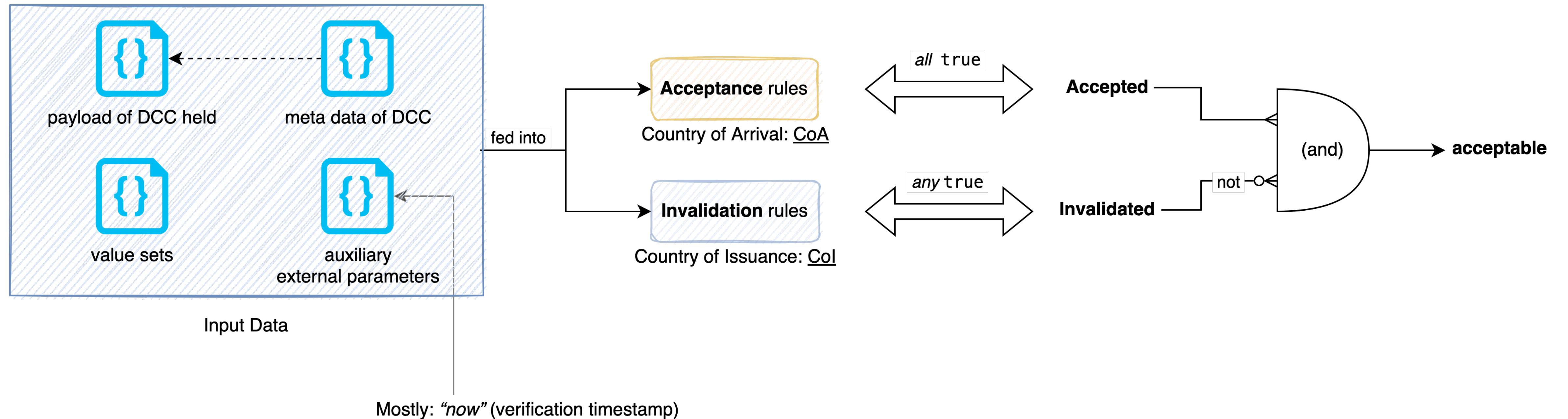
Solution: Publish business rules prescribed in an exchangeable, executable format on EU DCC Gateway

Design decision: Must be a JSON format



The EU DCC

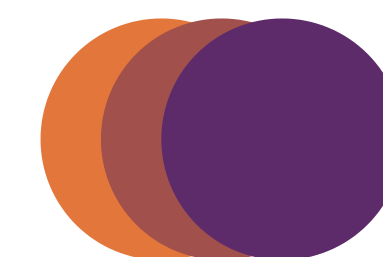
Validation framework



Prescribing business rules

Why a JSON format?

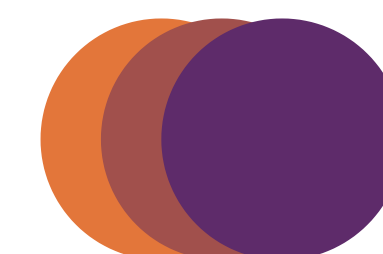
- 1) Logic as JSON is “just data” \Leftrightarrow e.g. compliant with Apple's bytecode policy
- 2) JSON is well-supported across many platforms
- 3) No need to write a parser for a textual DSL (for many platforms)
- 4) E.g. JsonLogic already somewhat known, and allegedly “human-readable”



Prescribing business rules

Why not use JsonLogic?

- 1) Not small: lots of operations, some with multiple variants (for convenience)
- 2) Behavior of implementations differs (\Leftrightarrow no specification)
- 3) Custom operations are needed for EU DCC; mainly: working with dates and timestamps



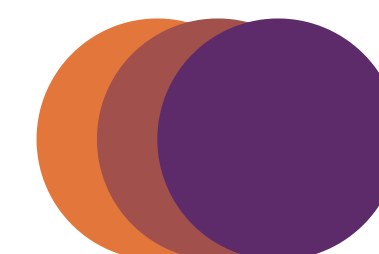
Prescribing business rules

What is *CertLogic*?

...a DSL?...

CertLogic is:

- A JSON format for expressing the logic part of business rules with
- “Inspired” by JsonLogic:
a **minimal** subset (on which it's compatible),
with a couple of *domain-specific* operations added
- Defined by: a specification that includes a test suite
- Implemented in: JavaScript (TypeScript), Java (Kotlin), Swift, Dart

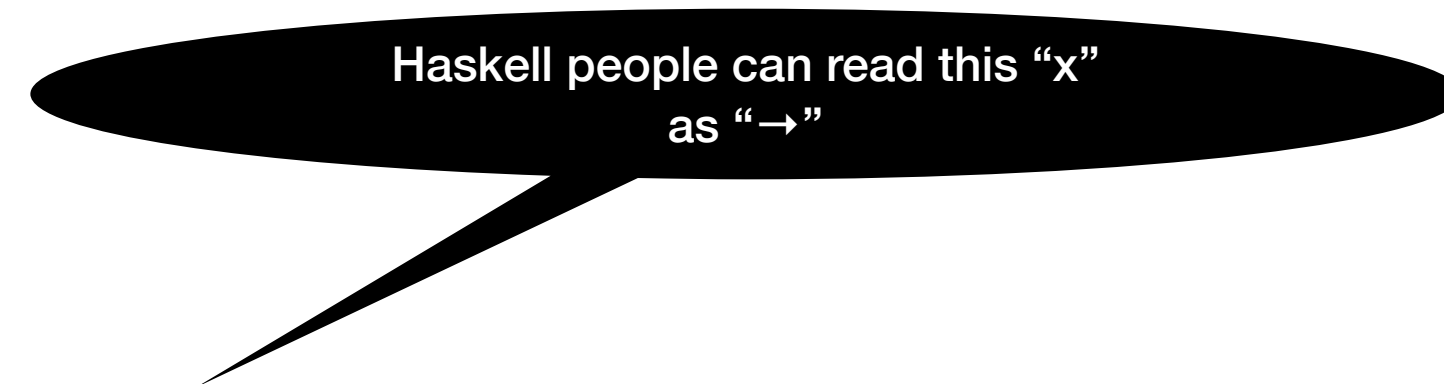


CertLogic

What is *CertLogic*?

A CertLogic expression *evaluates* (or: “is interpreted”) against given data.

As a function:

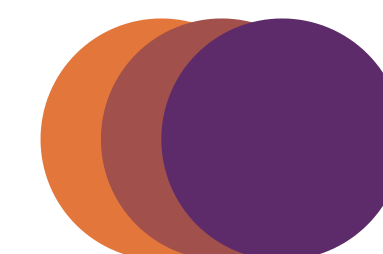


evaluate: $\langle\langle expr \rangle\rangle \times \langle\langle data \rangle\rangle \rightarrow \langle\langle result \rangle\rangle$

$\langle\langle expr \rangle\rangle$ and $\langle\langle data \rangle\rangle$ are in JSON format

$\langle\langle result \rangle\rangle$ is often JSON, but can contain Date objects

Can also throw an error if the expression is invalid,
or a type incompatibility is encountered.



CertLogic

The “grammar”

Valid CertLogic expressions are:

- a simple literal: a *«boolean»*, an *«integer»*, or a "*«string»*"
- an *operation* of the form

$$\{ \text{"«operation»"} : [\text{«operand}_1\text{»}, \text{«operand}_2\text{»}, \dots] \}$$

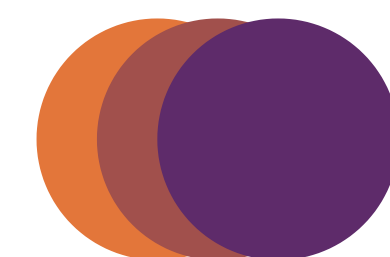
- an array of CertLogic expressions:

$$[\text{«expr}_1\text{»}, \text{«expr}_2\text{»}, \dots]$$


CertLogic

Operations (1/3)

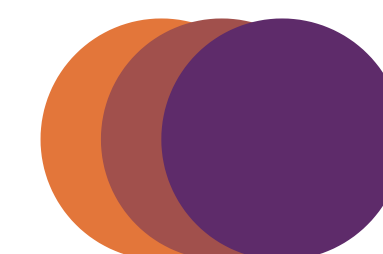
- data access: { "var": "«*path*»" }
Semantics: e.g. path = "v.0.f" evaluates to 1 on
{ "v": [{ "f": 1 }] }
- if: { "if": [«*guard*», «*then*», «*else*»] }
- and: { "and": [«*operand*₁», «*operand*₂», ...] }
- not: { "!" : [«*operand*»] }
- reduce: { "reduce": [«*operand*», «*lambda*», «*initial*»] }



CertLogic

Operations (2/3)

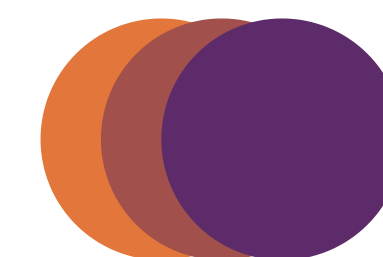
- equality: { "≡": [*operand₁*, *operand₂*] }
- membership: { "in": [*operand*, *array*] }
- integer and date comparisons:
{ "*operator*": [*operand₁*, *operand₂*[*, operand₃*]] }
- integer plus: { "+": [*operand₁*, *operand₂*] }



CertLogic

Operations (3/3)

- working with dates:
 - { "plusTime": [*operand*, *amount*, *unit*] }
Semantics: e.g. "2022-04-01" + 713 days = 2023-03-15
 - { "dccDateOfBirth": [*operand*] }
Semantics: "round up" a partial DOB YYYY[-MM] to latest possible date,
e.g. "2002" → 2002-12-31, and "2004-02" → 2004-02-29
 - { "extractFromUCI": [*operand*, *index*] }
Semantics: ("URN:UCI:01:NL:M6B3Y3663FA6REKP6KRL42#9", 2) → "M6B3Y3663FA6REKP6KRL42"



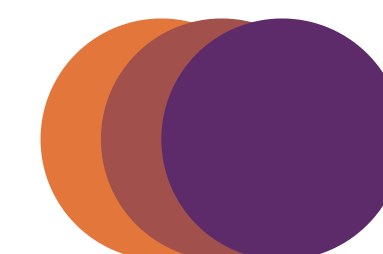
Demo: running a business rule

CertLogic

Operations (4/3)

“Where's my **OR**?!”

Desugaring to the rescue:

$$\{ \text{"or"}: [\langle expr_1 \rangle, \langle expr_2 \rangle] \}$$
$$\equiv \{ \text{"if"}: [\langle expr_1 \rangle, \langle expr_1 \rangle, \langle expr_2 \rangle] \}$$


CertLogic

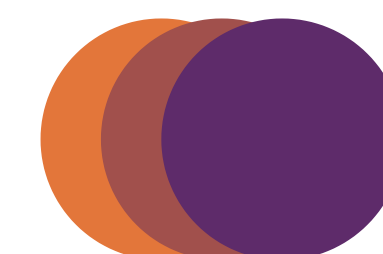
Implementations

- **JavaScript** (TypeScript): [GitHub](#) [NPM](#)
- **Java** (Kotlin): [GitHub](#)
- **Swift**: [GitHub](#)
- **Dart**: [GitHub](#) [pub.dev](#)
- JS util for working with EU DCC business rules: [GitHub](#) [NPM](#)
- JS component to render CertLogic expressions in HTML: [GitHub](#) [NPM](#)

CertLogic

Other tools

- Playground: **CertLogic-fiddle** (demo)
- Analyses of rules on EU DCC Gateway:
 - **GitHub**
 - **Vaccine-country matrix** (view)
 - **Vaccine acceptance per country** (view)
- Checking a DCC against rules of all participating countries:
DCC Crosscheck (demo)



CertLogic

Partial evaluation

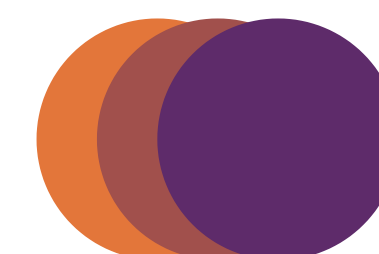
Idea

Mark values in the *«data»* as Unknown

Modify evaluate function (“interpreter”) so it doesn't reduce an *«expr»* that would produce Unknown (or any value that's not a CertLogic expression)

Usage






Partially evaluate and(*«all Acceptance rules of a country»*) against a DCC payload with `dt = Unknown` to derive which vaccines are accepted, and what their *validity ranges* are

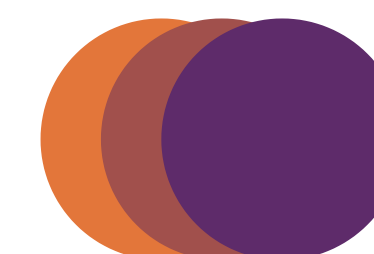


Analysing business rules

Using partial evaluation

Vaccine	AL	AT	CH	CY	CZ	DE	EE	ES	FI	FR	HR	IE	LT	LU	LV	ME	MT	NL	PL	RO	RS	SI	SK	TG	UA
AZD2816																									
Abdala																									
BBIBP-CorV																									
CVnCoV																									
Convidecia																									
CoronaVac																									
Covaxin																									
CoviVac																									
Covid-19-adsorvida-inativada																									
Covid-19-recombinant																									
Covifenz																									
Covishield																									
Covovax																									
Spikevax (Moderna)																									
Janssen																									
Comirnaty (Pfizer/BioNTech)																									
Vaxzevria (AstraZeneca)																									
Nuvaxovid																									
EpiVacCorona																									
EpiVacCorona-N																									
Hayat-Vax																									
Inactivated-SARS-CoV-2-Vero-Cell																									
MVC-COV1901																									
NVSI-06-08																									
Nuvaxovid (deprecated encoding)																									
R-COVI																									
SCTV01C																									
Sputnik-Light																									
Sputnik-M																									
Sputnik-V																									
VLA2001																									
Vidprevtyn																									
WIBP-CorV																									
YS-SC2-010																									

Accepted vaccines	1/1	2/2	2/1	3/3
Luxembourg (LU - )				
regs. on Re-open EU (regs. on)				
Covid-19-recombinant, Covishield, Spikevax (Moderna), Janssen, Comirnaty (Pfizer/BioNTech), Vaxzevria (AstraZeneca), Nuvaxovid, Nuvaxovid (deprecated encoding), R-COVI	14-270	0-270	0-366	0-366
Latvia (LV - )				
regs. on Re-open EU (regs. on)				
BBIBP-CorV, CoronaVac, Covaxin, Covishield, Spikevax (Moderna), Comirnaty (Pfizer/BioNTech), Vaxzevria (AstraZeneca), Nuvaxovid, Nuvaxovid (deprecated encoding)	15-270	15-270	0-	0-
Janssen	15-270	0-	0-	0-
Montenegro (ME - )				
regs. on Re-open EU (regs. on)				
BBIBP-CorV, CoronaVac, Covishield, Spikevax (Moderna), Comirnaty (Pfizer/BioNTech), Vaxzevria (AstraZeneca), Inactivated-SARS-CoV-2-Vero-Cell, Sputnik-V, WIBP-CorV	0-	0-180	0-	0-
Janssen	0-180	0-180	0-	0-
Malta (MT - )				
regs. on Re-open EU (regs. on)				
Spikevax (Moderna), Janssen, Comirnaty (Pfizer/BioNTech), Vaxzevria (AstraZeneca)	0-	0-270	0-	0-
Netherlands (NL - )				
regs. on Re-open EU (regs. on)				
BBIBP-CorV, CoronaVac, Covaxin, Covishield, Spikevax (Moderna), Comirnaty (Pfizer/BioNTech), Vaxzevria (AstraZeneca), Nuvaxovid, Nuvaxovid (deprecated encoding)	14-270	14-270	0-	0-
Janssen	28-270	0-	0-	0-

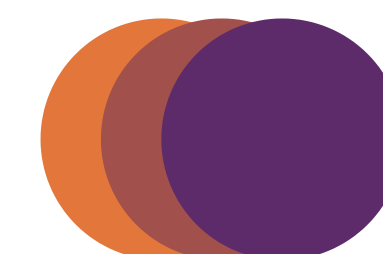


Demo: partial evaluation

Prescribing business rules

Things that went well

- 1) **Short time-to-market** In ~2 months from 0 to:
 - CertLogic spec + implementations
 - validation framework
 - business rules published on EU DCC Gateway
 - implemented in verifier apps
- 2) **Small spec** (and keeping it that way) Allowed quick implementation and controlled evolution, but flexible enough to adapt to changing requirements
- 3) **Versioning** Versioned specification and implementations independently
- 4) **Analysis** Analysed rules using language engineering techniques



Prescribing business rules

Things that could have gone better

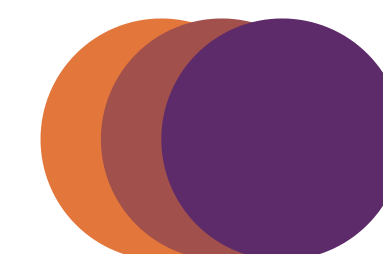
1) **Limited scope**

Only small part of entry regulations “fit” in validation framework, which was hard to extend.

2) **Adoption** Not all countries participating in the EU DCC share their entry regulations using the validation framework. Reasons:

i) Only small part of entry regulations covered - fear of “false positives”

ii) Writing business logic in JSON is tedious - a real DSL editor could help



Prescribing business rules

Observation

Little variance

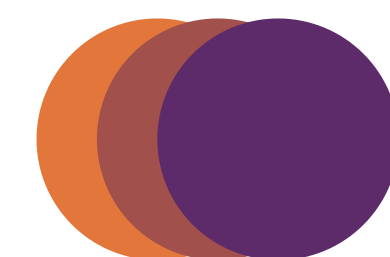
Most countries' business rules are pretty similar.

Was a language really necessary?

Could a simple (but more “fluid”) configuration have worked as well?



...probably not really...

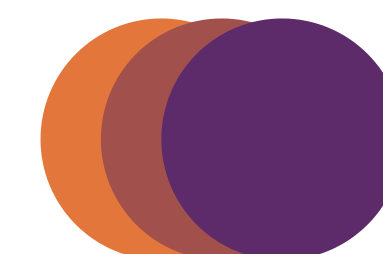


The EU DCC

The future

Use of EU DCC is being ramped down - remaining work:

- Maintenance
- Make “shelf-ready”:
 - Document what has been done
 - Document what could be done
(type system, more generic validation framework, better syntax + editor, ...)
- Use as “stepping stone” for e.g. new eHN initiatives (eID, ePrescription)



Questions?

Thank you!

