COVID-19 Verified Credentials meet reality

Can a Rules Engine help?

Neil Thomson

QueryVision, ToIP, MyData Canada

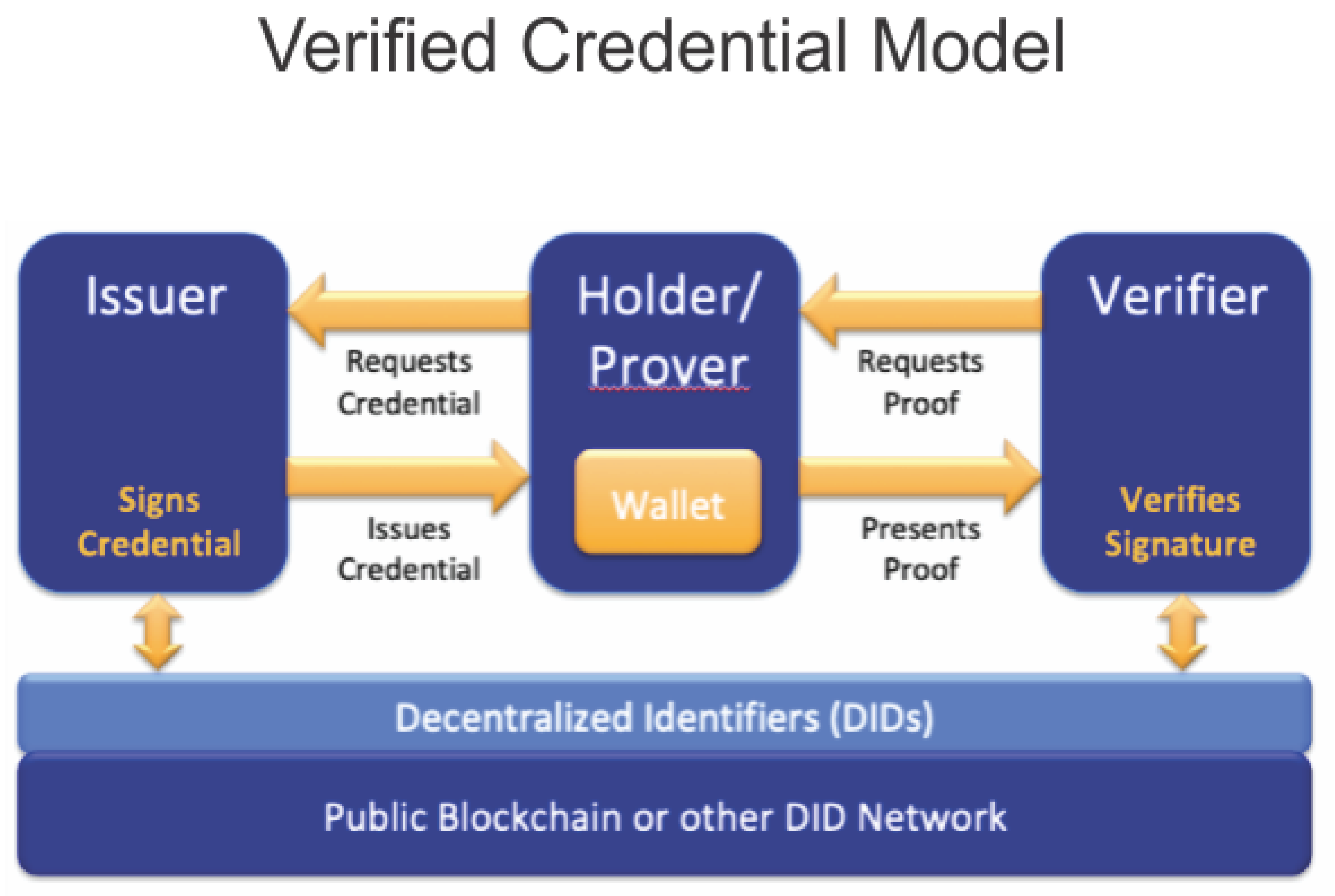
Goal of this Presentation

- See if a Rules Engine approach can address some issues raised by early feedback on how governments intend to use vaccine passports
- Present early thoughts on issues that need to be resolved, plus requirements (and suggestions) on possible solutions, benefits, trade-offs, …
- Generate Feedback
  - Feasible? Worth pursuing?
  - Issues that need to be fixed (or added to requirements)
  - Alternatives

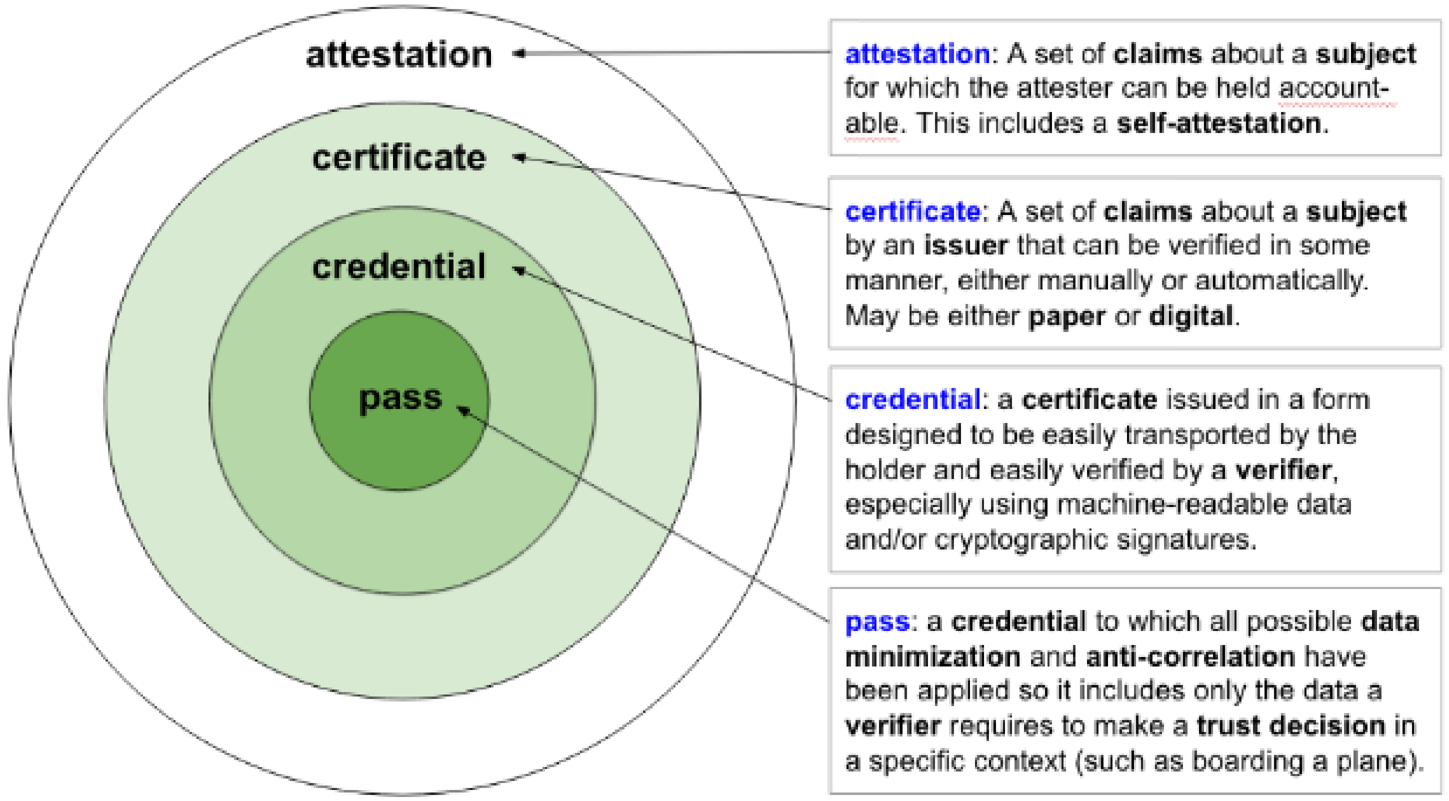
What is unique about the Good Health Pass?

- Technical Buy In - It’s a well thought through solution by a large number of industry players.
- Speed Dating - It is attempting to address an urgent, controversial world wide problem, that is getting near real-time reactions on how non-technical governance groups (governments, public-health) react to it
- Early Stages - It’s early enough that future directions can be suggested

Standard VC Issuer/Holder/Verifier model



GHP Blueprint – Data structure



attestation: A set of **claims** about a **subject** for which the attester can be held accountable. This includes a **self-attestation**.

certificate: A set of **claims** about a **subject** by an **issuer** that can be verified in some manner, either manually or automatically. May be either **paper** or **digital**.

credential: a **certificate** issued in a form designed to be easily transported by the holder and easily verified by a **verifier**, especially using machine-readable data and/or cryptographic signatures.

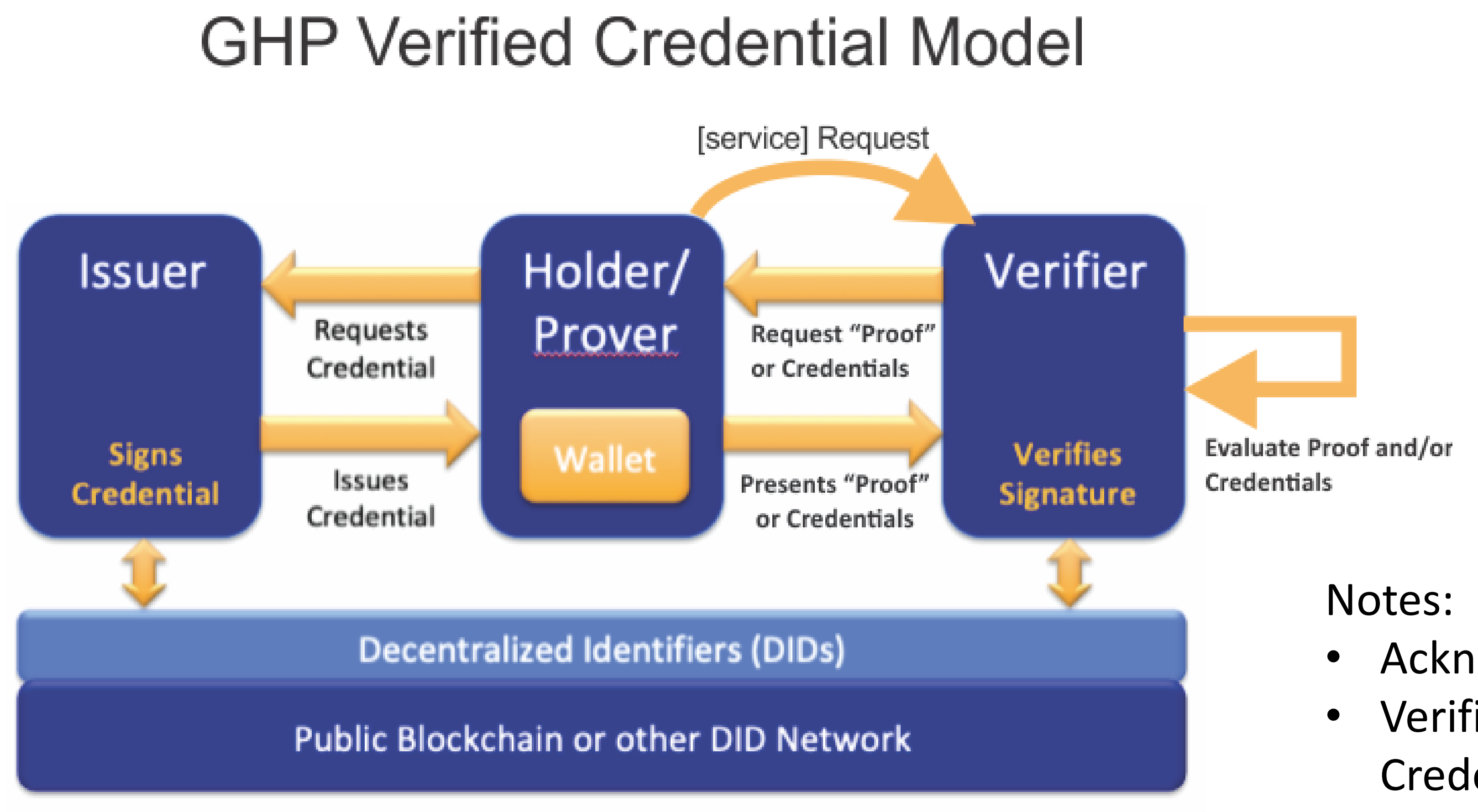
pass: a **credential** to which all possible **data minimization** and **anti-correlation** have been applied so it includes only the data a **verifier** requires to make a **trust decision** in a specific context (such as boarding a plane).

(Current) GHP “Blueprint

GHP VC definition

- A package of
  - Credentials (vaccine, test, recovery) with “minimal data” to support “Proof & presentation”
  - GHP "Pass": name, DOB, status (T/F) for:
    - Vaccination, [infection] Test, Recovery
- Alternative (to Pass.status):
  - Verifier asks for Holder consent to access credentials and the use them with (Verifier) internal logic to determine VC acceptance

GHP VC Issuer/Holder/Verifier model


GHP Verified Credential Model
[service] Request
Issuer
Holder/ Prover
Verifier
Requests Credential
Request "Proof" or Credentials
Evaluate Proof and/or Credentials
Signs Credential
Issues Credential
Wallet
Presents "Proof" or Credentials
Verifies Signature
Decentralized Identifiers (DIDs)
Public Blockchain or other DID Network

Notes:

- Acknowledge initiating request
- Verifier can ask for Proof or Credentials (or sub-set)
- (GHP) Proof -> "Pass"

Reality – initial Vaccine acceptance criteria

- US Center for Disease Control (CDC), Cruise Lines (& others following CDC)
  - Only Moderna and Pfizer vaccines
  - No mixing of vaccines (widely practiced in Canada, Spain, UK, ...)
- France
  - AstraZeneca, but not if made in India

Verifiers are the driver for VCs

Potential impact: GHP model

- Vaccine credential data upgrades
  - Multiple Vaccine credentials (vs single)
  - Supporting data [credential] claims
    - Medicinal product claims (Vaccines, Test)
      - Manufacturer (Country, site (manufacturing plant))
      - Date of manufacture, batch number
- Verifiers bypass GHP Passes – use GHP Credentials directly
  - Potential for PII/personal data compromised
  - Transparency issues?
  - Potential for jurisdiction change of criteria without notice
  - Holder uncertainty

If this trend continues?

1. [Worst case scenario]

   (Verifier) jurisdiction specific VCs
   - Custom evaluation logic
   - Custom (or extended) schema model, data

   Potential Impact:
   - Holders needing many VCs
   - Jurisdiction specific vaccine, test, recovery data required

2. [Unavoidable?]

   Upgrade GHP data from minimal to realistic to support jurisdiction acceptance criteria

3. …?

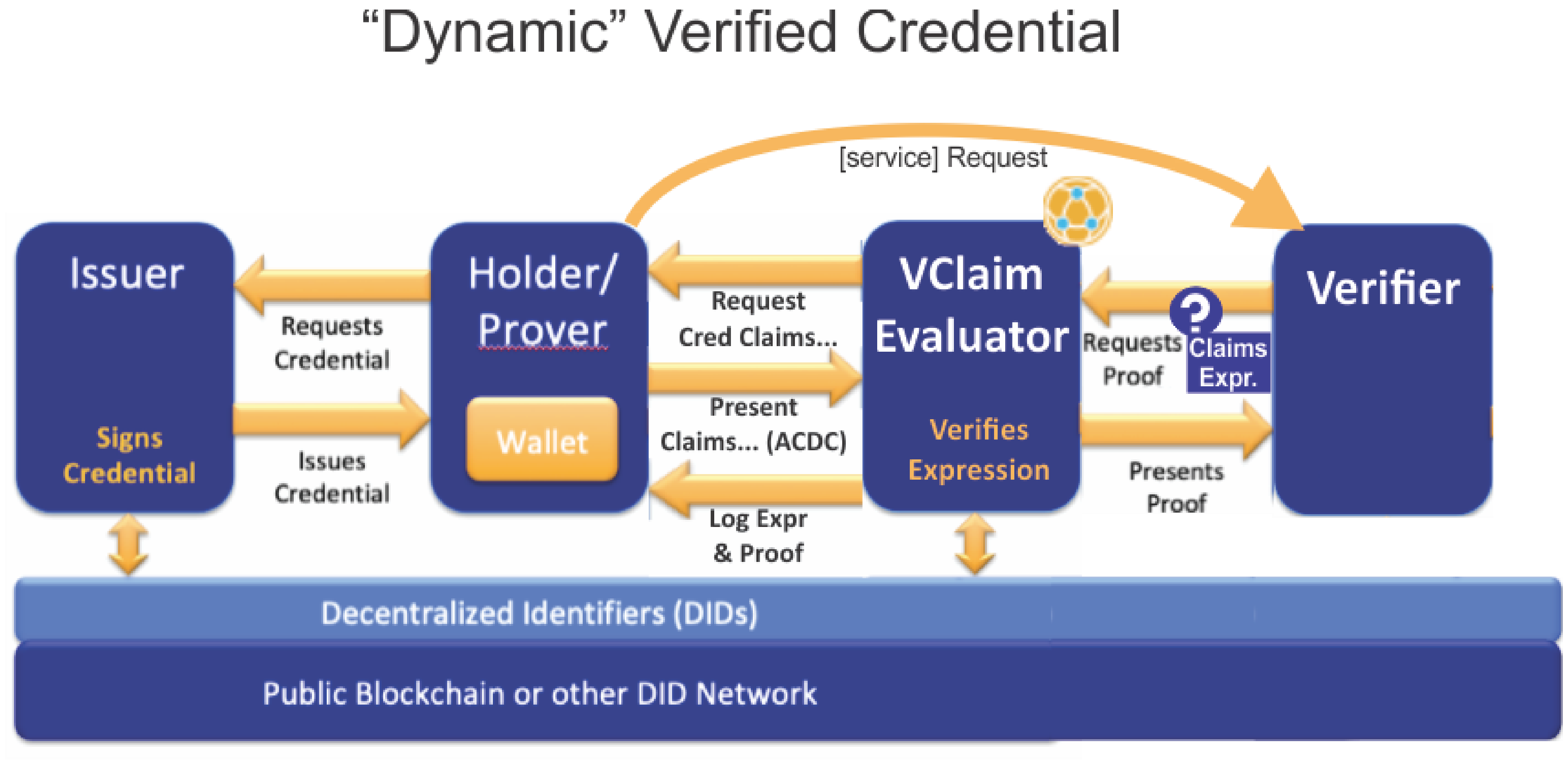
Could a Rule Engine help?

- Observations
  - The data problem has to be fixed, and that should be doable
  - VC acceptance evaluation logic (on the same data) can (and will) vary widely
  - (Ungoverned) Verifier (internal/black box) evaluation logic likely to become an issue at jurisdiction and personal levels due to lack of transparency.
    - If all I know is my GHP was rejected, how do I resolve the issue(s)?
- Proposal: provide an independent MyData Operator type component to evaluate (Holder) VC credentials based on Verifier supplied evaluation logic, with only the results returned (pass/fail)

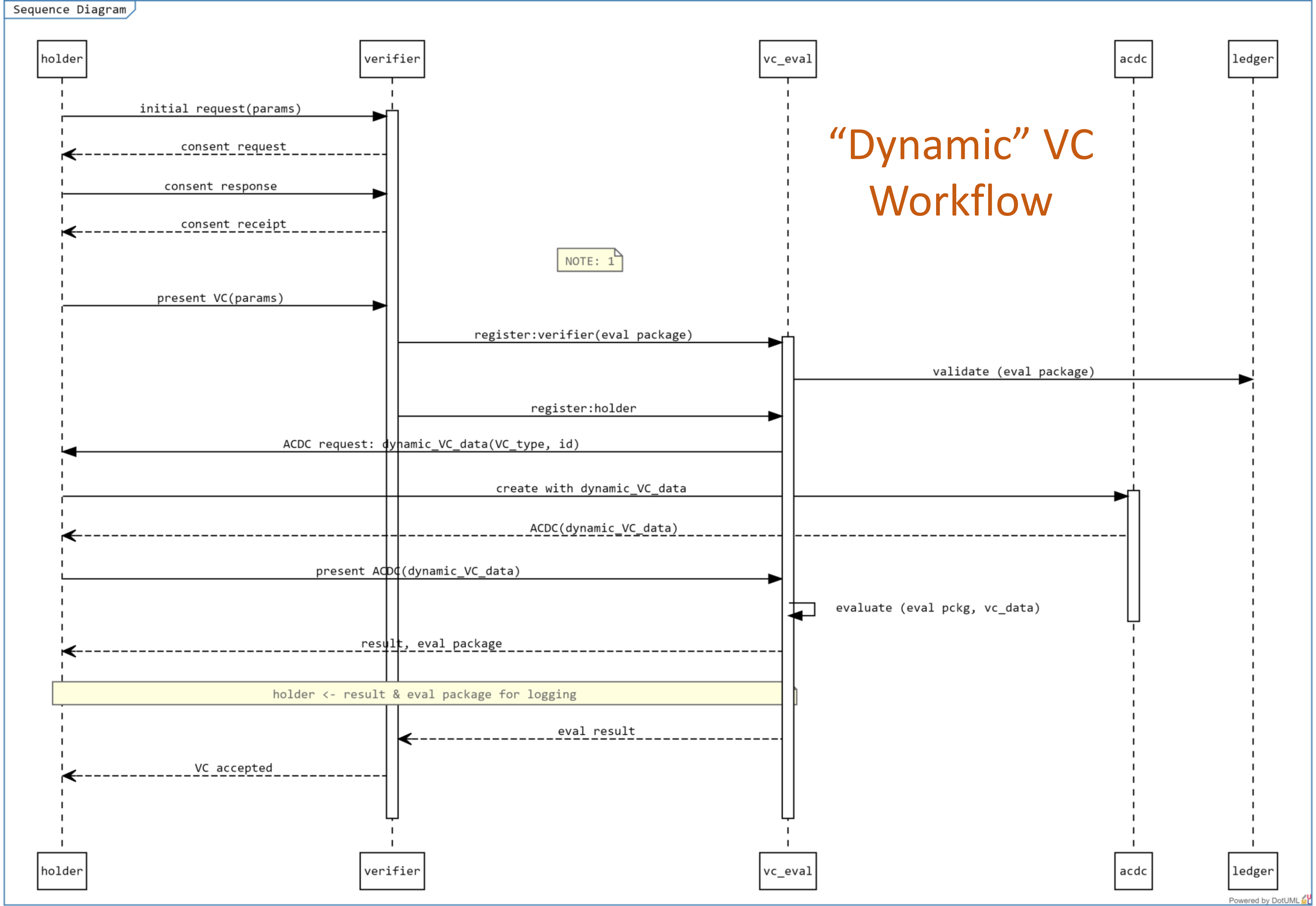
VClaim Evaluator - requirements

- Evaluator cannot compromise Issuer, Holder, Verifier security
  - It must be as resistant to MITM attacks as I/H/V components
- Evaluator communication with Holder, Verifier uses the same mechanisms (proofs, trust exchange, etc.)
- Evaluator must be a validated and approved component, evaluated by a Governance Authority or certified agent/agency
- The evaluation (rules) engine must support procedural logic and generate (and consume) data/query requests via APIs in a web environment, including agains DDE Data Containers (e.g. ACDC)
  - Preferred to use an off the shelf solution (JavaScript, Ruby, …) with VC Eval specific libraries

"Dynamic VC" model



"Dynamic" VC Workflow



VClaim Evaluator

- Holder provides Credential data via DDE Data Container (ACDC)
- Verifier supplies an evaluation expression written in a procedural language (e.g., JavaScript function) as source code
  - Verifier must sign the expression and register it with the Ledger prior to use
  - The Evaluator must validate the expression (with the ledger) at run time
  - Data access queries must be VC schema aware and have appropriate authorization
  - Access to VCs in a Data Container will be via Data Container (standardized) API calls
  - The default result of the evaluation is Pass/Fail
    - Other data returned in the result requires Holder consent (and related Governance)
- Result is passed to
  - Verifier - equivalent to Proof
  - To Holder - provides proof or reason for refusal, plus copy of evaluation expression (for logging, compliance, …)

VClaim Evaluator – why JavaScript?

- Evaluation core is queries against credential data
  - ACDC/Data Container will need at least minimal query API
  - Alternative
    - Eval component extracts ACDC data
- Procedural logic also required
  - If multiple vaccines and none have more than 1 dose
    then status = fail, reason = "mixed vaccines not acceptable"
- Why re-invent the wheel?
- Interpretive desired
  - Source code readable, auditable, easily portable, signable
  - Can run in DDE environments

VClaim Evaluator - Benefits

- Trusted 3rd party evaluation
- Transparency on evaluation criteria, outcomes (returns results to Holder)
- Minimal exposure of Holder credentials and claims. Only claims specifically included in the results (with consent) are presented to the Verifier
- Replaces the need for ZKP of ownership of claims/credential details as Evaluator providing equivalent service
- VCs can now be a passive container as evaluation partitioned to a separate component
- Use of a standard interpretive procedural language + data library provides all the required functionality without custom query language or evaluation engine