

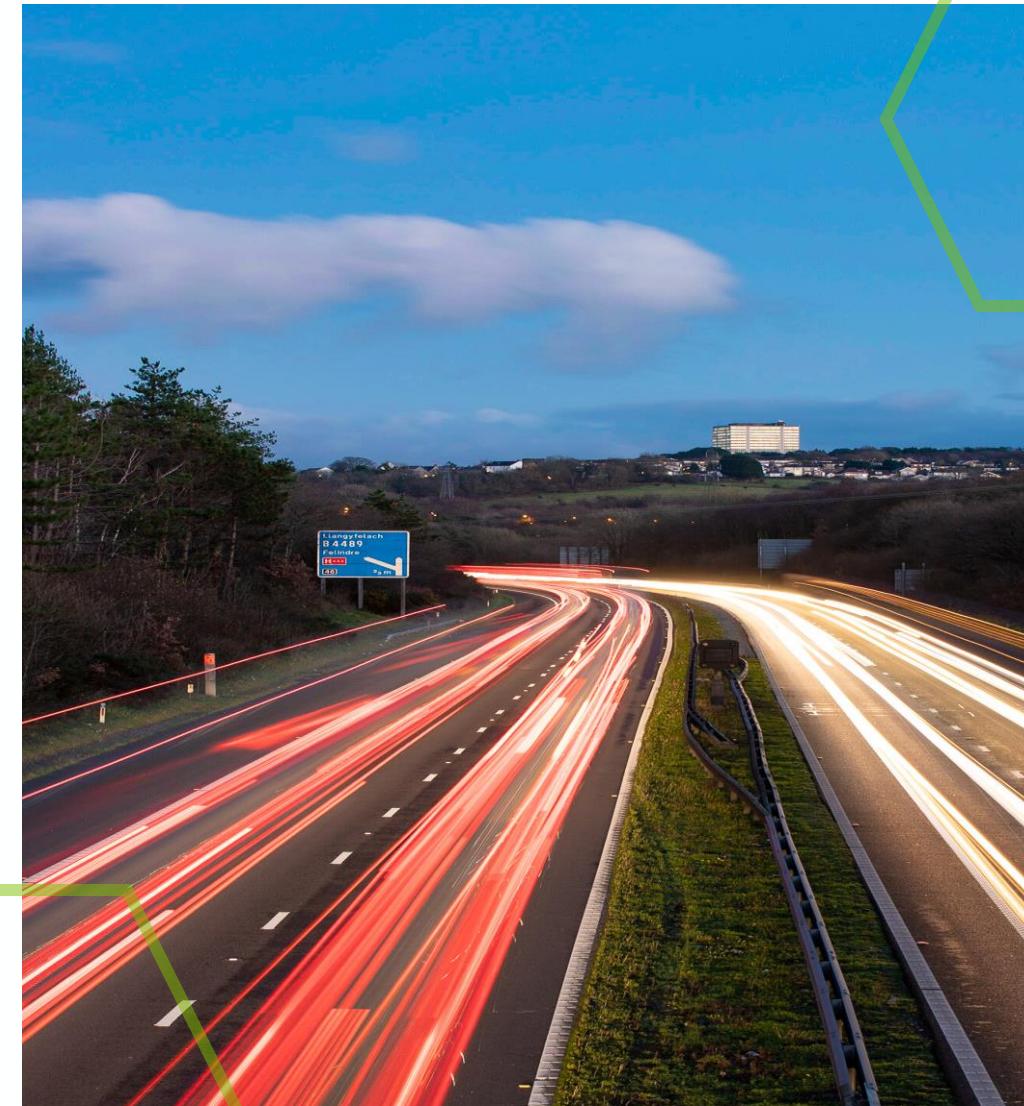


# JSON Schema for data design and contract, client and code generation

Tom Collins

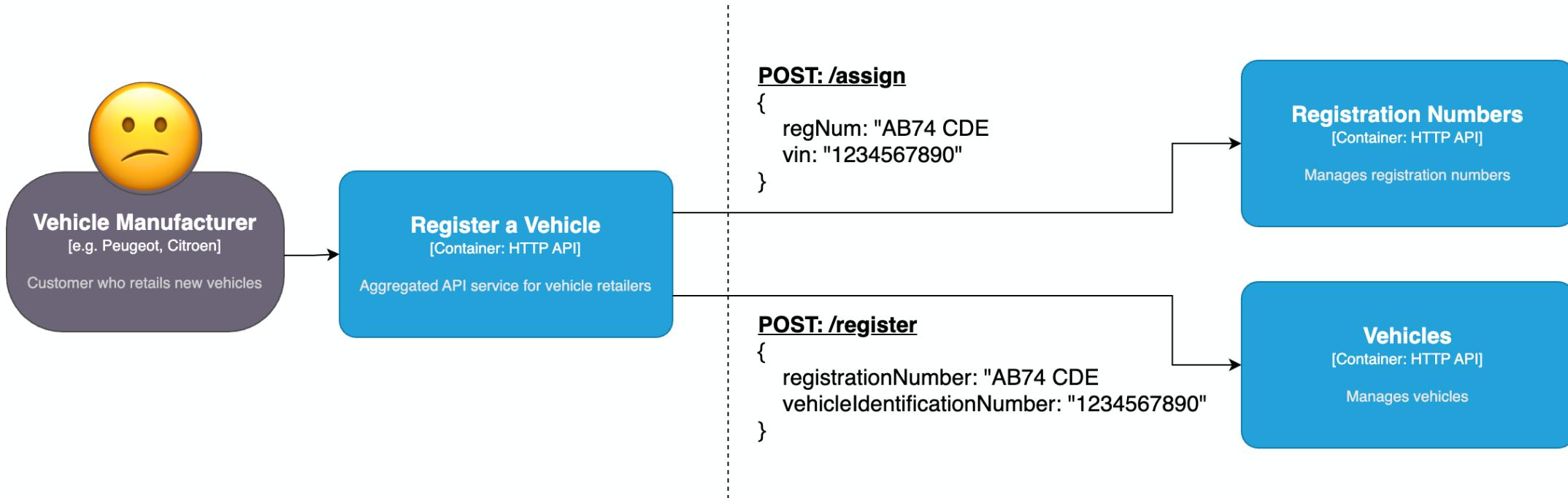


- Part of the UK government
- Maintain national registers of drivers and vehicles
- Collect over £6 billion a year in revenue
- Early adopters of public cloud
- Using JSON Schema extensively for over 5 years



# Background – Naming

DVLA Digital



# Background – Naming

DVLA Digital



**Team 1**

**POST: /assign**

```
{  
    "regNum": "AB74 CDE",  
    "vin": "1234567890"  
}
```



**Team 2**

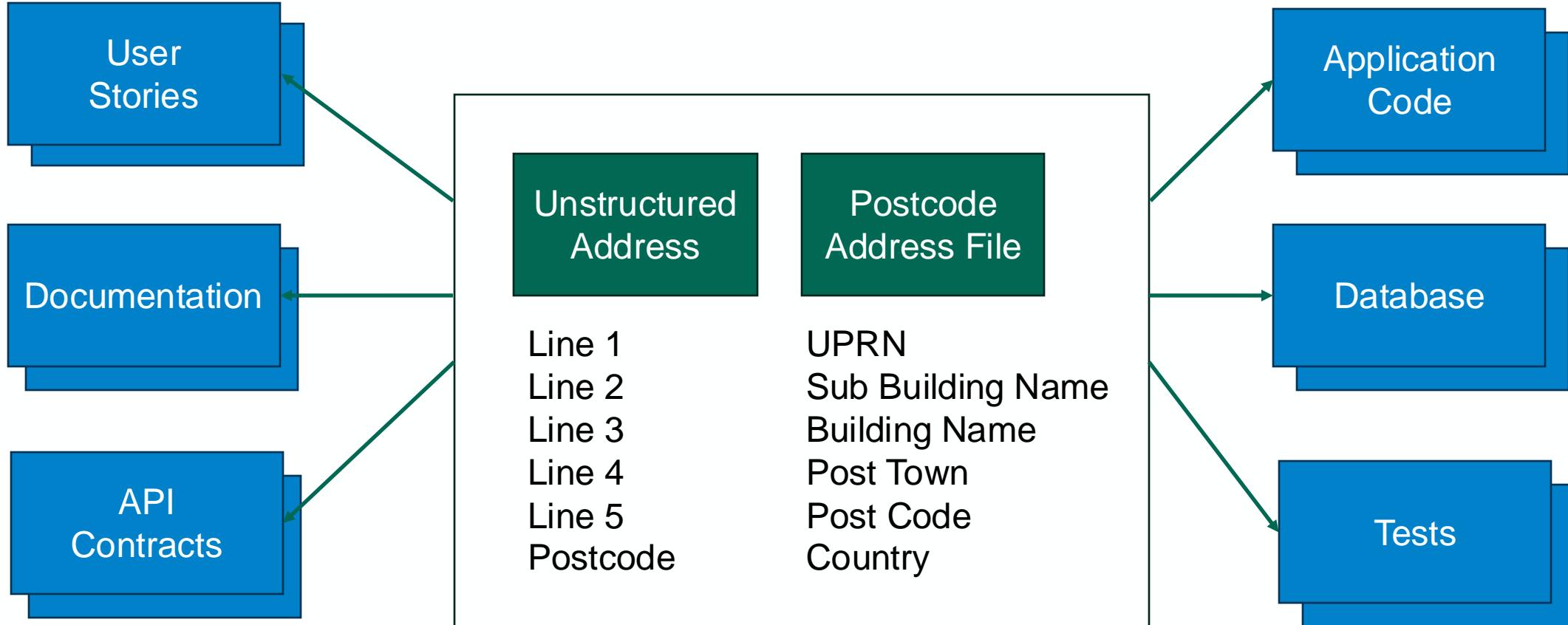
**POST: /register**

```
{  
    "registrationNumber": "AB74 CDE",  
    "vehicleIdentificationNumber": "1234567890"  
}
```



# Background – Data Models

DVLA Digital



Make it easy for teams to use our standard data models and naming conventions across their data, contracts and code.



### OSL Data Dictionary Schemas

Search

Address  
Application  
Common  
Customer  
Driver enquiries  
Drivers  
Driving licence application  
Enquiries platform  
Identity  
Internal portal  
Payment  
Personalised registration  
Print  
Standard  
Ved reminder  
Vehicle enquiries  
Vehicles

[Address](#) / [Types](#) / [v1](#) / [Address](#)

## Address

A DVLA address entity, which will be one of its child types as described in  
<https://technical.architecture.dvla.gov.uk/utilities/addressing/addressing-common-data-format.html>

\$id	<a href="https://osl-data-dictionary-schemas.engineering.dvla.gov.uk/address/types/v1/address.json">https://osl-data-dictionary-schemas.engineering.dvla.gov.uk/address/types/v1/address.json</a>
\$schema	<a href="http://json-schema.org/draft-07/schema#">http://json-schema.org/draft-07/schema#</a>

## Properties

Name	Type
One of:	<a href="#">Object (of type Structured Address)</a>
	<a href="#">Object (of type Unstructured Address)</a>
	<a href="#">Object (of type BFPO Address)</a>
	<a href="#">Object (of type International Address)</a>

## Example

```
{  
    "structuredAddress": {  
        "uprn": "10008904551",  
        "udprn": "4198105",  
        "subBuildingName": "UNIT 6",  
        "buildingName": "KISMET PARK",  
        "thoroughfareName": "PENARTH ROAD",  
        "postTown": "CARDIFF",  
        "country": "GBR"  
    }  
}
```



**Schema: 1608**

Data Types: 520

Request: 362

Response: 255

Events: 142

Message: 83

Applications: 7

Misc



## Focus on structure

- Describe structure
- Keep things simple
- Shape, names, formats

## Optimise for tooling

- Consider compatibility with tooling
- Don't describe business rules
- Avoid "logic" keywords e.g. not, if, then, else, allOf, oneOf, etc

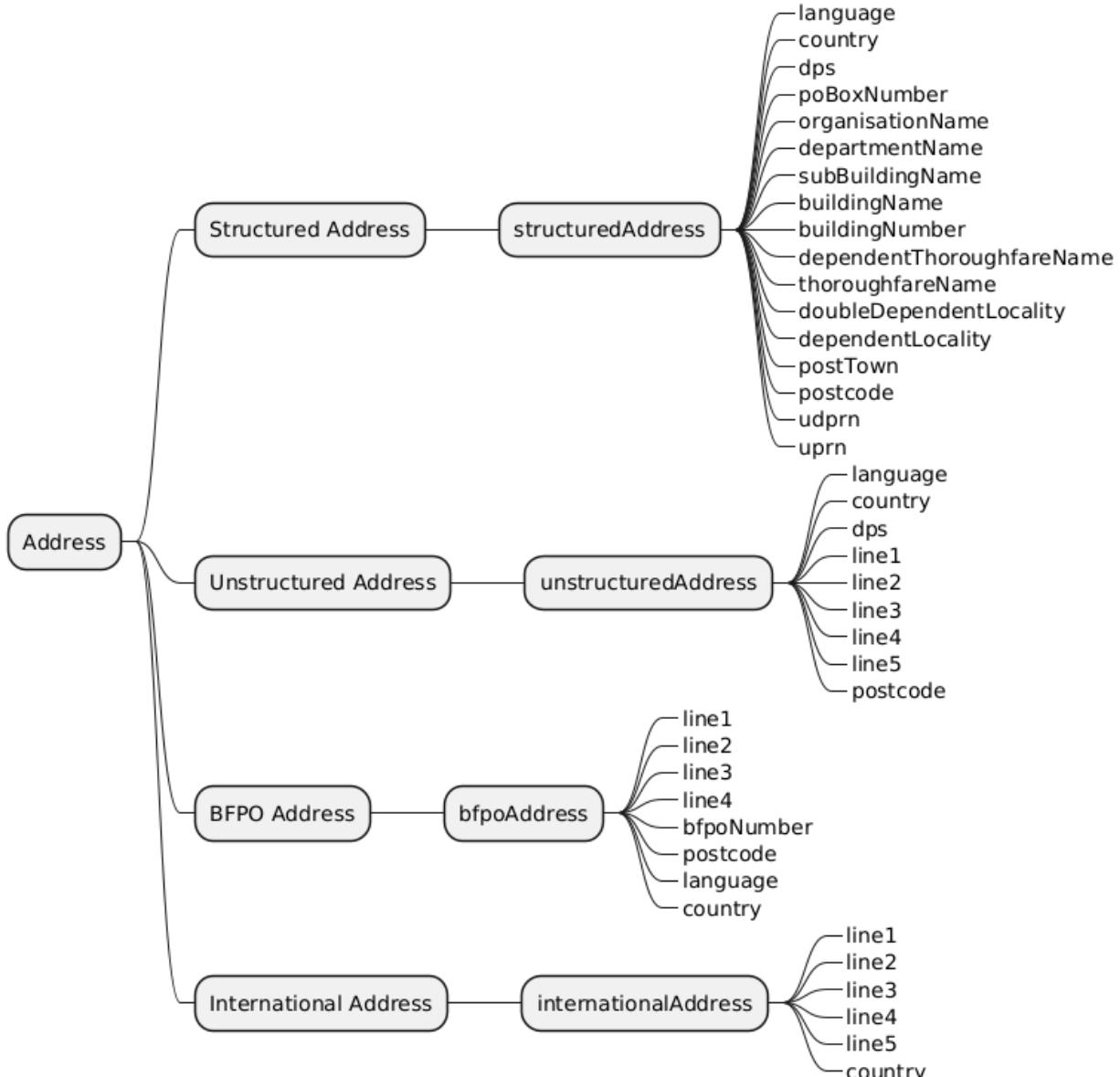
## Composition

- Avoid literal duplication
- Use \$ref where possible
- Use composition patterns



# Example - Address

DVLA Digital



# Example - Address

DVLA Digital

```
yaml > address > types > v1 > ! structured-address.yml
 1  $id: https://osl-data-dictionary-schemas.engineering.dvla.gov.uk/address/types/v1/structured-address.yml
 2  $schema: http://json-schema.org/draft-07/schema#
 3  title: Structured Address
 4  description: A DVLA structured address entity, a validated PAF address plus metadata as described in https://
 5  examples:
 6    - structuredAddress:
 7      uprn: "10008904551"
 8      udprn: "4198105"
 9      subBuildingName: UNIT 6
10      buildingName: KISMET PARK
11      thoroughfareName: PENARTH ROAD
12      postTown: CARDIFF
13      postcode: CF11 8TT
14      country: Wales
15      dps: 1A
16      language: EN
17      type: object
18      properties:
19        structuredAddress:
20          $ref: "#/definitions/structuredAddress"
21      required:
22        - structuredAddress
23      additionalProperties: false
24      definitions:
25        structuredAddress:
26          title: Structured Address Properties
27          type: object
28          properties:
29            language:
30              $ref: address.yml#/definitions/language
31            country:
32              $ref: address.yml#/definitions/country
33            dps:
34              $ref: address.yml#/definitions/dps
35            poBoxNumber:
36              $ref: "#/definitions/poBoxNumber"
37            organisationName:
38              $ref: "#/definitions/organisationName"
39            departmentName:
40              $ref: "#/definitions/departmentName"
41            subBuildingName:
42              $ref: "#/definitions/subBuildingName"
43            buildingName:
44              $ref: "#/definitions/buildingName"
45            buildingNumber:
46              $ref: "#/definitions/buildingNumber"
47            dependentThoroughfareName:
48              $ref: "#/definitions/dependentThoroughfareName"
49            thoroughfareName:
50              $ref: "#/definitions/thoroughfareName"
```

```
dist > json-schema-json > address > types > v1 > ! structured-address.json > ...
 1  {
 2    "$id": "https://osl-data-dictionary-schemas.engineering.dvla.gov.uk/address/types/v1/structured-address.json",
 3    "$schema": "http://json-schema.org/draft-07/schema#",
 4    "title": "Structured Address",
 5    "description": "A DVLA structured address entity, a validated PAF address plus metadata as described in https://",
 6    "examples": [
 7      {
 8        "structuredAddress": {
 9          "uprn": "10008904551",
10          "udprn": "4198105",
11          "subBuildingName": "UNIT 6",
12          "buildingName": "KISMET PARK",
13          "thoroughfareName": "PENARTH ROAD",
14          "postTown": "CARDIFF",
15          "postcode": "CF11 8TT",
16          "country": "Wales",
17          "dps": "1A",
18          "language": "EN"
19        }
20      },
21      {
22        "type": "object",
23        "properties": {
24          "structuredAddress": {
25            "$ref": "#/definitions/structuredAddress"
26          }
27        },
28        "required": [
29          "structuredAddress"
30        ],
31        "additionalProperties": false,
32        "definitions": {
33          "structuredAddress": {
34            "title": "Structured Address Properties",
35            "type": "object",
36            "properties": {
37              "language": {
38                "$ref": "address.json#/definitions/language"
39              },
40              "country": {
41                "$ref": "address.json#/definitions/country"
42              },
43              "dps": {
44                "$ref": "address.json#/definitions/dps"
45              },
46              "poBoxNumber": {
47                "$ref": "#/definitions/poBoxNumber"
48              },
49              "organisationName": {
50                "$ref": "#/definitions/organisationName"
51              }
52            }
53          }
54        }
55      }
56    ]
57  }
```

{ json }



Driver & Vehicle Licensing Agency

# Example - Address

```
target > generated-sources > jsonschema2pojo > uk > gov > dvla > osl > osldatadictonaryschemas > address > types > v1 > StructuredAddress.java
 1 package uk.gov.dvla.osl.osldatadictonaryschemas.address.types.v1;
 2
 3 import javax.annotation.processing.Generated;
 4 import javax.validation.constraints.NotNull;
 5 import javax.validation.constraints.Size;
 6
 7 import com.fasterxml.jackson.annotation.JsonInclude;
 8 import com.fasterxml.jackson.annotation.JsonProperty;
 9 import com.fasterxml.jackson.annotation.JsonPropertyDescription;
10 import com.fasterxml.jackson.annotation.JsonPropertyOrder;
11
12
13 /**
14  * Structured Address Properties
15  * <p>
16  *
17  */
18 */
19 @JsonInclude(JsonInclude.Include.NON_NULL)
20 @JsonPropertyOrder({
21     "language",
22     "country",
23     "dps",
24     "postBoxNumber",
25     "organisationName",
26     "departmentName",
27     "subBuildingName",
28     "buildingName",
29     "buildingNumber",
30     "dependentThoroughfareName",
31     "thoroughfareName",
32     "doubleDependentLocality",
33     "dependentLocality",
34     "postTown",
35     "postcode",
36     "udprn",
37     "uprn"
38 })
39 @Generated("jsonschema2pojo")
40 public class StructuredAddress {
41
42     /**
43      * Language
44      * <p>
45      * Used to indicate language, where known.
46      *
47      */
48     @JsonProperty("language")
49     @JsonPropertyDescription("Used to indicate language, where known.")
50     @Size(min = 0, max = 256)
51     private String language;
```

```
dist > types > address > types > v1 > TS_structured-address.dts > ...
1  /**
2   * This file was automatically generated by json-schema-to-typescript.
3   * DO NOT MODIFY IT BY HAND. Instead, modify the source JSON Schema file,
4   * and run json-schema-to-typescript to regenerate this file.
5   */
6  /**
7   * A DVLA structured address entity, a validated PAF address plus metadata as described in https://tech
8   */
9  export interface StructuredAddress {
10    structuredAddress: {
11      /**
12       * Used to indicate language, where known.
13       */
14      language?: string;
15      /**
16       * Value as listed in ISO 3166-2, except for subdivision name used specifically for Wales and
17       */
18      country?: string;
19      /**
20       * Uniquely identifies each address (Delivery Point) within a single postcode.
21       */
22      dps?: string;
23      /**
24       * When the Post Office (PO) Box Number field is populated, it is typically prefixed with 'PO BO
25       */
26      poBoxNumber?: string;
27      /**
28       * Name of the organisation registered at this address.
29       */
30      organisationName?: string;
31      /**
32       * Used to supplement Organisation Name to identify a department within the organisation.
33       */
34      departmentName?: string;
35      /**
36       * When a premise is split into individual units such as flats, apartments or business units.
37       */
38      subBuildingName?: string;
39      /**
40       * Name of residential or commercial premise.
41       */
42      buildingName?: string;
43      /**
44       * Number to identify premise on a thoroughfare or dependant thoroughfare.
45       */
46      buildingNumber?: string;
47      /**
48       * Used to supplement thoroughfare. When a thoroughfare name is used twice in the same Post Town
49       */
50      dependentThoroughfareName?: string;
```

```
dist > ruby-faker-maker-factories > address > types > v1 > structured-address.rb
1   FakerMaker.factory(:structured_address_properties) do
2     language(json: 'language') { Faker::Lorem.characters(number: (1..256).to_a.sample) }
3     country(json: 'country') { Faker::Lorem.characters(number: (1..256).to_a.sample) }
4     dps(json: 'dps') { Faker::Lorem.characters(number: (1..2).to_a.sample) }
5     po_box_number(json: 'poBoxNumber') { Faker::Lorem.characters(number: (1..6).to_a.sample) }
6     organisation_name(json: 'organisationName') { Faker::Lorem.characters(number: (1..80).to_a.sample) }
7     department_name(json: 'departmentName') { Faker::Lorem.characters(number: (1..60).to_a.sample) }
8     sub_building_name(json: 'subBuildingName') { Faker::Lorem.characters(number: (1..30).to_a.sample) }
9     building_name(json: 'buildingName') { Faker::Lorem.characters(number: (1..50).to_a.sample) }
10    building_number(json: 'buildingNumber') { Faker::Lorem.characters(number: (1..4).to_a.sample) }
11    dependent_thoroughfare_name(json: 'dependentThoroughfareName') { Faker::Lorem.characters(number: (1..60).to_a.sample) }
12    thoroughfare_name(json: 'thoroughfareName') { Faker::Lorem.characters(number: (1..60).to_a.sample) }
13    double_dependent_locality(json: 'doubleDependentLocality') { Faker::Lorem.characters(number: (1..35).to_a.sample) }
14    dependent_locality(json: 'dependentLocality') { Faker::Lorem.characters(number: (1..35).to_a.sample) }
15    post_town(json: 'postTown', required: true) { Faker::Lorem.characters(number: (1..30).to_a.sample) }
16    postcode(json: 'postcode', required: true) { Faker::Lorem.characters(number: (1..8).to_a.sample) }
17    updrn(json: 'updrn') { Faker::Lorem.characters(number: (1..2).to_a.sample) }
18    uprn(json: 'uprn') { Faker::Lorem.characters(number: (1..2).to_a.sample) }
19  end
20
21  FakerMaker.factory(:structured_address) do
22    |structured_address| { structured_address(json: 'structuredAddress', required: true) { FakerMaker[:structured_address_properties].build } }
23  end
24
```



# Example - Address

## Address

```
yaml > address > types > v1 > ! address.yml > {} definitions > {} country > [ ]examples
 1   $id: https://osl-data-dictionary-schemas.engineering.dvla.gov.uk/address/types/v1/
 2   address.yml
 3   $schema: http://json-schema.org/draft-07/schema#
 4   title: Address
 5   description: A DVLA address entity, which will be one of its child types as described
 6   in addressing-common-data-format.html
 7   examples: ...
 8   type: object
 9   oneOf:
10     - $ref: structured-address.yml
11     - $ref: unstructured-address.yml
12     - $ref: bfpo-address.yml
13     - $ref: international-address.yml
14   definitions:
15     postcode:
16       title: Postcode
17       description: "The Postcode is part of a coding system created and used by the
18       Royal Mail across the United Kingdom for sorting mail.
19         In other words, Postcodes are an abbreviated form of address, and enable a
20         group of Delivery Points to be specifically identified.
21         For the purpose of retaining legacy compatibility no regex based validation is
22         outlined below.
23         If you are validating customer input UK postcodes then the following regex may
24         be used:
25           ^([Gg][Ii][Rr] 0[Aa]{2})|(([A-Za-z][0-9]{1,2})|(([A-Za-z][A-Ha-hJ-Yj-y][0-9]{1,
26           2})|(([A-Za-z][0-9][A-Za-z])|(([A-Za-z][A-Ha-hJ-Yj-y][0-9]?[A-Za-z])))) {0,1}
27           [0-9][A-Za-z]{2}$"
28
29   type: string
```

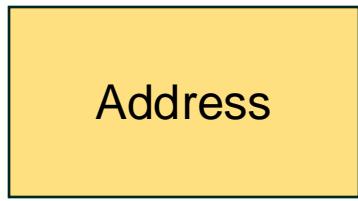
# Customer

```
yaml > customer > customer-domain > types > v1 > ! customer.yml > {} definitions > {} customerId
  1 $id: https://osl-data-dictionary-schemas.engineering.dvla.gov.uk/customer/
    customer-domain/types/v1/customer.yml
  2 $schema: http://json-schema.org/draft-07/schema#
  3 title: Customer
  4 type: object
  5 > examples: ...
115 properties:
116   customerId:
117     $ref: "#/definitions/customerId"
118
119
120
121
122
123
124   customerType:
125     $ref: "#/definitions/customerType"
126   address:
127     $ref: "../../../../address/types/v1/address.yml"
128   emailAddress:
129     $ref: "#/definitions/emailAddress"
130   phoneNumber:
131     $ref: "#/definitions/phoneNumber"
132   individualDetails:
133     $ref: "./individual-details.yml"
134   contactPreferences:
135     $ref: "#/definitions/contactPreferences"
136
137
```

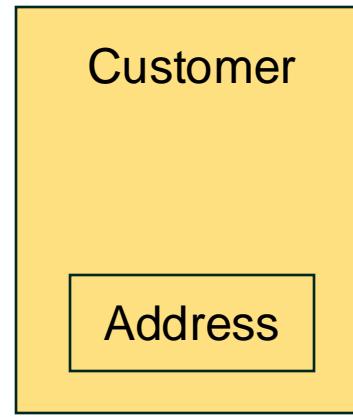


# Example - Address

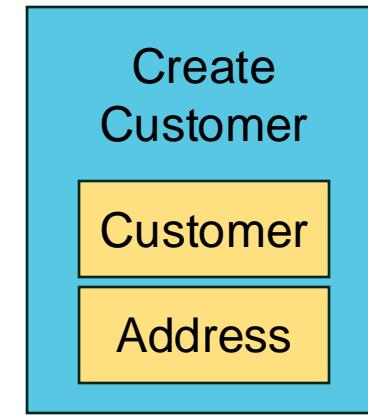
DVLA Digital



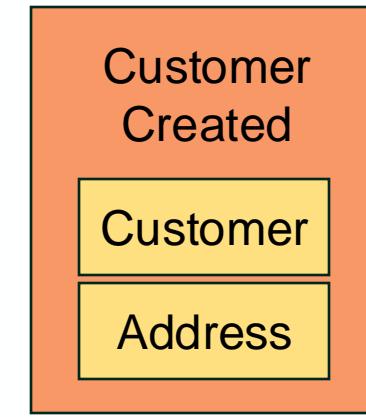
Type



Type



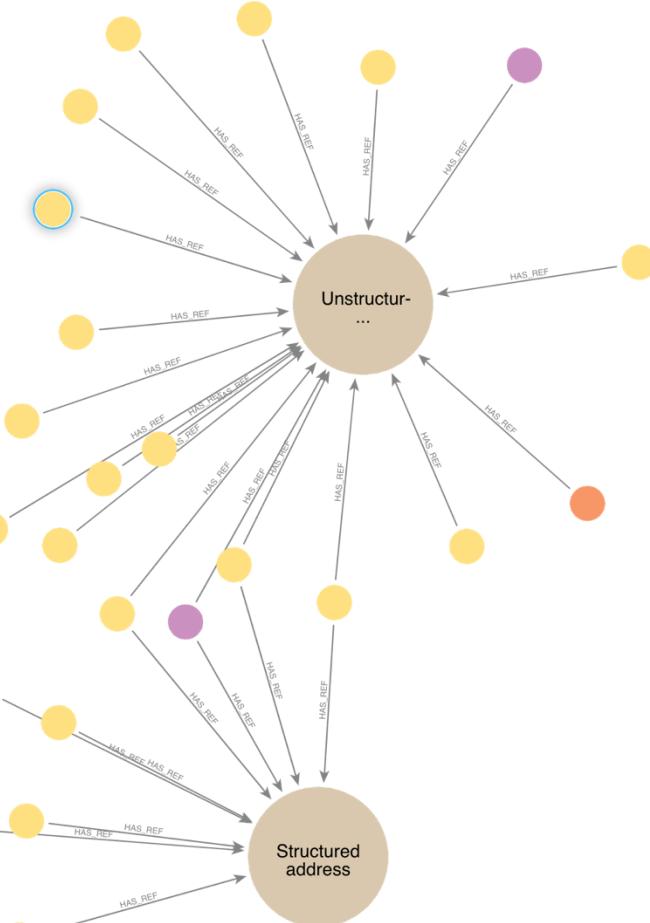
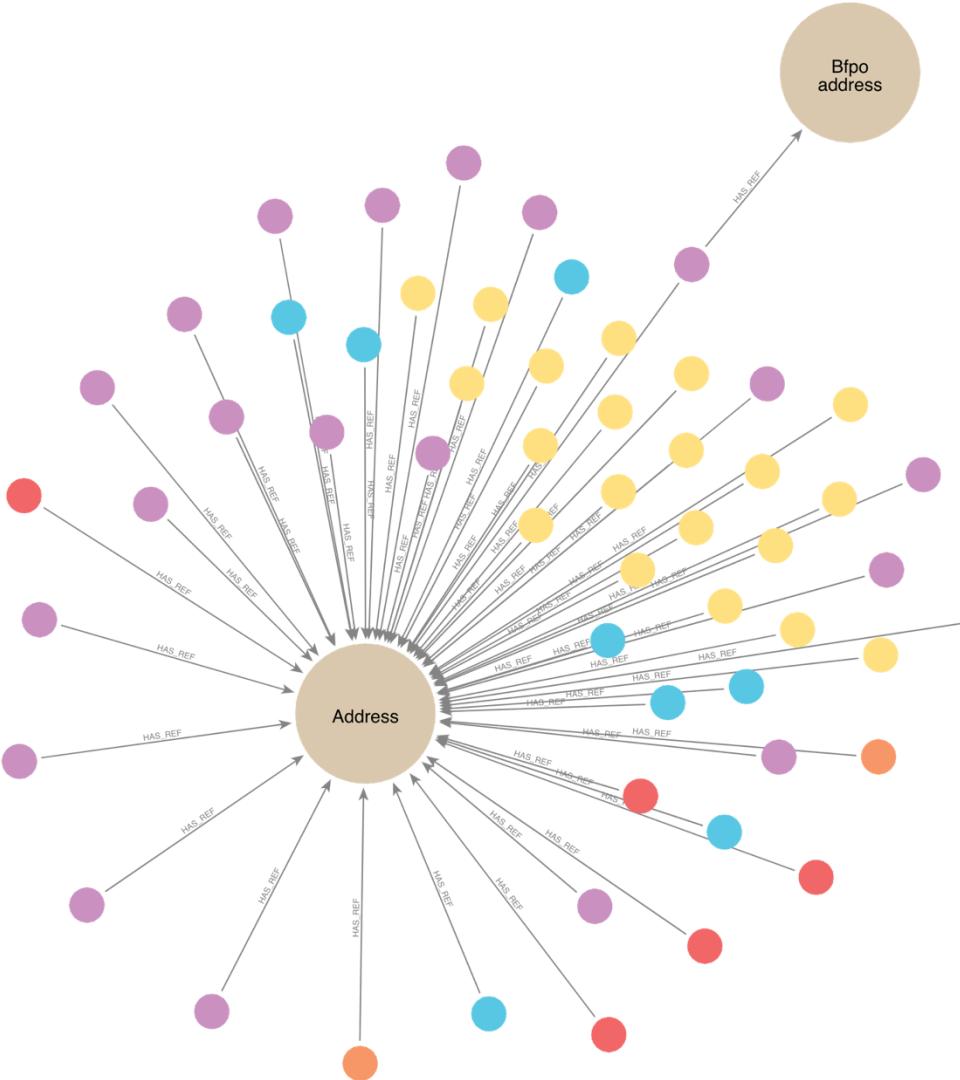
Request / Response



Event

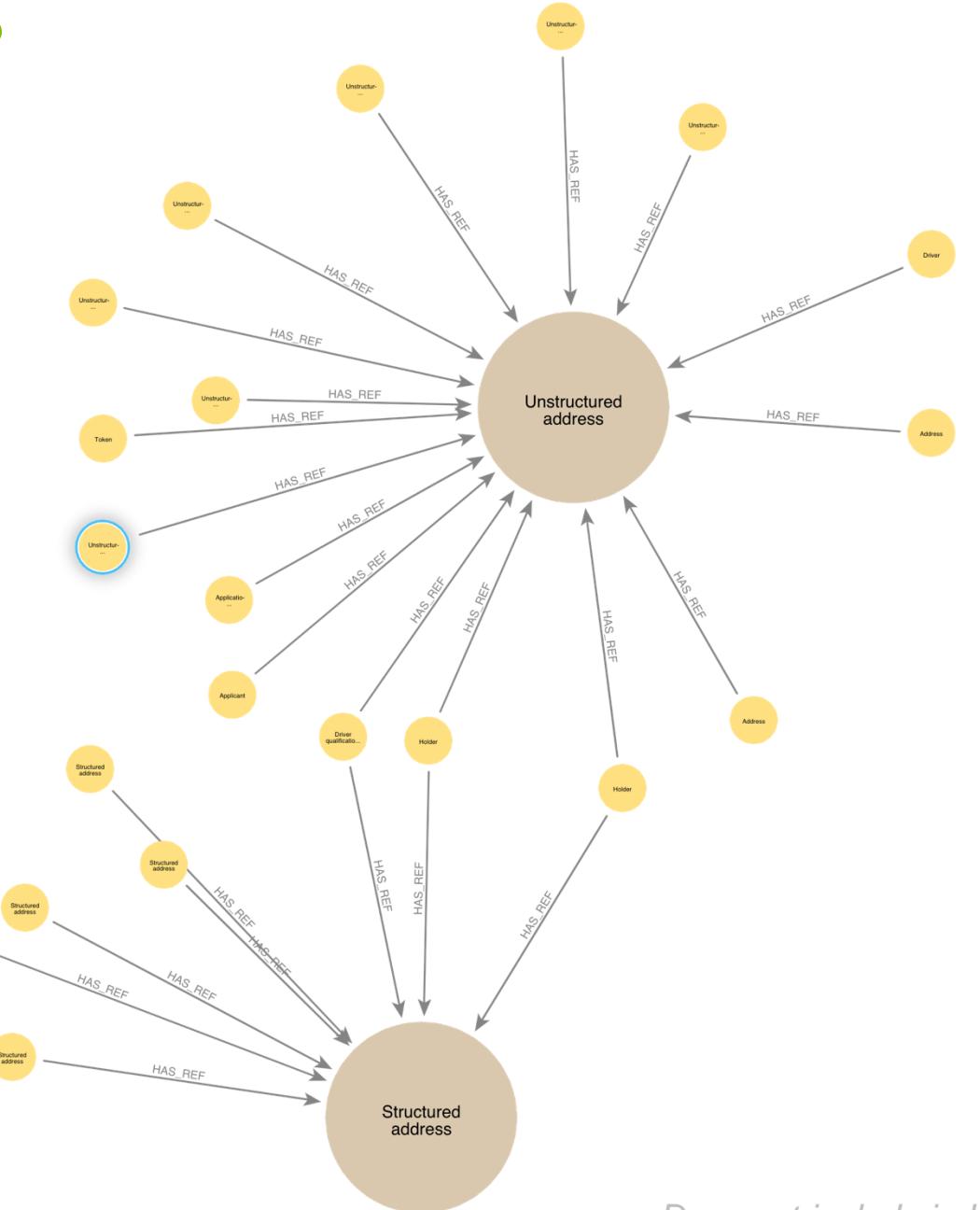
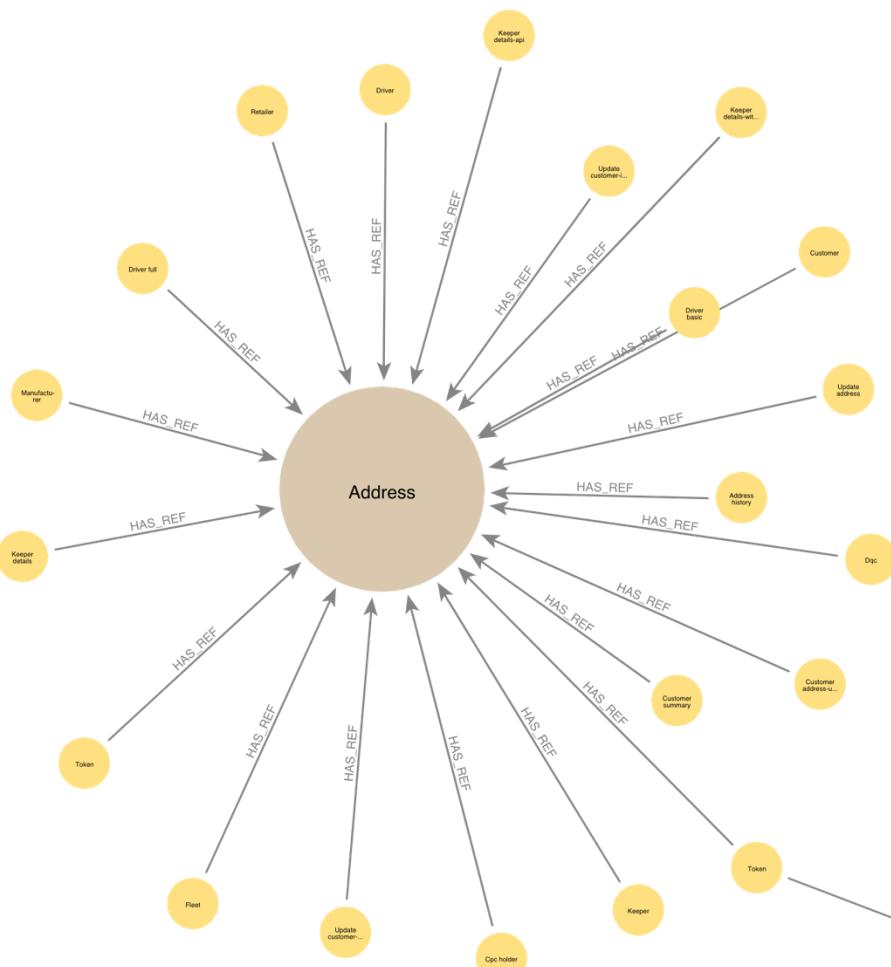


# Address \$ref



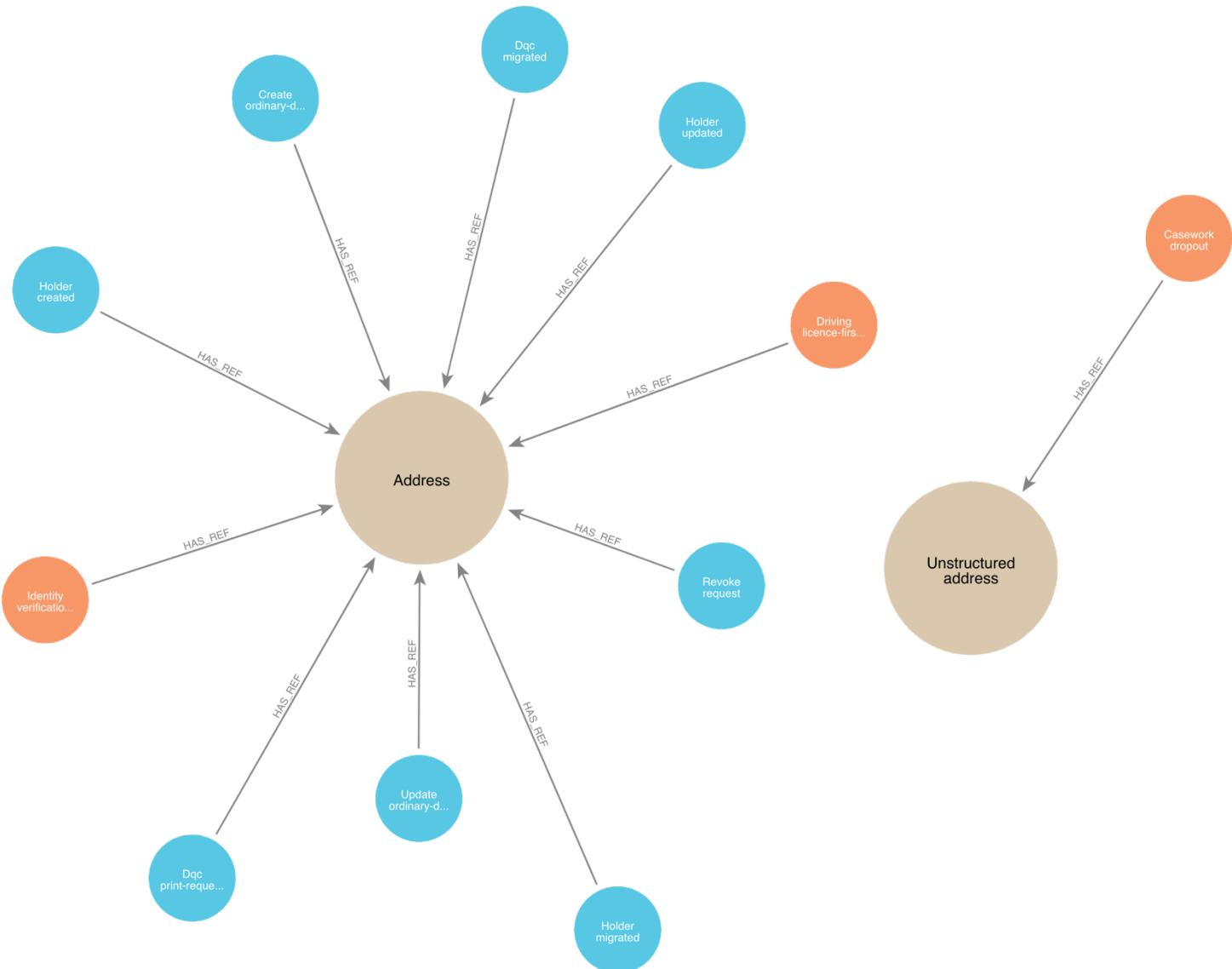
*Does not include indirect references.*

# Address \$ref Data Types



*Does not include indirect references.*

# Address \$ref Events and Messages



*Does not include indirect references.*

# Clearly document references

DVLA Digital

- \$ref becomes a link
- The target schema title is used as a label
- Helps understand composition
- Simple navigation between schema

Customer / Customer Domain / Types / v1 / Customer

## Customer

\$id	<a href="https://osl-data-dictionary-schemas.engineering.dvla.gov.uk/customer/customer-domain/types/v1/customer.json">https://osl-data-dictionary-schemas.engineering.dvla.gov.uk/customer/customer-domain/types/v1/customer.json</a>
\$schema	<a href="http://json-schema.org/draft-07/schema#">http://json-schema.org/draft-07/schema#</a>

## Properties

Name	Type
customerId	String
customerType	String
address	Object (of type <a href="#">Address</a> )
emailAddress	String
phoneNumber	String
individualDetails	Object (of type <a href="#">Individual details</a> )
contactPreferences	Array [ <a href="#">Contact preference item</a> ]



- meta:enum custom keyword

```
yaml > cpc > types > v1 > ! cpc-status.yml > [ ]examples
1   $id: https://osl-data-dictionary-schemas.engineering.dvla.gov.uk/cpc/types/v1/cpc-status.yml
2   $schema: http://json-schema.org/draft-07/schema#
3   title: CPC Status
4   type: string
5   description: The status of the CPC entitlement
6   enum:
7     [
8       CURRENT,
9       REVOKED,
10      SUPERSEDED
11    ]
12  meta:enum:
13    Current: The current CPC entitlement for the driver
14    Revoked: This CPC entitlement for the driver has been revoked
15    Superseded: This CPC entitlement for the driver has been superseded
16
17  examples:
18    - CURRENT
19    - REVOKED
```

Cpc / Types / v1 / Cpc Status

## CPC Status

The status of the CPC entitlement

\$id	https://osl-data-dictionary-schemas.engineering.dvla.gov.uk/cpc/types/v1/cpc-status.json
\$schema	http://json-schema.org/draft-07/schema#

## Example

"CURRENT"

### ► Faker maker examples

\$id	https://osl-data-dictionary-schemas.engineering.dvla.gov.uk/cpc/types/v1/cpc-status.json
Title	CPC Status
Description	The status of the CPC entitlement
Type	String
Enum	<p><b>Current</b> The current CPC entitlement for the driver</p> <p><b>Revoked</b> This CPC entitlement for the driver has been revoked</p> <p><b>Superseded</b> This CPC entitlement for the driver has been superseded</p>
Examples	<ul style="list-style-type: none"><li>• CURRENT</li><li>• REVOKED</li></ul>



# Examples - meta:title & meta:description

DVLA Digital

- meta:title and meta:description custom keywords for examples

```
$id: https://osl-data-dictionary-schemas.engineering.dvla.gov.uk/address/types/v1/address.yml
$schema: http://json-schema.org/draft-07/schema#
$title: Address
$description: A DVLA address entity, which will be one of its child types as described in https://technical.architecture.dvla.gov.uk/utilities/addressing/addressing-common-data-format.html
$examples:
  - meta:title: Structured Address
    meta:description: A structured address based on the PAF format
    $structuredAddress:
      uprn: "10008904551"
      udprn: "4198105"
      subBuildingName: UNIT 6
      buildingName: KISMET PARK
      thoroughfareName: PENARTH ROAD
      postTown: CARDIFF
      postcode: CF11 8TT
      country: Wales
      dps: 1A
      language: EN
```

Address / Types / v1 / Address

## Address

A DVLA address entity, which will be one of its child types as described in  
<https://technical.architecture.dvla.gov.uk/utilities/addressing/addressing-common-data-format.html>

\$id	<a href="https://osl-data-dictionary-schemas.engineering.dvla.gov.uk/address/types/v1/address.yml">https://osl-data-dictionary-schemas.engineering.dvla.gov.uk/address/types/v1/address.yml</a>
\$schema	<a href="http://json-schema.org/draft-07/schema#">http://json-schema.org/draft-07/schema#</a>

## Example

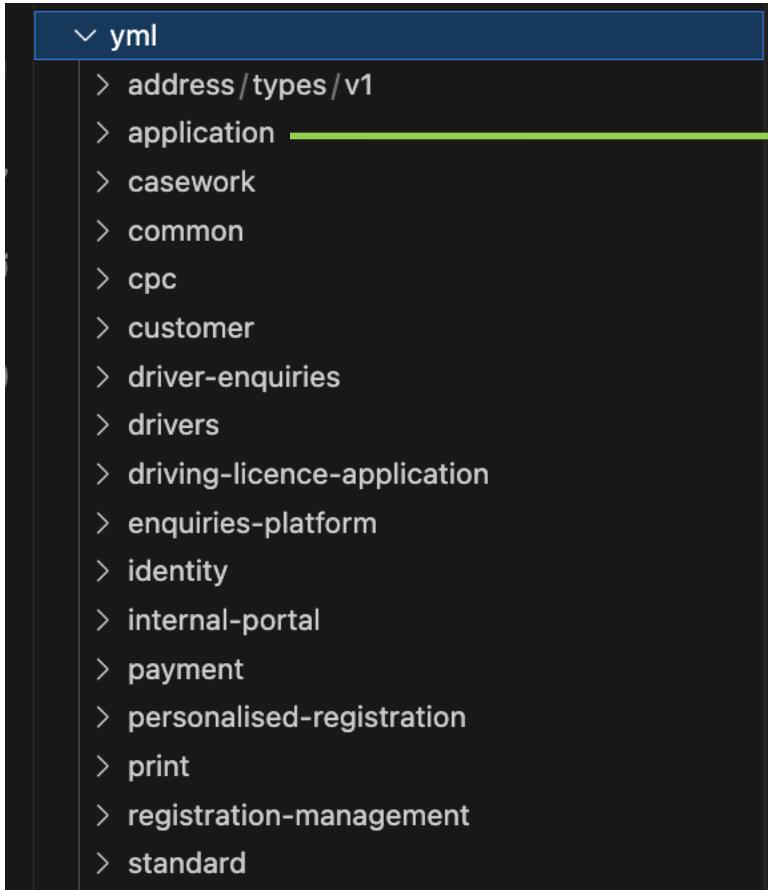
### Structured Address

A structured address based on the PAF format

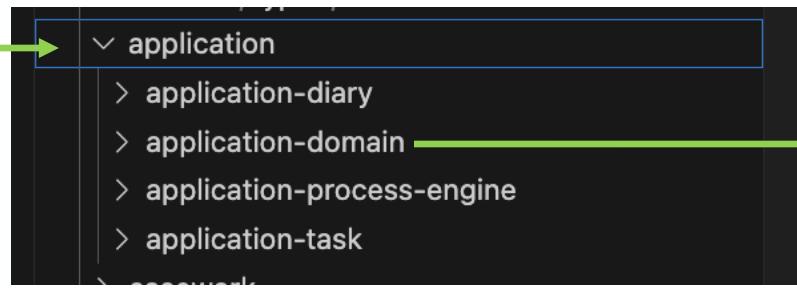
```
{
  "structuredAddress": {
    "uprn": "10008904551",
    "udprn": "4198105",
    "subBuildingName": "UNIT 6",
    "buildingName": "KISMET PARK",
    "thoroughfareName": "PENARTH ROAD",
    "postTown": "CARDIFF",
    "postcode": "CF11 8TT",
    "country": "Wales",
    "dps": "1A",
    "language": "EN"
  }
}
```



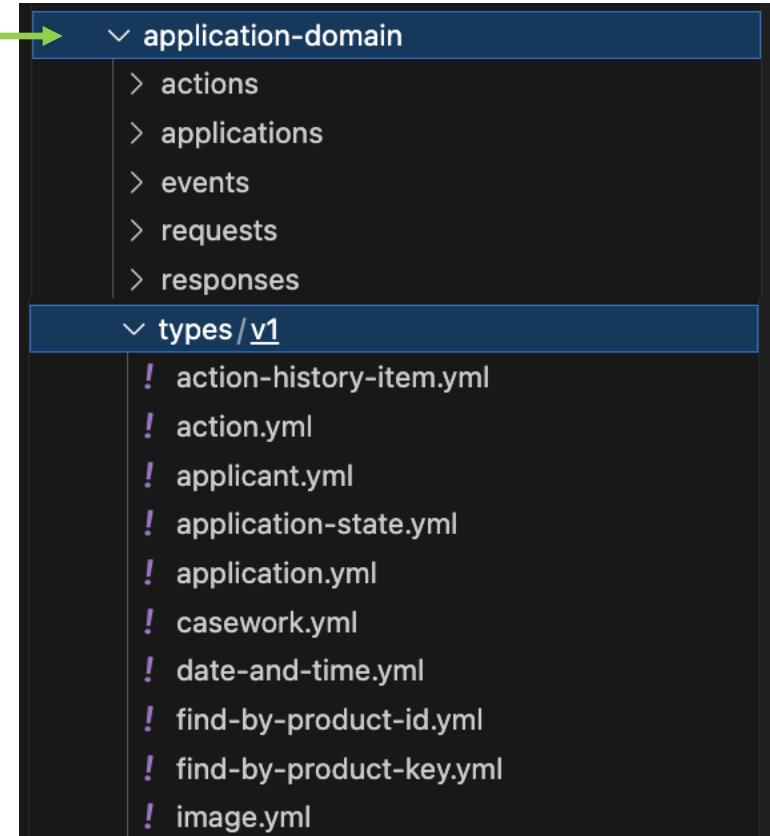
## Products



## Components

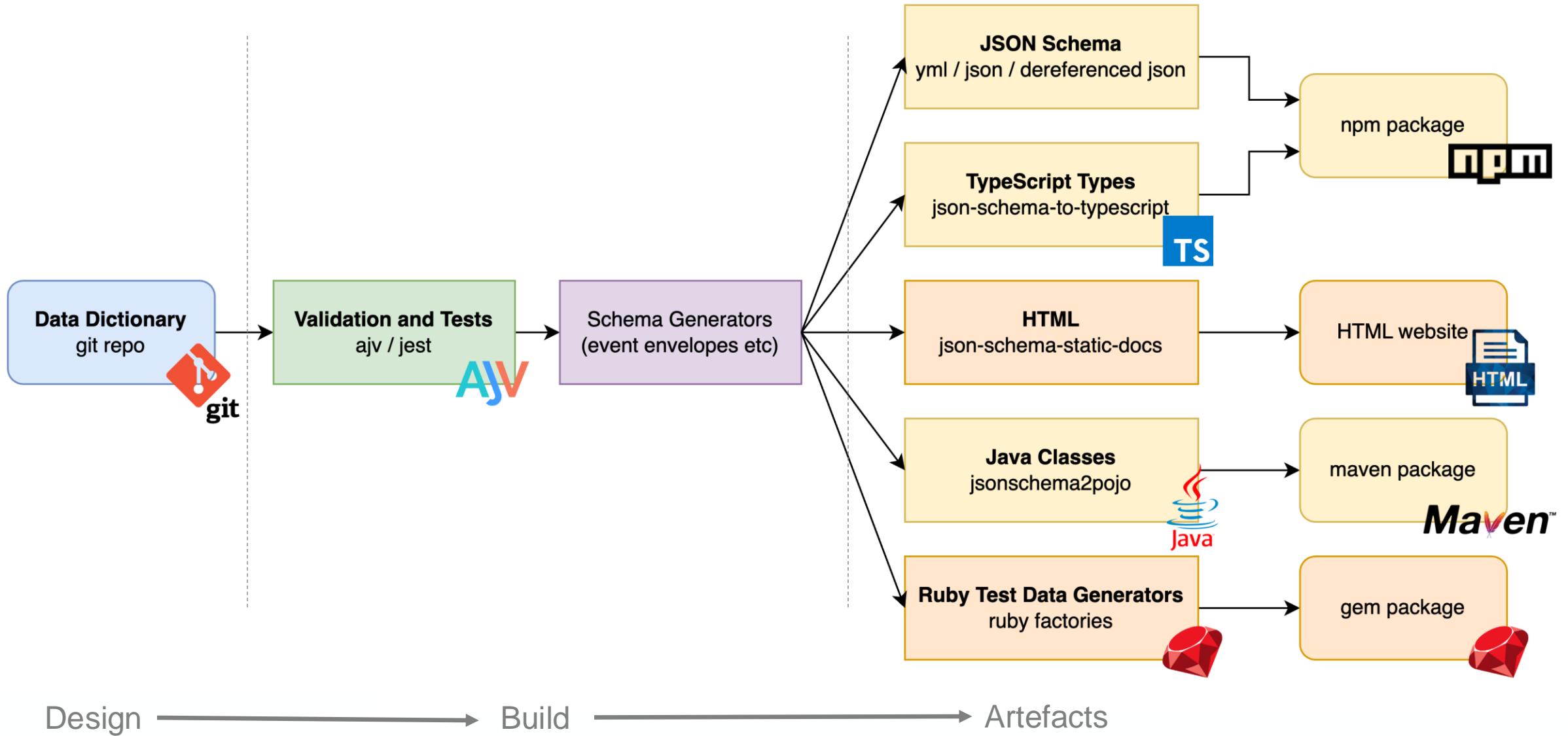


## Schema Categories



# Build Process

DVLA Digital



## Validate

Validate schema against specification

Ensure \$ref values resolve

Validate examples within schema

Ensures team A does not break team B

## Unit Tests

\$id matches DVLA URL pattern

All schema have a title and examples defined

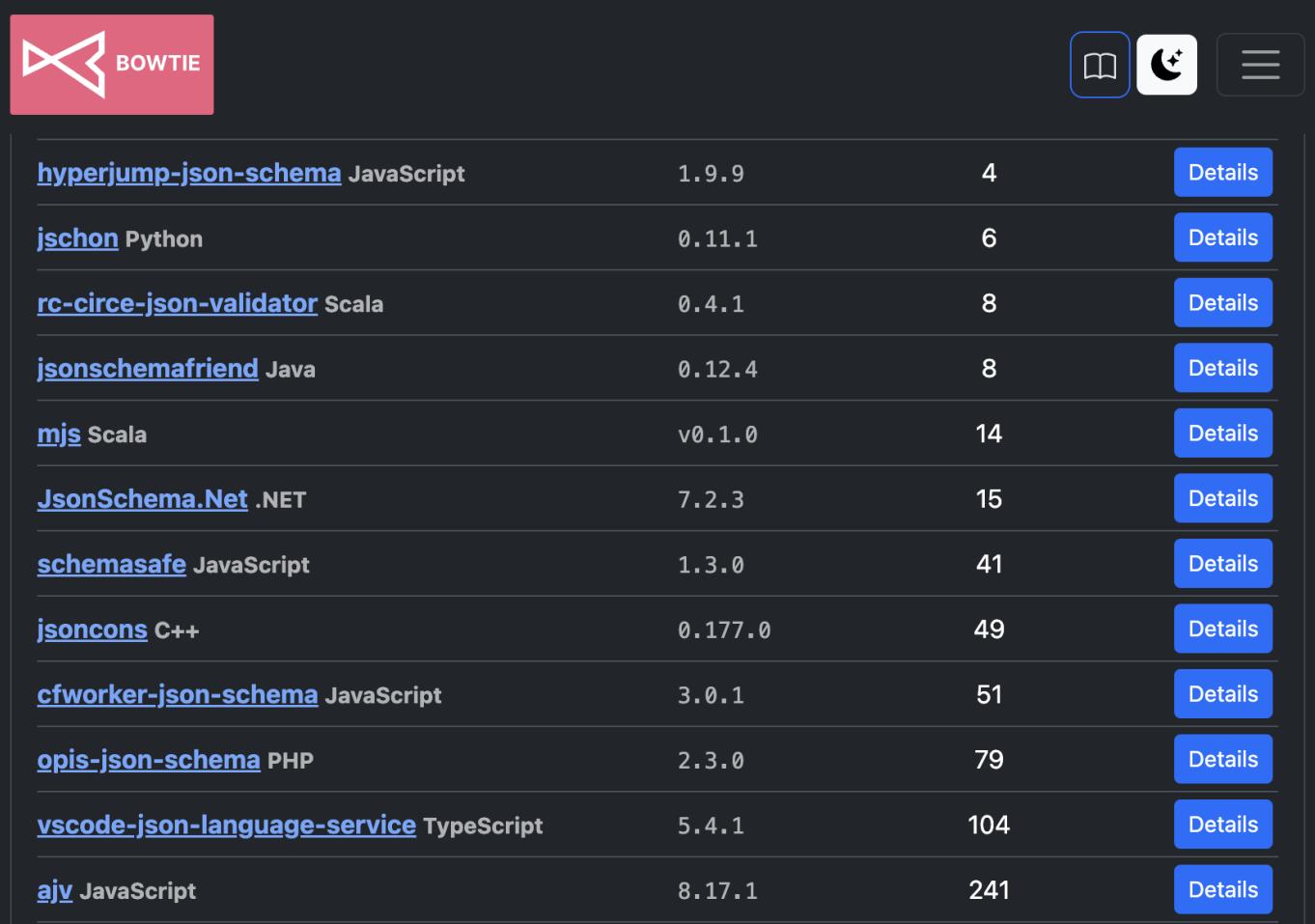
Other internal standards

Verify output of build process



# Build Process

DVLA Digital

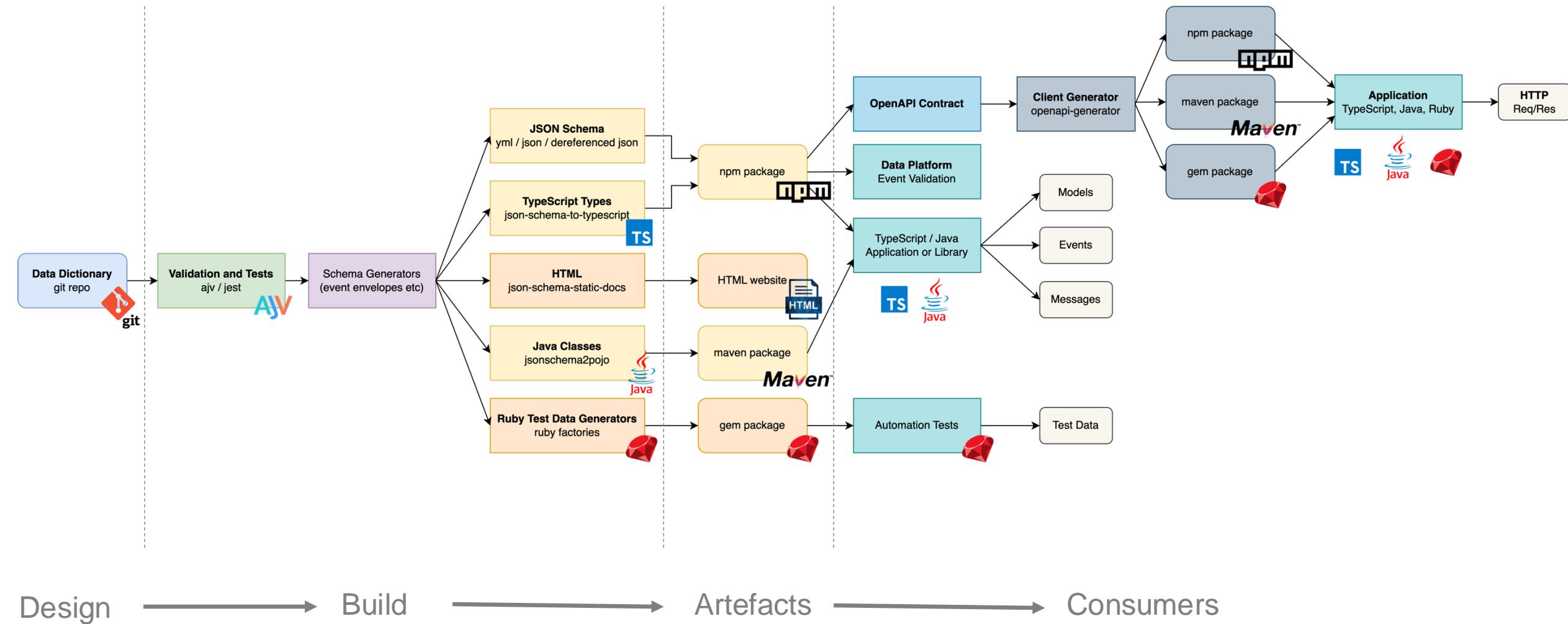


The screenshot shows a dark-themed interface for managing build dependencies. At the top left is the BOWTIE logo. On the right are three icons: a blue square with a white book icon, a white square with a blue crescent moon icon, and a white square with three horizontal lines icon. Below these are two rows of dependency entries, each with a blue 'Details' button.

<a href="#">hyperjump-json-schema</a>	JavaScript	1.9.9	4	<a href="#">Details</a>
<a href="#">jschon</a>	Python	0.11.1	6	<a href="#">Details</a>
<a href="#">rc-circe-json-validator</a>	Scala	0.4.1	8	<a href="#">Details</a>
<a href="#">jsonschemafriend</a>	Java	0.12.4	8	<a href="#">Details</a>
<a href="#">mjs</a>	Scala	v0.1.0	14	<a href="#">Details</a>
<a href="#">JsonSchema.Net</a>	.NET	7.2.3	15	<a href="#">Details</a>
<a href="#">schemasafe</a>	JavaScript	1.3.0	41	<a href="#">Details</a>
<a href="#">jsoncons</a>	C++	0.177.0	49	<a href="#">Details</a>
<a href="#">cfworker-json-schema</a>	JavaScript	3.0.1	51	<a href="#">Details</a>
<a href="#">opis-json-schema</a>	PHP	2.3.0	79	<a href="#">Details</a>
<a href="#">vscode-json-language-service</a>	TypeScript	5.4.1	104	<a href="#">Details</a>
<a href="#">ajv</a>	JavaScript	8.17.1	241	<a href="#">Details</a>



# End-to-end process

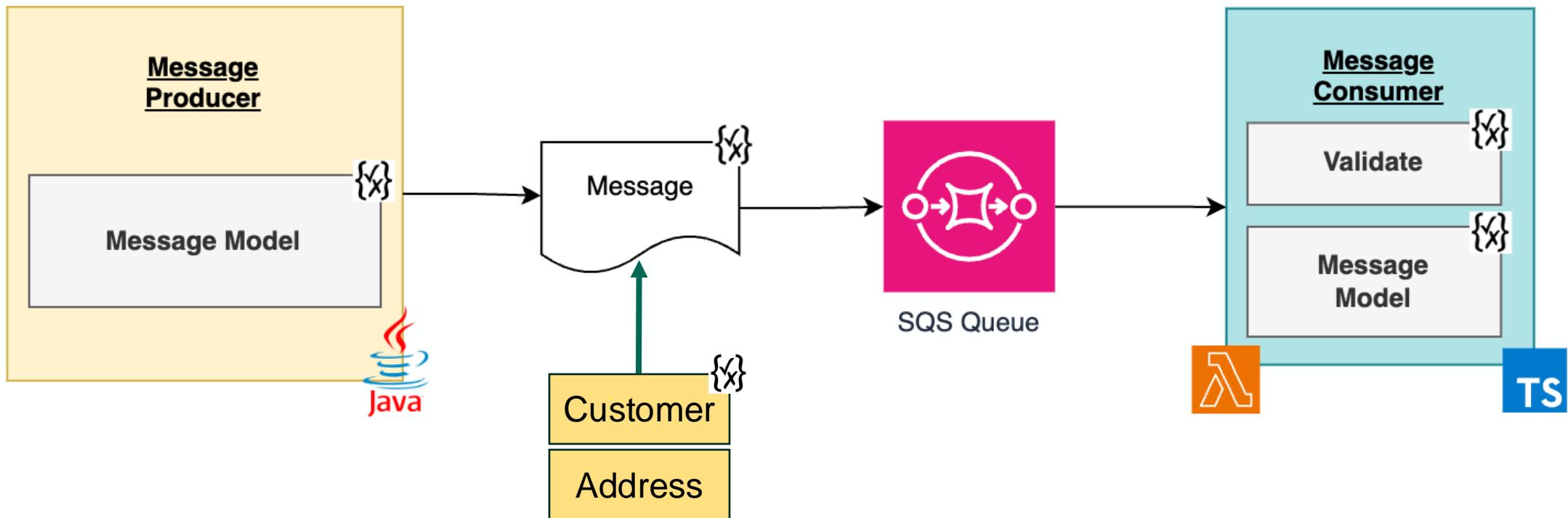


# Consumers

```
/v3/conversions/unstructured:  
  post:  
    operationId: POST-Address-Conversion-Structured-to-Unstructured  
    summary: Convert a DVLA Structured address to a DVLA Unstructured address, deprecated by new generic v4 endpoint  
    deprecated: true  
    parameters:  
      - $ref:  
      - $ref:  
      - $ref:  
    requestBody:  
      description: >-  
        Structured address to convert. At least one of the following address  
        fields must not be empty: poBoxNumber, organisationName,  
        departmentName, subBuildingName, buildingName, buildingNumber,  
        dependentThoroughfareName, thoroughfareName, doubleDependentLocality,  
        dependentLocality, postTown.  
      required: true  
      content:  
        application/json:  
          schema:  
            $ref: '...../node_modules/osl-data-dictionary-schemas/dist/json-dereferenced/address/types/v1/structured-address.json'  
    responses:  
      '200':  
        description: Converted common display format address  
        content:  
          application/json:  
            schema:  
              $ref: '...../node_modules/osl-data-dictionary-schemas/dist/json-dereferenced/address/types/v1/unstructured-address.json'
```

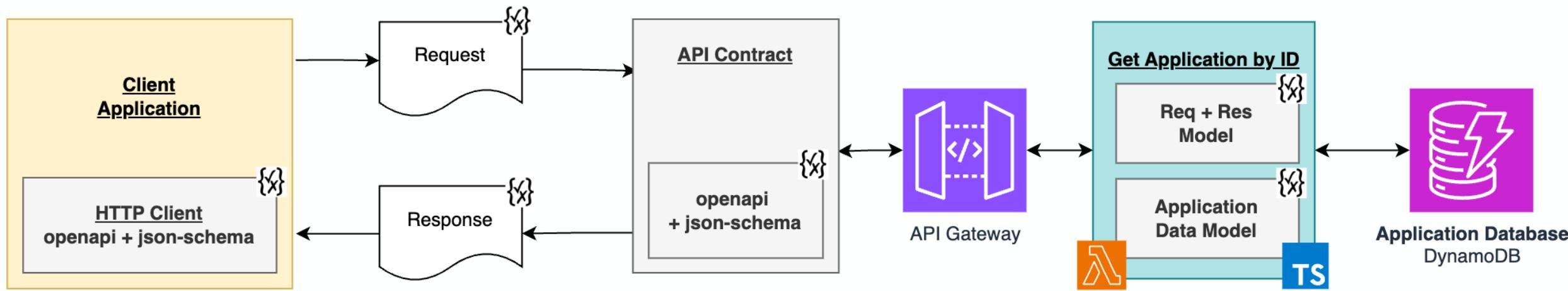
# Use Case – Messages

DVLA Digital



# Use Case – Client / Server

DVLA Digital



# Code Generation

DVLA Digital



```
customerNumber:  
  title: Customer Number  
  type: string  
  description: a human readable identifier that doesn't change over the life of the Customer  
  ...
```



```
export interface Customer {  
  /**  
   * a human readable identifier that doesn't change over the life of the Customer  
   */  
  customerNumber: string;
```



```
FakerMaker.factory(:customer) do  
  customer_number(json: 'customerNumber', required: true) { Faker::Lorem.word }
```



```
@Generated("jsonschema2pojo")  
public class Customer {  
  /**  
   * Customer Number  
   * <p>  
   * a human readable identifier that doesn't change over the life of the Customer  
   * (Required)  
   *  
   */  
  @JsonProperty("customerNumber")  
  @JsonPropertyDescription("a human readable identifier that doesn't change over the life of the Customer")  
  @NotNull  
  private String customerNumber;
```



- JSON Schema is awesome!
- The ecosystem of tooling is powerful
- Take some common data models and make it easy for people to use them across your technology stacks
- Automate your end-to-end process to remove handoffs between the source of truth and your code and data

