GovStack

Building Block Definition

Digital Registries

Developed by: Frank Grozel, Ingmar Vali, Tambet Artma, Saurav Bhattarai, Dr Ramkumar, Rauno Kulla
In cooperation with GIZ, ITU, DIAL, and the Government of Estonia

GovStack

Table of Contents

1 Version History 4

2 Description 4

3 Terminology 6

4 Key Digital Functionalities 7

4.1 Out of Scope Assumptions 8

5 Cross-Cutting Requirements 8

6 Functional Requirements 10

6.1 User Story 1 - Registry Schema User Interface 10

6.2 User Story 2 - CRUD Data in User Interface 15

6.3 User Story 3 - CRUD Data APIs 18

6.4 User Story 4- Registry Schema API 20

6.5 Coverage Map 22

7 Data Structures 25

7.1 Standards/Protocols 25

7.2 Resource Model 25

7.3 Data Elements 26

8 Service APIs 27

8.1 DATA CREATE 29

8.2 DATA UPDATE 29

8.3 DATA UPDATE-OR-CREATE 29

8.4 DATA UPDATE-ENTRIES 29

8.5 DATA LIST (Search) 29

8.6 DATA READ 29

8.7 DATA READ-VALUE 30

8.8 DATA EXISTS 30

8.9 DATA DELETE 30

8.10 DATA My personal data usage 30

8.11 DATABASE SCHEMA READ 30

8.12 DATABASE SCHEMA MODIFY 30

8.13 DATABASE SCHEMA DELETE 31
8.14 DATABASES LIST 31
9 Workflows 31
10 Other Resources 31
11 Key Decision Log 32
12 Future Consideration 38
Appendix 1- Database Read-Schema Response Example 44
Appendix 2- Example Schema with Foreign Keys 54

1 Version History

Version	Authors	Comment
1.0.0	Frank Grozel, Ingmar Vali, Tambet Artma, Saurav Bhattarai, Dr Ramkumar, Rauno Kulla. ingmar.vali@gmail.com	Initial revision
1.1.0	Frank Grozel, Ingmar Vali, Tambet Artma, Saurav Bhattarai, Dr Ramkumar, Rauno Kulla. Reviewers: Neil Roy, Aare Lapõnin	Applied feedback from technical review.

2 Description

This Digital Registries Building Block is a no-code application meant to offer intuitive database/register creation and management functionalities.

The Digital Registries BB´, no-code development platform, uses graphical wizards to create and build software, unlike the traditional approach which uses computer programming languages. It is simple to use, similar to online Excel with advanced data management, log and connectivity options for advanced users. Each register created in the system has a simple User Interface to see and edit data and an API connector with automatically created Open API services for machine-to-machine communication. The Digital Registry System does not contain data capturing and workflow functionality, however if a user interface and data processing is needed then DR can be combined with the Registration BB (see here) as plug-and-play.

It provides services to other building blocks and to external systems, to store and manage data/claims on any entity (persons, places and things), in forms of uniquely identifiable records in a database.

For example, these records could contain health and medical information, ownership of property, vehicles, money, qualifications, birth/expiry of people and entities, land surveys, manufacturing information of vehicles and equipment, banking and commercial transactions, etc. Given the diversity of such information, this building block provides services useful to abstract the structure, linkages and grouping of information into various records and collections such as financial, legal, medical, social, educational, commercial, etc., as needed.

The building block provides capability to capture, store, search, distribute and present data with zero or minimal need for software development. It also maintains and reports logs of all operations taking place on schemas and data. It contains various functional components, data resources to abstract away all the details and complexity and to expose capabilities as service-APIs to external Building Blocks/applications.

The Registries BB is an optional BB for other Govstack BBs that have the need to store information. Any traditional database platform could be used alone or in combination with Digital Registries BB. Digital Registry BB can operate as a standalone service and could be implemented as one centralized instance per domain, containing multiple registries in one instance, or many instances per domain, each database in its own server.

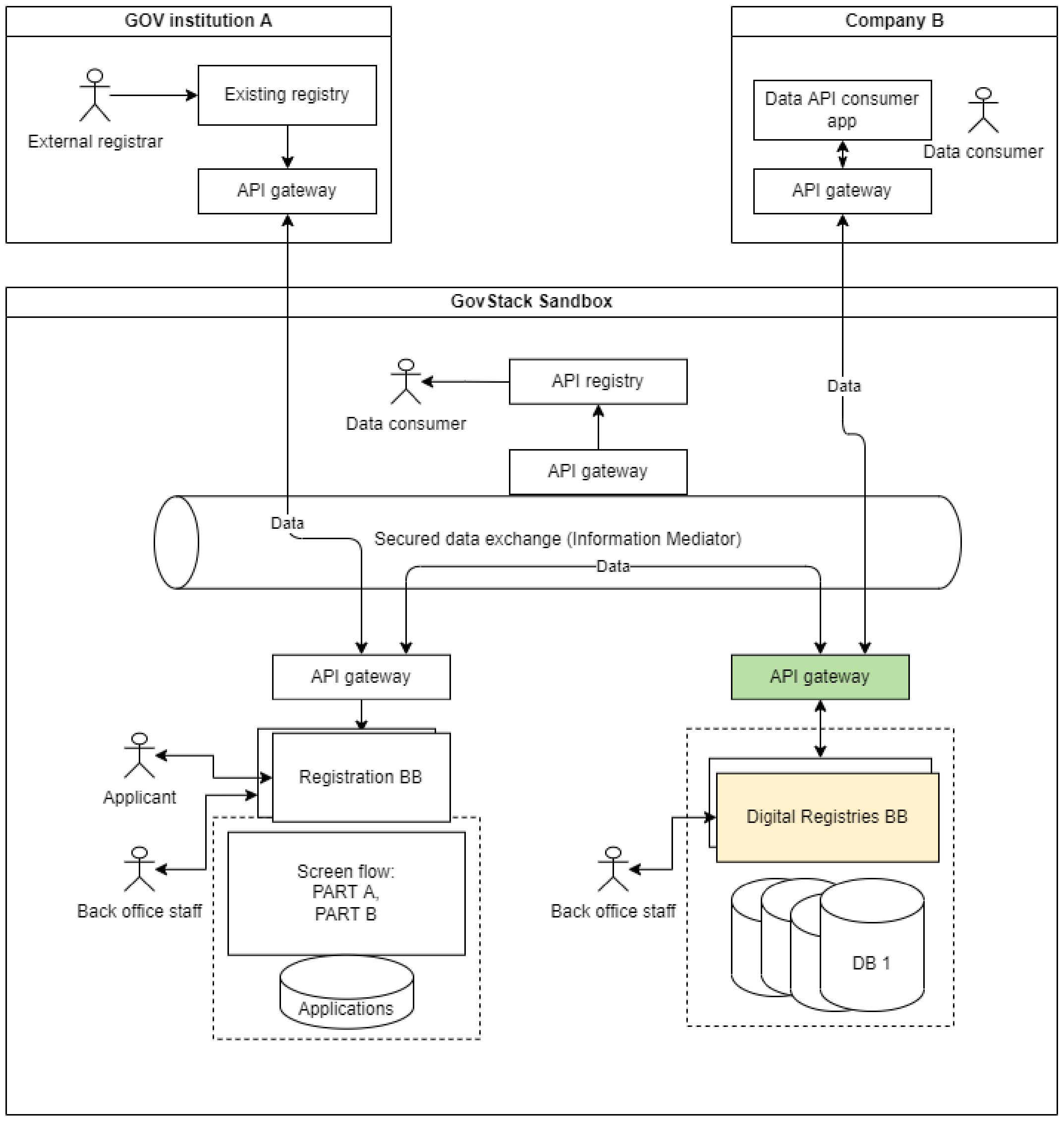


Illustration 1- Digital Registries BB in GovStack sandbox (editable image)

3 Terminology

Registry

A paper-based or electronic database (centralized or decentralized, i.e. blockchain) where claims are stored and can be consulted.

Registration

Process through which an entity gets claims recorded in a registry.

Entity

A thing with distinct and independent existence, such as a person, organization, or device.

Claim

An attribute asserted by an entity, about itself or another entity.

Asserter

An entity that asserts a claim.

Registrar

An entity that is authorized to register, in a registry, claims submitted by an applicant.

Applicant

Entity that requests the registration of claims in a registry.

Operator

A registrar or a staff of a registrar who is processing the request of an applicant.

Administrator/Analyst

A registrar or a staff of a registrar who is building a new registry.

4 Key Digital Functionalities

The Key Digital Functionalities describe the core (required) functions that this building block must be able to perform. These functionalities are described as business processes as opposed to technical specifications or API definitions.

The first user of the BB is an **Administrator/Analyst** who is building a new registry. The Analyst is the person who is building the new registry database, changing the existing database configuration or simply administering the API user authorization. The Administrator/analyst is using a web user interface. The main functions and UI of the BB for Analysts are:

1. Create a new register/database (API or Web user interface);
2. Create and configure schema of the register(API or Web user interface);
3. Change schema configuration and publish new version of the database and API services (API or Web user interface);
4. Enter data to the register (API or Web user interface);
5. View data records in the register (API or Web user interface);
6. Update data in the register (API or Web user interface);
7. Import/export data from/to external files;
8. Import/export registry database schema;
9. Create API services;
10. View statistics(API or Web user interface);
11. Inspect transaction log of registry data operations (API or Web user interface).
12. Manage access to registry data. Authorize users to see and edit registry records or data field (ABAC based access management).
13. Share data with other users via e-mail, or via unique and secure URL. sharing can be field level or record level.

The second main user is an **Applicant** who is consuming registry data via other BB (e.g. Registration BB) screen flow or via Information Mediator BB API services. The main functions of the BB for Applicants are:

1. Search data from the register (API service);
2. Read/pull data from the register (API service);
3. Create data in the register(API service);

4. Update data in the register(API service);

5. Delete data in the register (API service);

6. Validate if given content exists in specified register (API service);

7. Read statistics (API service).

The BB has a user interface to query and consult the registry data but in most cases the Applicants are using the end client applications like Registration BB to access the registry. Any building block can query data from Digital Registries BB via APIs if authorization is given.

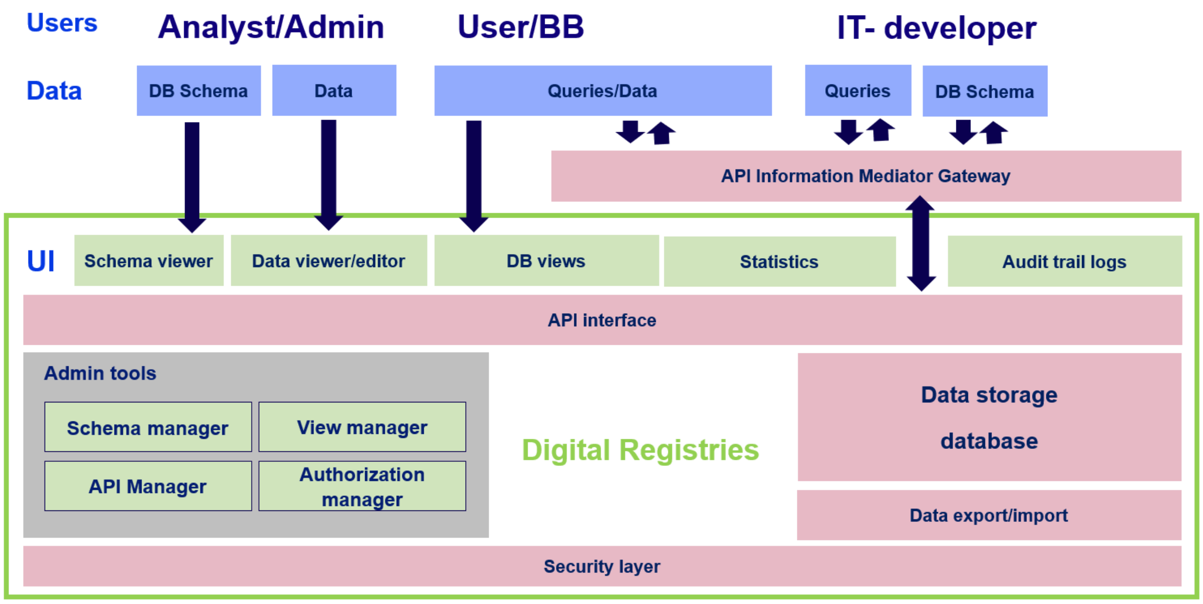


Illustration 2- Digital registries functional components

4.1 Out of Scope Assumptions

- Distributed database architecture; Blockchain solutions; integrations with distributed architecture solutions.
- Automated data migration from Digital Registries solution to external databases.
- Event notification to external endpoints.

5 Cross-Cutting Requirements

The Cross-cutting requirements described in this section are an extension of the cross-cutting requirements defined in the Architecture specification document (here) and Security requirements (here). This section highlights cross-functional requirements for Digital Registries BB and in addition describes any deviation to the Architecture BB cross-cutting requirements.

The following requirements should be optional:

Citizen-Centric (2.1 in Architecture Blueprint)

- *Right to be forgotten: everything must be deletable*. **- not a good practice for gov. registries.**

Open (2.2 in Architecture Blueprint)

- *Cloud native (Docker and Kubernetes).* ***- must have also on-site installation option***.

Robust (2.7 in Architecture Blueprint)

- *Operates in low-resource environments:*
    - *Occasional power* - ***not possible, should be optional. This can be solved with UPS and generator that keeps the systems running without interruptions.***
    - *Low-reliability connectivity* - ***Client-server systems are not reliable in this situation.***

Databases MUST not include business logic (3.21 in Architecture Blueprint)

- *This means no triggers/stored procedures shall be used.* - some stored procedures may be needed FOR database record ID generation.

The following requirement should be added to other BB cross cutting requirements:

Privacy and protection of user data

Each owner of the personal data (e.g. citizen) must be able to see who has looked at their personal data in the registry. All captured personal user data must be marked as "personal data". Users can make requests to see the information/logs of accessing personal information. API must be available for authenticated users to see their own personal data audit logs.

6 Functional Requirements

The functional requirements section lists the technical capabilities that this building block should have. These requirements should be sufficient to deliver all functionality that is listed in the Key Digital Functionalities section. These functional requirements do not define specific APIs - they provide a list of information about functionality that must be implemented within the building block.

6.1 User Story 1 - Registry Schema User Interface

As an Administrator/Analyst I want to use a web user interface to create a register database (example registry use case - social security program), so that I can configure and launch the registry database instantly to be used by internet users and client systems (e.g. Registration BB, Information Mediator BB) VIA web interface and API.

Actors: Analyst - An administrator user who is creating / changing registry database schema.

Main actor/user in these requirements is the Analyst.

Preconditions:

1. User is authenticated.
2. User is authorized as an admin;
3. The user interface is a web interface;
4. User has internet;
5. System has electricity.

Process:

1. Create a new registry database project.
2. Define the database fields.
3. Publish the database.
4. Validate/configure the API services.
5. Manage user rights to access the database and APIs.

Post conditions:

1. System contains a database that is ready to process new data.
2. System has API services to CRUD data (and API to validate if data exist).
3. User can enter data to the registry via web UI.
4. User can see log information in the UI.
5. User can see statistics in the UI.
6. User can give authorization to use the database and process data.

REQ-#	Requirement	Type/UC-nr
DRS-1	Analysts must have the option to create a new registry database by filling the following information: 1. Name of the database; 2. A short name; 3. Schema of the database (see DRS-3).	Must have UC1
DRS-2	Analysts can create multiple databases into one system instance. Databases must be linkable with foreign keys. See foreign key API description example in Appendix 2. Analysts can configure which databases and which fields are linked in the user interface. In this document and foreign key function we consider databases as database tables that can be linked with one another. See example illustration here. User story: As a user I can browse database content (Data) in the UI and when databases are linked, then I can click and move from one database to another where the corresponding linked data will open in the UI. In the Digital Registries Data user interface, it must be possible to open another database by clicking on the record ID in one database and all corresponding records from the other Database will open. It is required to have at least two levels of ID's (database ID and field ID) to link the databases. See example API below in Appendix 2. Example: In one registry database we store information about Mother and Child record. In the second registry database we store information about payments made for the mother. System must enable a foreign key link between the payment database to the Mother and child record database. Users can click in the payment database record UI to the Mother ID field and the system UI must open the corresponding record in the Mother and Child database.	Must have

DRS-3	Analysts must have the option to add fields to the database schema.
	
	Fields of the database should contain at least the following elements:
	a. Field name;
	b. Field type, at least with following types:
	i. Text;
	ii. Number;
	iii. Boolean;
	iv. Date;
	v. Date/time;
	vi. File (pdf, doc etc). File extensions/types must be
		Must have
		
		UC1

REQ-#	Requirement	Type/UC-nr
	configurable; vii. Edit grid (sub-table inside a field); viii. Block container (to group fields visually); ix. Catalog (holding value and key). c. Field advanced properties (mask, format, read-only, password (blinded), PersonalData, enum list selection); d. Validation options- Required Unique, max, min. e. Triggers (to create ID-s, merge fields, condition logic, transform-upper/lower case); f. Foreign keys (to link other databases in the same ecosystem). See example schema in Appendix 2.	
DRS-4	Analysts must have the option to publish the database. Publishing will reveal the database to users. ● Publish uses versioning. Every publish creates a new version of the database schema and API services; ● Old database schemas must be available to the users; ● Data stored in the old database versions must be usable in old versions and in new versions; ● Analysts can delete database schema versions. Same version API services must be deleted at the same time.	Must have
DRS-5	Analysts must be able to configure the API services per registry database. ● The system automatically creates API services to: ○ create data; ○ read data; ○ update data; ○ delete data; ○ validate data (if exists); ○ update or create data. ● Analysts can hide API services; ● Analysts can delete API services; ● Analysts can copy API services; ● Analysts can create custom API services; ● The system generates the API data structure from the dynamic database structure automatically each time a publish is done.	Must have

REQ-#	Requirement	Type/UC-nr
	<Example UI>	
DRS-6	Authorization to: a. create and manage databases; b. API usage per service, per record, per data field; c. access to DATA. Analyst must have the option to manage user rights of a database and data via API and via UI. ● Attribute Based Access Control (ABAC) logic could be used (API, Schema, data fields, record filter, users); ● Anonymous user role must be available; ● Any logged in user role must be available; ● Per user, per group of users option must be available. ○ Group is a set of users in a role. ● Role is a set of rights. Authorization/user rights are stored in addition in the central GovStack IAM system- read more here.	Must have
DRS-7	The system must log all data processing in the database. ● Schema changes must be logged; ● Data processing (CRUD) must be logged; ● Logs must be visible and searchable to the Analyst via UI; ● Every data owner (e.g. physical person) must have the option to see who has processed his/her data (PersonalData). The function is standard function for all registries. See more (DRS-14 API example) ● Change logs are protected with highest level of integrity (chaining of logs) ● Database logs could be logged with external blockchain(Nice to have). See more Audit Logging requirements here.	Must have UC3
DRS-8	Personal Data usage. System must automatically store all data read requests and store these in log table. ● Covers data read events via UI and via APIs. ● Personal Data logs are stored with PersonalData data tag, storing	UC3

REQ-#	Requirement	Type/UC-nr
	at least the following information. ◦ Log ID; ◦ Data record ID; ◦ Field ID; ◦ PersonalDataID(unique and unchangeable identifier of a person); ◦ Reader ID- who read the data; ◦ Reader name- name or initial of a person; ◦ When- the moment when the Personal Data was read. • Personal Data report is visible only for Analyst to see all data read logs and Data Owners (physical person) to see their own personal data usage log. Input is PersonalDataID field. • PersonalData report is usable as API service (read) • System must have API for PersonalData reports. API is per registry(database) • System must log Personal Data log read events to the log table.	
DRS-9	Analysts must be able to create views of a database. • View is a selection of data from a database; • View can be opened as OPEN DATA (anonymous user); • View can be created and it can be as a base for an API service (Custom API); • View is not for changing or deleting data, only for reading; • View rights are managed by the user rights management system.	Optional
DRS-10	Must have the option export/import database schema to JSON file, (optional: xls file).	Must have
DRS-11	Import DB structure • Must have the option to import database structure from JSON file • Must have the option to import database structure from XLS file	Optional
DRS-12	Service usage statistics • System must record all API service usage information.	Optional

REQ-#	Requirement	Type/UC-nr
	• System must record all searches made in the Registry UI and via APIs.	
DRS-13	Analyst must be able to mark a field as PersonalData log object- This field contains personal data.	UC3
DRS-14	Analyst must be able to mark a field as PersonalDataID. This is the data owner's ID.	UC3
DRS-15	Analyst must be able to mark a field as secret- This field contains secret data (credit card number). E.g. secret data (card data) must be encrypted while at rest. Information in transit between the BB-s is secured with encryption. Information in Transit is described and governed by Information Mediator BB.	Must have
DRS-16	Analyst must have the option to read database schema in the web UI.	Must have

6.2 User Story 2 - CRUD Data in User Interface

As an Administrator/ Analyst I want to process (CRUD) registry data so that I do not have to know the query language.

Actors:

- Analyst- main actor in these requirements is the Analyst/administrator.
- Data owner- is a physical person whose personal data is stored in the registry.

Preconditions:

1. Analyst is authenticated and authorized to use the BB and process data in the database.
2. The user interface is a web interface.
3. User has internet;
4. System has electricity.

Process:

1. Analyst searches a record via search or filter function.

2. Analyst selects a record.
3. Analyst processes a record.
4. System stores changes to the Change Log database.

Post conditions:

- Processing changes done by analysts are done and log for change is created.

REQ-#	Requirement	Type
DRS-17	Analysts must have a view to see all data in the registry. Two main views: ● Main registry records grid view. ● Record detail view. ○ See data; ○ See documents(open if image, download if other type); ○ Data log view (changes (create, update, delete). Data before and after). ○ Data read view (information about who has looked the data, /exported data). Data and data readers information is stored in the log registry.	Must have UC2 UC3
DRS-18	1. Analysts must have a view to edit data in the registry. Two main views: ● Main grid (inline editing); ● Detail record edit view. ○ Edit data ○ Remove/add documents (upload) ● All data changes are logged. 2. Analyst must have option to delete data in the registry. ● All data changes are logged.	Must have
DRS-19	Analysts can use additional functions to simplify data searching: ● Filtering by search criteria by field content	Must have

REQ-#	Requirement	Type
	• Full text data search • Order by each data field	UC2
DRS-20	Import data to the registry Analyst must have an option to import information to the database. Import formats are: json, csv, xls.	Must have
DRS-21	Export data from the registry Analyst must have an option to export selected/filtered data from a registry to CSV, XLS, Json.	Must have UC2
DRS-22	Statistical queries. The system should have the ability to: A. Produce standard statistical reports a. System must show statistics of all registered items in the registry, with various criteria for filtering. For example: i. Details of registered people ii. Details of registered services iii. Time series: Change in registration of people/services over time iv. Details of change to data elements (audit logs) b. Generate customizable reports based on the fields registered in the registry. B. Allow the analyst/user to analyze data collected in the system in various ways: a. (Option) Develop functionality to allow custom dashboards for analysts to analyze data within databases b. Provide APIs for extracting data from databases to analyze in external data analytics systems (eg. Tableau)	Must have
DRS-33	Users can share data with other users. Share data with other users via e-mail, or via unique and secure URL. Sharing must be record level and field level. Data sharing can be turned off in authorization module. Data can be shared to anonymous users with URL.	May have

6.3 User Story 3 - CRUD Data APIs

As an Applicant I want to process (Create, Read, Update, Delete) data in the registry database.

Actors:

- Applicant - Main actor in these requirements is an applicant via client system. In this use case applicant is any user who is using a client system (Registration BB). For example, a Health Care worker is an applicant in this user story; a mother, using the Registration BB. Example client system in this document is Registration BB.

Preconditions:

1. Applicant is using client system (e.g. Registration BB) that is connected to Information Mediator BB;
2. Client system has been given authorization to access Registry to process (CRUD) information;
3. Applicant has been given authorization to access Registry to process (CRUD) information;
4. Applicants are registered in the system and able to use authentication. Applicant is Authenticated by client system or Security BB (Authentication).
5. Applicant has internet;
6. System has electricity.

Process:

1. Applicant uses a client system to process data in the registry
    a. Applicant can create data;
    b. Applicant can read data;
    c. Applicant can update data;
    d. Applicant can delete data;
    e. Applicant can create or update data;
    f. Applicant can validate data.
2. System logs all processing events in the dedicated audit registry;

Post conditions:

1. When Applicant is authenticated by a client system (e.g. Registration BB) the registry allows processing (CRUD) information from the registry. All users who are authenticated can read

data.

2. When a user is not authenticated in the system, the system allows to process (CRUD) data from all databases where an anonymous user has been allowed to process data.

3. When a user has no authorization, one can not process (CRUD) any information in the registry.

REQ-#	Requirement	Type
DRS-23	BB must enable client systems to process (CRUD) the database records via Open API services. ● Applicant can search data; ● Applicant can create data; ● Applicant can read data; ● Applicant can update data; ● Applicant can delete data ● Applicant can create or update data BB authorizes client systems and users to process data.	Must have UC1;UC2
DRS-24	BB must have an Open API service list (Swagger) to visualize all API services and API service versions. ● Client systems must be able to see all API service descriptions including: ● Description of each field ● Example data of each field. If possible then the example must be real so that whoever is looking at the API specifications can test the example data in the service (try it).	Must have
DRS-25	System must have an API for PersonalData usage report. ● API input must be configurable by the analyst. Input must be a unique identifier of the data owner(e.g. personal identification number). ● If registry database schema is designed to store personal data then the analyst must be able to link the personal data to the owner of personal data (e.g. citizen).	Must have UC3
DRS-26	Statistical queries via API. ● System should make data accessible through the API ○ Registration Data	May have

REQ-#	Requirement	Type
	○ Programme Data ● API should allow querying data with multiple parameters ○ Date, time ranges ○ Registered Program ● Only authorized data should be available through the API.	
DRS-27	Using viewing event logs- every data owner has the right to see who has looked at their personal data. ● **Data owner** is a physical person whose personal data is stored in the registry. ● Data owner has the right to access data reading/processing event logs of the personal data they own. Personal data in a registry is marked accordingly (PersonalData) by the analyst. ● PersonalData logs are visible via API or via user interface (PersonalData report).	Must have UC3

6.4 User Story 4- Registry Schema API

As an IT-developer I want to Create/update/delete registry database schema via API services.

Actors:

- IT-developer (Developer)- Main actor in these requirements is planning to open a new business program and web form to capture applicants data. Captured data must be registered in the registry. In this use case a Developer is any user who is using API services to create and manage registries database.

Preconditions:

1. Developer is using API with a client system or a script that is connected to Information Mediator BB. Client system is any BB that is using API services via IM;

2. IT-Developer (IM organization) has been given authorization to Create/update/delete database schemas via API services.

3. Developer has internet;

4. System has electricity.

Process:

1. Developer uses a client system to edit registry database in the BB

a. Developer can create database/schema;

b. Developer can read database schema;

c. Developer can modify database schema;

d. Developer can delete database schema and all data in it.

Post conditions:

1. When Developer is authorized to use BB API then the Digital Registries BB allows processing (CRUD) schema of a registry. All authorized users can read/create/update/delete database schema;

2. When Developer is not authorized to process/CRUD the database schema, the system allows to process schema of all databases where an anonymous user has been allowed to edit database schema (simplification for GovStack Sandbox instance);

3. When a user has no authorization, one can not create nor change (CRUD) any schema in the BB.

REQ-#	Requirement	Type
DRS-28	Developer must have the option to create a new registry database by sending data via API: 1. Name of the database; 2. A short name; 3. Schema of the database (see DRS-3). See full list of example API descriptions /database/modify here.	Must have
DRS-29	Developer can create multiple registry databases into one system instance.	Must have
DRS-30	Developer must have the option to publish the database. Publishing will reveal the database to users.	Must have
DRS-31	Developer must be able to modify API services per registry database. ● The system generates the API data structure from the dynamic database structure automatically each time a publish is done. ● The system automatically creates API services to: ○ create data; ○ read data; ○ update data; ○ delete data;	Must have

REQ-#	Requirement	Type
	○ validate data (if exists); ○ update or create data. ● Developer can hide API services; ● Developer can delete API services; ● Developer can copy API services; ● Developer can create custom API services.	
DRS-32	Developer must have the option to read database schema via API. Developer must have the option to read the list API services available per Database.	Must have

6.5 Coverage Map

The coverage map shows how the **Functional Capabilities** (by an applicant and a registrar) in specific use cases match the functional requirements described

Use Case	User Journey	Functional Capabilities	Technical Requirements of Admin tools	Technical Requirements of User tools

Registration	Postpartum and Infant Care	6. Update Register.
		The local register is updated for later use.
			4.1 User story 1 - Registry schema User Interface;
			4.4 User story 4- Registry schema API;
			DRS-1; DRS-3; DRS-4; DRS-5; DRS-6
				4.3 User story 3 - CRUD data APIs; 6. Service APIs;
				API - Schema;
				API - Data

Registration	Postpartum and Infant Care	9. Update Registry The card number and mothers details are sent to the Govn. Registry for update, e.g. Department of Health or Home Affairs.	DRS-5	4.3 User story 3 - CRUD data APIs; API - Data

Registration	Postpartum and Infant Care	11. Update Register
		The local register is updated for later use and submit the information for the remote Govn. Department Registry.
			DRS-5;
			DRS-13;
			DRS-14
				4.3 User story 3 - CRUD data APIs;
				API - Data

Registration	Postpartum and Infant Care	13. Give Reference Number Once the request is successfully committed the mother is given a reference number	DRS-5	4.3 User story 3 - CRUD data APIs; API - Data

Registration	Postpartum and Infant Care	14. Search Patient Case
		Generate a new folder for case records and link it to the child ID. If no case exists, create one and link the Card, otherwise identify the correct record and store the case UID.
			DRS-5;
			DRS-13;
			DRS-14
				4.3 User story 3 - CRUD data APIs;
				API - Data

Registration	Postpartum and Infant Care	15. Update Registry
		The card number, mothers details, childs details and birth details, are sent to the Govt, e.g. the Department of Home Affairs.
			DRS-5;
			DRS-13;
			DRS-14
				4.3 User story 3 - CRUD data APIs;
				API - Data

Payments	Postpartum and Infant Care	3. Capture details for the Mother
		
		Capture data to verify the mother and prevent fraud per legal requirements, for processing later.
			DRS-5;
			DRS-13;
			DRS-14;
			DRS-28;
			DRS-29;
			DRS-30;
			DRS-31.
				4.3 User story 3 - CRUD data APIs;
				API - Data

Payments	Postpartum and Infant Care	4.1 Validate the mother has completed all steps	DRS-2;
			DRS-5;
			DRS-9
				4.3 User story 3 - CRUD data APIs;
				API - Data

Payments	Postpartum and Infant Care	4.2 Verify mother has no pending incentive voucher for this milestone?	DRS-2;
			DRS-5;
				4.3 User story 3 - CRUD data APIs;

				API - Data

Payments	Postpartum and Infant Care	6. For the mother, a cash payment is given via a paper payment voucher	DRS-2;
			DRS-5;
				4.3 User story 3 - CRUD data APIs; API - Data

Payments	Postpartum and Infant Care	9. Record payment status and amounts in registry	DRS-3 DRS-5;	4.3 User story 3 - CRUD data APIs; API - Data
Case Management	Postpartum and Infant Care	3. HC workers have access to minimal required data for the purposes of completing this process.	DRS-5;	4.3 User story 3 - CRUD data APIs; API - Data

Case Management	Postpartum and Infant Care	5. The HC worker updates prescriptions for medication	DRS-2;
			DRS-5;
				4.3 User story 3 - CRUD data APIs;
				API - Data

Registration	Unconditiona l Social Cash Transfer	3. The admin may create a new registration record if none exists for for the potential beneficiary	DRS-2;
			DRS-5;
				4.3 User story 3 - CRUD data APIs;
				API - Data

Registration	Unconditiona l Social Cash Transfer	7. Optional: Programme specific data is often entered into a separate Beneficiary Registry associated with a Beneficiary Operations Management System (BOMS)*	DRS-1
			DRS-2;
			DRS-3;
			DRS-4;
			DRS-5;
				4.3 User story 3 - CRUD data APIs;
				API - Data

7 Data Structures

7.1 Standards/Protocols

The following standards are applicable to data structures in the registration building block:

1. All dates should follow ISO 8601.
2. RFC 7159 - The JavaScript Object Notation (JSON)
3. Open -API Version 3.1.0

7.2 Resource Model

The resource model shows the relationship between data objects that are used by this Building Block.

Resource Model:

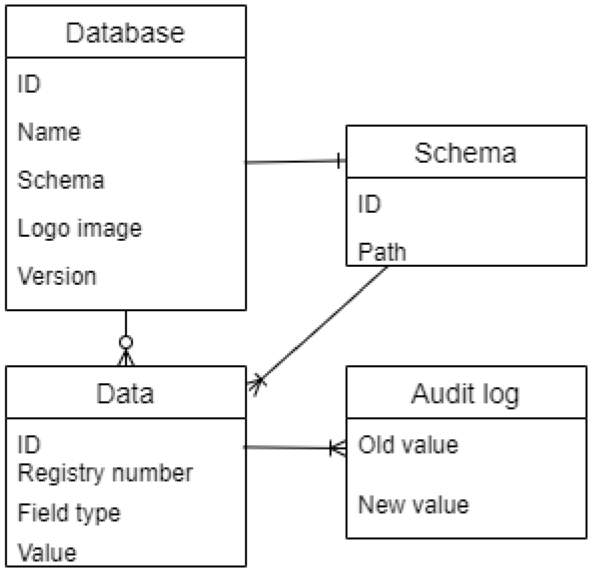


Illustration 3- Resource model. See editable image here.

7.3 Data Elements

The Data Elements provide detail for the Resource Model defined above. This section will list the core/required fields for each resource. Note that the Data Elements can be extended for a particular use case, but they must always contain, at the minimum, the fields defined here.

Minimum Required Data:

Name	Description	Type	Required
Database ID	Unique identifier of a database	integer	yes
Database name	Name that will define the database content. Name is public.	varchar	yes
Database schema	Database schema. See example in Appendix 1, Appendix 2.	json	yes
Database logo	Visual image for the database	bytea	no
Version	Database version. Each change in schema will produce the next version of the database and API services.	numeric	yes
Catalogue name	Database name in the list	varchar	yes
Data ID	Data element unique identifier	integer	yes
Registry number	Additional registry identifier. Unique identifier in the registry.	varchar	yes
Field type	Field type: datetime, date, boolean, text, number, file	varchar	yes
Field value	Field value, data stored in the field.	datetime, date, boolean, text, number	yes
Audit log old value	Field value before change	datetime, date, boolean, text, number	yes
Audit log new value	Field value after the change	datetime, date, boolean, text, number	yes

8 Service APIs

This section describes external APIs that must be implemented by the building block. Additional APIs may be implemented by the building block (all APIs must adhere to the standards and protocols defined), but the listed APIs define a minimal set that must be provided by any implementation.

Registries BB may contain multiple registries/databases. The dynamic nature of the database structure requires a standard set automatically generated APIs for all databases hosted on the platform. The system generates default API method endpoints automatically after each publish of the database schema. A new API service version is generated after each schema publish. Database schema version and API versions are in sync.

The naming convention and a structure of the API endpoint is the following:

/{information type}/{registry acronym or code}/{version}/{API method as a name}.

Example 1: `/api/data/cr/1.0/create`

Example 2: `/api/v1/database/modify`

Each registry contains a unique set of data and the BB enables an Analyst to change the data storage structure/schema on the fly. In this following example API descriptions are generated for one example dataset for the Postpartum Infant Care Program registry, where Caretaker and infant child is registered and registration ID is issued.

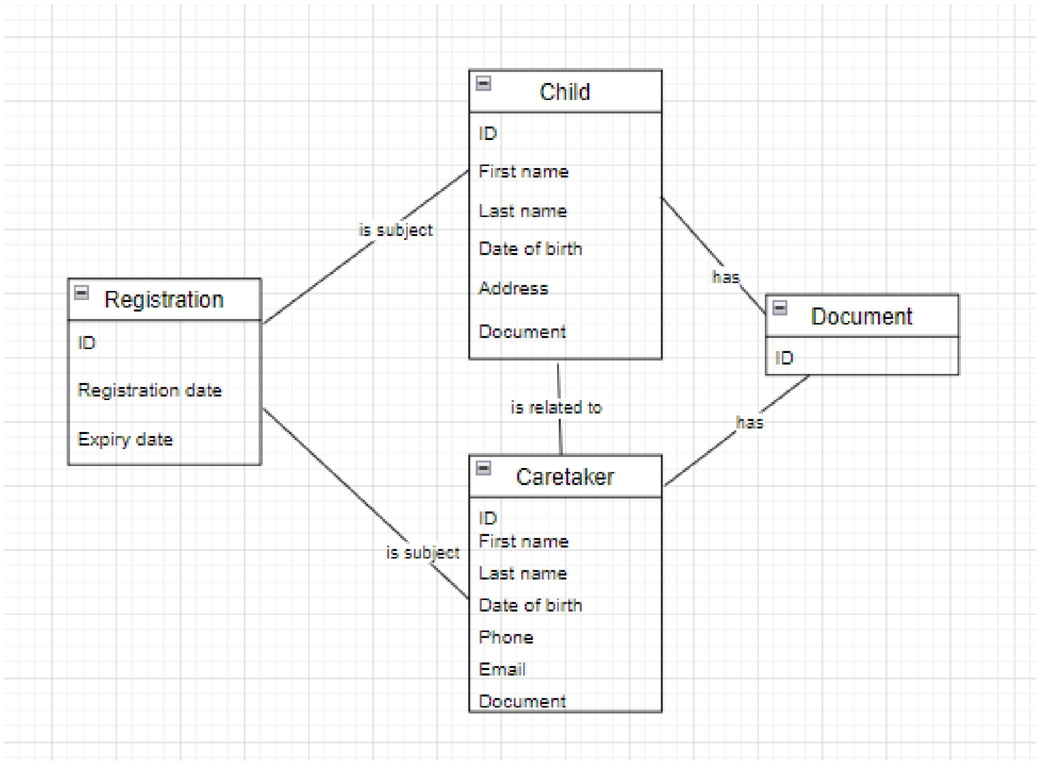


Illustration 4- Example registry database logical data model. Editable version is here.

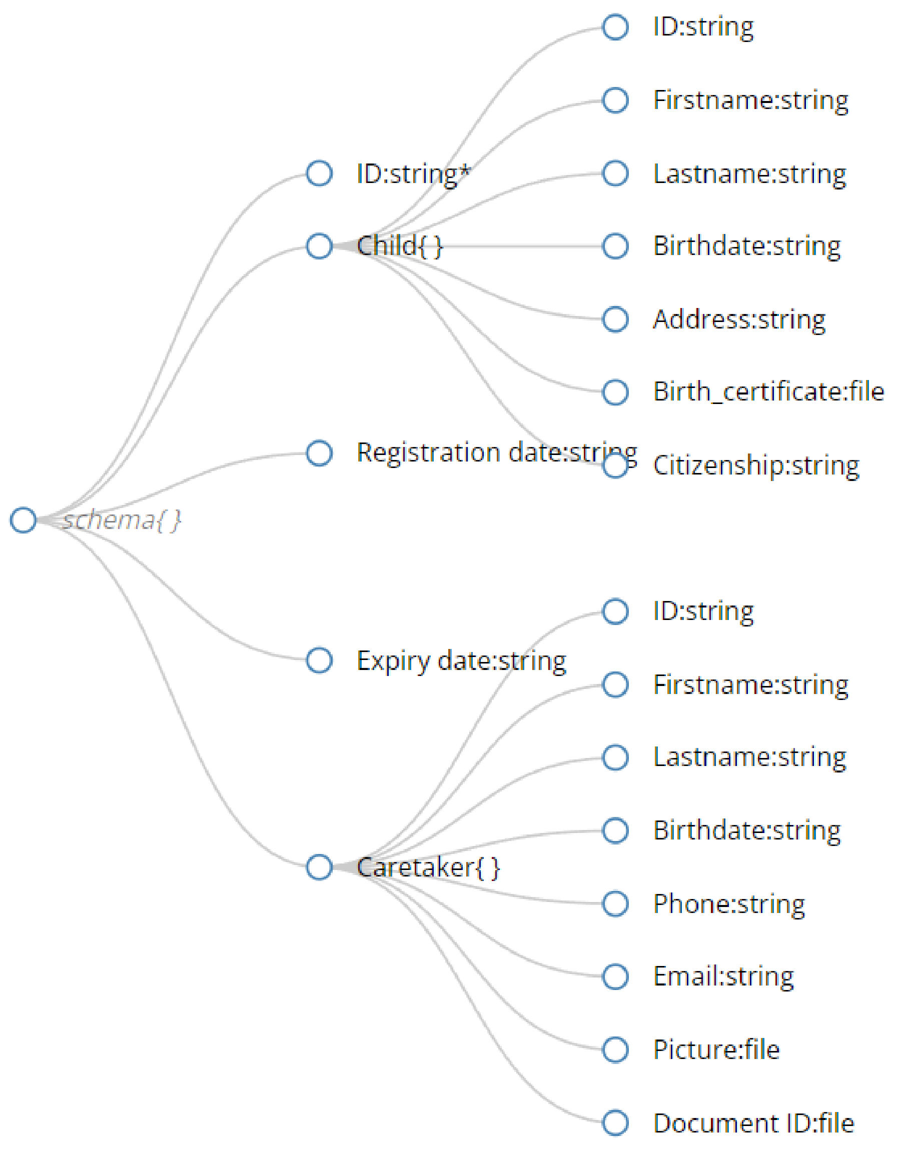


Illustration 5- Example registry database Json schema.

Digital registries BB is expected to host the following API services for each database hosted on the platform.

8.1 DATA CREATE

Description: Creates a new record in the registry database.

Request endpoint: POST /data/{code}/{version}/create

Example API: see in github.

8.2 DATA UPDATE

Description: Updates one existing record in the registry database.

Request endpoint: PUT /data/{code}/{version}/update

Example API: see in github.

8.3 DATA UPDATE-OR-CREATE

Description: API updates existing record if matching with input parameters is successful. If record is not found the API will create a new record.

Request endpoint: POST /data/{code}/{version}/update-or-create

Example API: see in github.

8.4 DATA UPDATE-ENTRIES

Description: Updates multiple records in the registry database that match the input query.

Request endpoint: PUT /data/{code}/{version}/update-entries

Example API: see in github.

8.5 DATA LIST (Search)

Description: Searches (Regex supported) and returns multiple records as an array-list.

Request endpoint: GET /data/{code}/{version}

Example API: see in github.

8.6 DATA READ

Description: Searches and returns one record.

Request endpoint: POST /data/{code}/{version}/read

Example API: see in github.

8.7 DATA READ-VALUE

Description: Searches and returns one record's one field value.

Request endpoint: GET /data/{code}/{version}/{ID}/read-value/{field}.{ext}

Example API: see in github.

8.8 DATA EXISTS

Description: Searches records based on input parameters and returns boolean answer (true/false).

Request endpoint: POST /data/{code}/{version}/exists

Example API: see in github.

8.9 DATA DELETE

Description: Delete record.

Request endpoint: DELETE /data/{code}/{version}/{ID}/delete

Example API: see in github.

8.10 DATA My personal data usage

Description: The purpose of this API is to make personal data protection better and make BB personal data usage transparent by showing who has looked at personal data of the user. Each user can see who has looked at their personal data and when. The definition of personal data is described by each BB owner in the respective country.

Request endpoint: GET /data/MyPersonalDataUsage/{version}

Example API: see in github.

Database Schema APIs are following:

8.11 DATABASE SCHEMA READ

Description: API reads existing registry database schema.

Request endpoint: GET /api/V1/database/{id}

Example API: see in github.

8.12 DATABASE SCHEMA MODIFY

Description: API creates a new registry database schema or updates existing schema if matching with input parameters is successful. If schema is not found the API will create a new schema.

Request endpoint: POST /api/V1/database/modify

Example API: see in github.

8.13 DATABASE SCHEMA DELETE

Description: API deletes registry database.

Request endpoint: DELETE /api/v1/database/{id}

Example API: see in github.

8.14 DATABASES LIST

Description: API gets all databases and schema versions as a list array.

Request endpoint: GET /api/v1/databases

Example API: see in github.

Standards/Protocols:

The API is built using representational state transfer (REST) software architectural style (https://restfulapi.net/) and described in Open API 3 standard (https://swagger.io/specification/) using YAML (a human-readable data-serialization language - http://yaml.org/). Request and response body is in JSON (lightweight data-interchange format - https://www.json.org/json-en.html).

9 Workflows

This BB does not have internal workflows.

10 Other Resources

- API descriptions in Github
  https://github.com/ingmarvali/BuildingBlockAPI/blob/main/RegistrationBB/GovStack_Registration_BB_API_template-1.1.0-resolved.json

- Security requirements:
  https://docs.egovstack.net/v1.1.0/Security_Requirements_v1.1.0.pdf

- Architecture requirements:
  https://docs.egovstack.net/v1.1.0/Architecture_and_NonFunctional_Requirements_v1.1.0.pdf

- Information Mediator requirements:
  https://docs.egovstack.net/v1.1.0/Information_Mediator_Building_Block_Specification_v1.1.0.pdf

- Logical process blueprint:
  https://docs.google.com/document/d/1DRjpuyINjf6YVBRrEh9Q6VdB0zVzq1aqGQOukpktWZ8/edit#heading=h.z0zf4zjfif4c

- Infrastructure capabilities
  This depends on the amount of users, but optimal estimated hardware requirements are:
  - 8 Core CPU
  - 32GB RAM
  - 500GB SSD/HDD
  - 1000Mbit Network connectivity with dedicated publicly routable IPv4 address

System should be able to run either on VirtualMachine or on a dedicated server.

11 Key Decision Log

- The UNCTAD's Generic Database Builder (eRegistrations) system will be used as a reference system in describing the functional requirements.

- 23.09.2021 - WG meeting, based on review recommendations made by Architecture WG, we decided to add API services or IT-specialists to create/modify/delete registry database schema.

- 22.11.2021 - Coverage map chapter will be added to the document.

- 28.02.2022 - Review comments incorporated to the main document (v1.1.0) 24.02.2022 (see below). Future consideration chapter updated based on reviewers comments. Recommendations not to be considered in this building block documented (see below)

Comments/Feedback	**Suggested Action/Reason**
1. Incorporated in V1	
2. Key Digital Functionalities. I would add here capability to manage access to the Registry data	Yes, see requirement DRS-6 Modifications to be absorbed in this version Sharing data with others is a function that was added to the requirements. See DRS-33
2. Key Digital Functionalities. currently, my impression is that the Registration BB is optimised for entering and processing data and not for retrieval and	Registration BB functionality is described in another document. See here . Pull (read) data functionality in Registration BB improved. Modifications to be absorbed in this version:

Comments/Feedback	Suggested Action/Reason
1. Incorporated in V1	
usage. I see it as a missing capability requirement.	1. add arrows to the drawing to reflect the two-way communication. See data retrieval API Open API services descriptions for more information.
DRS-2. Foreign keys. I am not sure, I can understand the way how FK-s suppose to work (see comment below in Appendix 2)	Databases in this concept are stored as tables, thus the foreign key works the same as in a structured database. In the Digital Registries user interface it must be possible to open another database by clicking on the ID of one database and all corresponding records from the other database will open. In API, the developers can decide how to use the Foreign key to improve the UX. Modifications to be absorbed in this version: We improved the functionality description.
DRS-3 1. does it includes Dropbox option to select from List of Values (LOV)? How can I define List values and how can I attach it to field in a form? 2. Does it includes option for hierarchy of List of Values, where selected value in one LOV defines subset of allowable options in another LOV?	1. yes, catalog/select values are used, however this is managed by client UI (Registration BB). Digital registries is storing data/ key of the list element. Catalogs are managed in Registration BB/ other BB. 2. Sub-catalogues function and the control could be added to Registration BB when required by the Use Case. Digital Registries BB contains Enum list validation option. Modifications acceptable but to be taken up in future version
6. I was talking about the REST endpoint URL. The URL has a placeholder for version (marked as {version}). I believe it to be the version of the API being called, but the document throughout talks about DB schema version, and no mention of this version being the version of the API. That could create confusions on what the parameter "version" is.	Thank you for the clarification. Will be implemented in this version. The system generates default API method endpoints automatically after each publish of the database schema. A new API service version is generated after each schema publish. Database schema version and API versions are in sync. I hope this clarifies the confusion.

3. Not to be considered	
General recommendation. Harmonize all URLs across all building blocks. The slide deck and the specs documents don't correspond very well and at times contradict. For example, the Purpose of the Registries BB isn't articulated in the Specs doc but appears only in the slide deck.	Not implemented. Purpose is described in the specification and in the powerpoint. It is unclear what was expected.
2. Registries MUST adhere to core principles of data being Live, Reusable and Trustworthy.	Registry management principles would be another document - user manual, or best practices to build digital registries. Some examples: UNCITRAL Legislative Guide on Key Principles of a Business Registry | United Nations iLibrary
Registry SHALL support use of multiple data stores like RDBMs, NoSQL and Graph to represent the data representation across domains.	We are not restricting the support of any mentioned database types.
Registry MAY optionally support to do payload level encryption (especially for highly sensitive data) over transport layer to ensure backend services (beyond TLS termination) handle data securely.	This is solved by Information Mediator BB.
Section 7 states, this BB doesn't have internal workflows. But this BB MUST support meta data about workflows on who has attested the data.	Yes, metadata is like any data that can be added to the Digital Registries Database. Analyst can decide how to store the processing information.
Since APIs are available to access the digital registries, there is no need to have a separate data access mechanism for the administrators. It is important to use the same APIs in the user interface proposed for the administrators.	UI is needed for analysts to improve the UX. Same API will be used to create databases.
1. Description. I understood, that authors divided overall registration domain into two parts: Registration BB covers process of submitting	This BB is about creating and managing data. IAM BB for user rights and roles management is a separate BB. However the system has its own internal authorization system.

3. Not to be considered	
applications and processing the data to make decision and here suppose to be a Registry usage part. However, I do not see here specifics about usage of registries. For example, usage of Registry may have complex access rules, requirements of fast access to large data sets etc. I do not see any discussion here on those matters	Internal ABAC has been specified in DRS-6. Open API for data exchange has been described below. According to Use Case, no large datasets available for storing. Modifications not feasible.
1. Description. Should there not be an intro what "registry" is and why in some cultures there are different concepts for registries and databases? I would also think that it would be appropriate to guide a potential reader to right choice of register, i.e more policy choice questions for the intro part	Registry is defined in the Glossary. Domain specific rules of registration are different, so there is no common principles how to build or manage a generic registry. This could be generated in the future. Registry owner- Analyst has all the rights to decide the data set to be stored. Basic registry functionality has been added to the technology and described as requirements. The analyst has the opportunity to decide by him/herself what data to store in the Digital Registries BB. The principles of traditional Registries management is not in the scope of this document. See example domain specific principles here: UNCITRAL Legislative Guide on Key Principles of a Business Registry | United Nations iLibrary
1. Description -Digital Registries is simple to use. This is true only if specific functional domain is properly embedded into the no-code platform as kinda of DSL. However, in case if I need to design complex models by myself, then usage of no-code platforms may be additional burden and bring too cumbersome development experience, which ultimately will decrease sustainability of the solution and increase TCO and even making kinda vendor-lock-in to the platform	With current task in hand, the use case is simple enough to be nicely fit into No-Code digital registry. In the future when we will find a domain and data set that needs something more complex, then we can analyze how to solve it. It is always possible to use traditional methods to build registry databases if Digital registries is not suitable. The training how to build a domain specific registry is the future challenge of next organizations. A marketplace would help to solve these challenges. This spec enables to build any domain registry.

3. Not to be considered	
2. Key Digital Functionalities this capability should include also metadata aka configuration schema of a Registry in order to configure schema in dev environment, test it in next environment and then deploy it to production. This capability should be shown here explicitly. Also automation is needed for that.	Yes , this is point nr 1 and 2. See more requirement DRS-10 for schema import and export. Modifications not feasible.
3. (7) Import/export data from/to external files; This capability should include also metadata aka configuration schema of a Registry in order to configure schema in dev environment, test it in next environment and then deploy it to production. This capability should be shown here explicitly. Also automation is needed for that.	Yes , this is point nr 1 and 2. See more requirement DRS-10 for schema import and export. Modifications not feasible.
DRS-3 Does it includes option of pre-filling of forms based on Applicant context in the current registry as well as in other 3rd partied sources?	Pre-filling of forms is Registration BB functionality. User Interface is managed in Registration BB. Digital registries has Triggers to prefill data fields in the database (ID, prefix etc.) Modifications not feasible
in case of notaries there is a need for more complex schema of user rights as far as Notar is independent private sector entity	In this case Notary/ health worker is using Registries BB via Information Mediator. System must have basic User right management. Additional Roles may be added after IAM system is in place.
in complex organisations with implies requirement to have an organisational data here and to able to create permissions for positions in organisational units. Also, there is aspect of substitution in case of illness, vacations etc.	Yes, we must link the system with IAM BB where we will get those roles.
DRS-13 I would suggest to have similar abstraction for legal entity as well. Also, for relations between persons, legal entities, addresses etc. Otherwise it will be too complicated for analysts to model domains.	What would be the use case? the Digital registries allows to store any data. Modifications not feasible
to create meaningful UI to use complex data, for example, from licensing domain using	Digital registries has capabilities to facilitate registry from any domain. If there is such a

3. Not to be considered	
such generic schema will be just too complicated and cumbersome task. It will not fly globally.	data schema that Digital Registres BB id not capable to facilitate, then traditional structured databases can be used instead. Digital Registries functions well with the Mother and Child Program. It will take 15 minutes to create and publish a database for the program without writing any line of code.
9. General comment. I do not think, such generic level as it is now the specification will be helpful for practitioners let's say in Rwanda, Botswana, Ukraine, Moldova etc. It does not bring simplification. It is more like you have to learn some proprietary cumbersome language to make simple things in a very complicated manner.	Digital Registries capabilities are following no-code principles, thus no need to learn cumbersome languages. However any IT system (including MS SQL and Oracle, My SQL) requires admin/analyst user who must learn the principles of maintaining and configuring the system. In this case we have envisioned this role to be Analyst. Additional specifications to this BB can be added in the next iteration. Currently it is MVP and focused on one domain. Next , when new domains will be added, then we can add additional flesh on the bones. The system has capabilities for admins and analyst to build databases, sometimes even temporary registries and this is the best tool for it.
9.5 Should maybe integrity of registration process itself merit special concern depending on the registry in question and the obligation of owner to guarantee the correctness of data, e.g. think of property database and potential abuse?	This question requires additional debate. According to the methodology the focus is in the Use Case of Mother and Child Program and related processes. Let's not focus on all things at once. Next iteration we can add processes and use cases. So far the Digital Registries BB fulfilled the needs of the Domain process.
normally, database may contain several containers/tables, which may have also its own key. Like in case of RDBMS foreign key would have reference to database, table and record. Here is only 2 key. Would it be sufficient?	Noted. Currently it is sufficient.
DRS-15 Does not clarify how transit is secured. (Encryption of a payload can be added as a "May")	Information transit between building blocks is governed by Information Mediator BB. All information transit is encrypted between the BB-s. See more in Information Mediator. Decision: Added a comment on transit to this requirement.

12 Future Consideration

12.1 Integration with a blockchain solution to guarantee the integrity of the data and logs. The function would notice unauthorized changes in data. This option may be available with a fee therefore should be optional.

12.2 Implementation improvements- system installation could be done with more simplifications so it would be a full SAAS cloud solution.

12.3 Data validation and evaluation tools could be developed into the core Registries BB logic so it would generate warnings if data quality issues appear.

12.4 Event based automated data export from Registries BB to legacy databases. Connection with messaging and workflow BB is required.

12.5 Plug-in no-code connector to existing registry databases. It's currently a challenge to connect to existing registry databases. As a solution, the Digital Registries BB could offer a plug-in tool to connect existing databases to Information Mediator without the need to develop custom connectors.

12.6 Open data component. It should be possible to mark down the data that must be visible as open data (API, bulk download).

12.7 Personal data usage in Digital Registries in synchronization with Consent Management BB capabilities.

12.8 Enable to connect Digital Registries BB to Verifiable Credential networks (W3C VC).

12.9 Analyze a way to enable analyst to decide which data must be encrypted while in rest. The goal is to secure data while in rest.

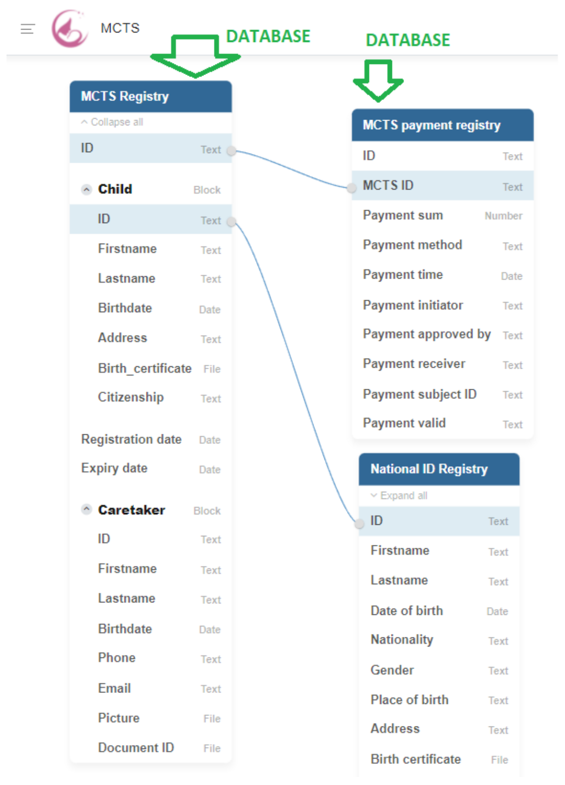
12.10 Data MUST be protected by proper anonymisation with analytics and related reporting functions. Proper analysis and user requirements mapping must be done based on a real use case.

12.11 Maintenance functionalities and roles to help everyday operations of registries.

12.12 Review results to be added to the next versions of the specifications. See decisions in the following table.

Comments/Feedback	**Suggested Action/Reason**
DRS-3 Add fields to DB schema. Does that means that collection types are not supported? For example, I want Applicant to submit her employments records history and I will define fields for one employment and then I want to define a history object, which is collection of employment records sorted by time. Then I want to add some validation rules not to a field but to a collection. Can I do that? How can I do that?	In most cases, the validation is done upon capturing data (Registration BB). In Digital Registries basic validation rules for single record are in place (required, unique). When the use case requires to add more validation functionality (e.g. to collection types), it will be added in the next iteration.
Associated meta data against (for) these actions (Enrol, Authenticate, Consent, Attest, Claim, Discover and Update actions) MUST be part of the registry to bring in the authenticity and non-repudiability of registry data.	The UI and API has all necessary functions. If new functions are needed then these will be added during develop time. 11.1 Integration with a **blockchain** solution to guarantee the **integrity of the data** and logs. The function would notice unauthorized changes in data. This option may be available with a fee therefore should be optional.
Registries MUST support verifiable credentials (W3C VC) services using the trusted registry data as an integrated service with in this BB. This VC SHALL work both in Online and Offline modes.	11.8 Enable to connect Digital Registries BB to **Verifiable Credential** networks (W3C VC).
Data MUST be secured during capture, transmission and rest with right encryption along with integrity protection.	Data is secured when in transit by Information Mediator (BB). The communication uses encryption in all endpoints. **Data in rest** will be taken as a next challenge in V2. Integrity protection is planned in V2 as well. 11.9 Analyse a way to enable analyst to decide which data must be encrypted while in rest. The goal is to secure data while in rest.
Data MUST be protected for privacy with appropriate roles and access permission for downstream consumption. This SHALL be using other BBs.	Yes, in Key Digital Functionalities chapter we describe it like this: 12. Manage access to registry data. Authorize users to see and edit registry records or data field (ABAC based access management). DRS-6- Authorization to create and manage databases, API usage and access to DATA.

Comments/Feedback	Suggested Action/Reason
	Additional requirements will be added based on real Use Cases.
Data MUST be protected by proper anonymisation with analytics and related reporting functions.	11.10 Data MUST be protected by proper anonymisation with analytics and related reporting functions. Proper analysis and user requirements mapping must be done based on a real use case.
The scope is limited to simple registries. In real life, we are likely to have registries that require a multi level structure. In RDBMS terms, the data for a registry may need to be stored in multiple tables. It is not easy to handle such entities by defining foreign keys among tables. There should be a clear mechanism identified to handle digital registries that have a multi-level structure. While it may appear easy to create a new API end point for every version, this places undue burden on the calling systems. Whenever a new API version is launched, the calling system will have to be modified to refer to the new API end point. It is also quite difficult to invoke different API end points from the same front-end. Instead of this, the API end point should remain the same and the payload should indicate the version.	The requirements are based on a USE CASE with minimal viable product methodology. Therefore the complex database structure was not needed, thus the requirements focused on simple case. However the requirements to build more complex data storing structures can be added when use cases require it. Digital Registries enables to create multi level registries. For example if we rename a database to a table, and tables can be linked with foreign keys, then we have a multi level registry. In Digital Registries description we decided to name the tables as databases because this improves the user experience. See example illustration below:

Comments/Feedback	Suggested Action/Reason
	 MCTS DATABASE DATABASE MCTS Registry Collapse all ID Text Child Block ID Text Firstname Text Lastname Text Birthdate Date Address Text Birth_certificate File Citizenship Text Registration date Date Expiry date Date Caretaker Block ID Text Firstname Text Lastname Text Birthdate Date Phone Text Email Text Picture File Document ID File MCTS payment registry ID Text MCTS ID Text Payment sum Number Payment method Text Payment time Date Payment initiator Text Payment approved by Text Payment receiver Text Payment subject ID Text Payment valid Text National ID Registry Expand all ID Text Firstname Text Lastname Text Date of birth Date Nationality Text Gender Text Place of birth Text Address Text Birth certificate File The issue with API endpoint versions may need some clarification in the specification. Currently we have written (DRS-4): “Publish uses versioning. Every publish creates a new version of the database schema; Old database schemas must be available to the users; Data stored in the old database versions must be usable in old versions and in new versions; “ This description should be clear enough to explain that the users can still work with old API versions and do not have to switch to new API versions right away. However there is a risk that large schema changes may influence the old APi versions and therefore the system has a limit. We will analyze the risks and schema version options in the V2.
2.2 Event based notifications. I see that as an issue. Subject of the Registry record may have specific life-cycle and on event of life-cycle there is a need to do some things, which is the reason why Registry even exists. For example, when	Yes, this function will be added in the iteration 2. Use cases did not require such functionality right away.

Comments/Feedback	Suggested Action/Reason
driver license is expiring and state cannot notify person about that - I would say it is dysfunctional registry of driver licenses.	11.4 Event based automated data export from Registries BB to legacy databases. Connection with messaging and workflow BB is required.
DRS-3 you should add also time range	Modifications acceptable but to be taken up in future version
DRS-3 what kind of language should analyst to learn in order to express all that? One may provide, for example, embedded JavaScript editor. Will it be OK for being compliant with the spec?	We have no-code policy. Therefore, all functions must be available via user interface (and API). Advanced regex, JavaScript options could be added, but this is the decision in the implementation phase by the implementers. Modifications acceptable but to be taken up in future version
DRS-4 before publishing I would recommend to have option for testing	Noted. Will be added to the future functionality.
3. Registries MUST support Enrol, Authenticate, Consent, Attest, Claim, Discover and Update actions using Registry and/or other BBs. Discover and update SHALL naturally fit into Registry BB.	Enroll, Authenticate is solved by Information mediator and Security BB. Additional IAM system will be available for authorization. Consent is in the scope of Consent management BB. Integration will be added in V2.
4.4 there are couple of important technical requirements missing with regard to API. For example, see Payments BB section 6.1.1 API management Gateway. I suggest to have generic API requirements description and here just refer to that.	To be analyzed in V2. API Management Gateway Handles all the API messaging calls and API access control verification from other BBs to the Payment BB and vice versa as well as within the Payment BB. All requests from other BBs first go through the API gateway. The gateway then routes requests to the appropriate application/service. The API Management gateway will: ● Use Identity and access management for authentication ● Perform input validation checks to prevent oversized message attacks, SQL injection attacks as well as JSON and XML threats, ● Require authentication for all API users; ● Manage access quotas and throttling; ● Logging of all API calls made

Comments/Feedback	Suggested Action/Reason
	• Allow API providers to limit the rate of consumption for all API users. • Transform backend error messages into standardized messages so that all error messages look similar; this also eliminates exposing the backend code structure.
The authors try to treat analyst and administrator as one user category and applicant as another user category. It could be a better idea to have three distinct roles viz., analyst, administrator and applicant. The analyst should be responsible for preparing the design and he/she should control the database design. The analyst should have no role to play in data administration. That responsibility should be with the administrator. Otherwise, we may be combining the responsibilities of architects and operators.	Use cases currently do not require administrators. These role descriptions can be added later during implementation phase. 11.11 Maintenance functionalities and roles to help everyday operations of registries.
6.3 Conflicting - Earlier in the doc, it mentions views for personal data, but no information on how personal data should be accessed Generic Comment - No ability to get user consent while accessing personal data.	According to our vision, personal data is like any other data that can be accessed by API or via user interface. Respective authorization must be granted in order to CRUD Personal data. The principles of user consent is governed by Consent Management BB. Modifications acceptable but to be taken up in future version. Decisions: I added a line in the Future scope for the Consent Management. 11.7 Personal data usage in Digital Registries in synchronization with Consent Management BB capabilities.

Appendix 1- Database Read-Schema Response Example

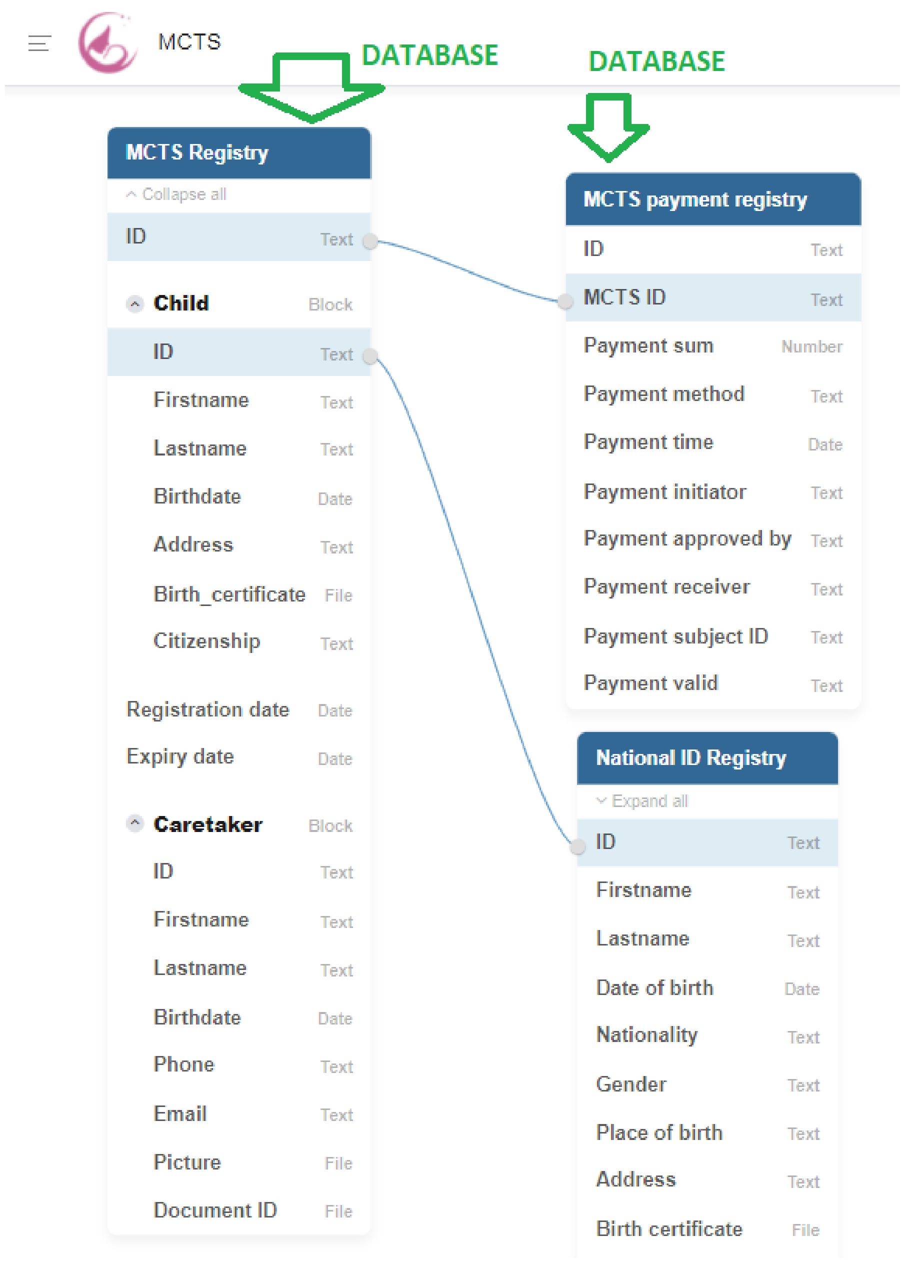
```
{
  "id": 353,
  "version": "2.7",
  "name": null,
  "description": null,
  "institution": null,
  "number_format": "{code}{indexNoByCode}",
  "schema": {
    "type": "object",
    "properties": {
      "ID": {
        "type": "string",
        "triggers": [
          {
            "conditions": [
              {
                "logic": "==",
                "value": "",
                "gate": "&&"
              }
            ],
            "actions": [
              {
                "type": "set-value",
                "value": "MCTS{indexNoByCode}",
                "field_id": 1
              },
              {
                "type": "upper-case",
                "field_id": 1
              }
            ]
      }
    ],
    "primaryKey": true,
    "readOnly": true,
    "description": "Registration ID",
    "example": "MCTS31",
    "$id": 1
  },
  "Child": {
    "type": "object",
    "properties": {
      "ID": {
        "type": "string",
        "description": "Child ID",
        "example": "ID2",
        "$id": 13
      },
      "Firstname": {
        "type": "string",
        "description": "Child first name",
        "example": "Usha",
        "$id": 3
      },
      "Lastname": {
        "type": "string",
        "description": "Child last name",
        "example": "Bajaj",
        "$id": 4
      },
      "Birthdate": {
        "type": "string",
        "format": "date",
        "description": "Child data of birth",
        "example": "2021-10-03T07:03:36Z",
        "$id": 5
      },
  "Address": {
    "type": "string",
    "description": "Child's address",
    "example": "Longroad 123, Welltown, Ethiopia",
    "$id": 7
  },
  "Birth_certificate": {
    "type": "file",
    "consumes": [
      "application/pdf",
      "image/jpeg",
      "image/png",
      "image/gif",
      "image/tiff",
      "image/bmp",
      "image/x-ms-bmp",
      "application/rtf",
      "text/rtf",
      "application/vnd.openxmlformats-officedocument.wordprocessingml.document",
      "application/vnd.oasis.opendocument.text"
    ],
    "antivirus": true,
    "description": "Child's birth certificate data. ISO8601/UTC",
    "example": "2021-10-03T07:03:36Z",
    "$id": 6
  },
  "Citizenship": {
    "type": "string",
    "description": "Child is a citizen of this country. ISO 3166-1 encoding list",
    "example": "ET",
    "$id": 20
  }
 },
 "description": "Child object data that is queried from database",
 "$id": 2
},
"Registration date": {
  "type": "string",
  "format": "date",
  "description": "Record registration date",
  "example": "2021-10-03T07:03:36Z",
  "$id": 14
},
"Expiry date": {
  "type": "string",
  "format": "date",
  "description": "Record expiry date",
  "example": "2021-10-03T07:03:36Z",
  "$id": 15
},
"Caretaker": {
  "type": "object",
  "properties": {
    "ID": {
      "type": "string",
      "description": "Caretaker's ID",
      "example": "ID1",
      "$id": 12
    },
    "Firstname": {
      "type": "string",
      "description": "Caretaker's first name",
      "example": "Sowmya",
      "$id": 9
    },
    "Lastname": {
      "type": "string",
      "description": "Caretaker's last name",
      "example": "Bajaj",
      "$id": 10
    },
    "Birthdate": {
```

```
  "type": "string",
  "format": "date",
  "description": "Caretaker's birth date",
  "example": "2021-10-03T07:03:36Z",
  "$id": 16
},
"Phone": {
  "type": "string",
  "description": "Caretaker's phone number",
  "example": "+3725278511",
  "$id": 11
},
"Email": {
  "type": "string",
  "description": "Caretaker's email",
  "example": "test@test.et",
  "$id": 17
},
"Picture": {
  "type": "file",
  "consumes": [
    "application/pdf",
    "image/jpeg",
    "image/png",
    "image/gif",
    "image/tiff",
    "image/bmp",
    "image/x-ms-bmp",
    "application/rtf",
    "text/rtf",
    "application/vnd.openxmlformats-officedocument.wordprocessingml.document",
    "application/vnd.oasis.opendocument.text"
  ],
  "antivirus": true,
  "description": "Caretaker's picture.",
  "$id": 18
```

```
        },
        "Document ID": {
          "type": "file",
          "consumes": [
            "application/pdf",
            "image/jpeg",
            "image/png",
            "image/gif",
            "image/tiff",
            "image/bmp",
            "image/x-ms-bmp",
            "application/rtf",
            "text/rtf",
            "application/vnd.openxmlformats-officedocument.wordprocessingml.document",
            "application/vnd.oasis.opendocument.text"
          ],
          "antivirus": true,
          "description": "Caretaker's document",
          "$id": 19
        }
      },
      "description": "Caretaker's information",
      "$id": 8
    }
  },
  "$incrementIndex": 20,
  "required": [
    "ID"
  ]
},
"schema_tags": [
  {
    "name": "",
    "path": "/Child/Citizenship",
    "is_fulltext": true
  },
  {
    "name": "",
    "path": "/Child/Lastname",
    "is_fulltext": true
  },
  {
    "name": "",
    "path": "/Child/Firstname",
    "is_fulltext": true
  },
  {
    "name": "",
    "path": "/ID",
    "is_fulltext": true
  }
],
"schema_flags": [
  {
    "name": "mandatory",
    "path": "/ID"
  },
  {
    "name": "unique",
    "path": "/ID"
  }
],
"fields_uniques": [
  []
],
"is_draft": false,
"is_disabled": false,
"is_archived": false,
"modified_at": "2021-10-03T08:35:01.775915Z",
"by_user_name": "ingmar.dev",
"by_user_auth_id": 1,
"by_on_behalf_of_user_auth_id": null,
"by_on_behalf_of_user_name": null,
"generic_services": [
{
"service_id": 1,
"name": "data-create",
"is_visible": true,
"used_count": 0
},
{
"service_id": 2,
"name": "data-read",
"is_visible": true,
"used_count": 0
},
{
"service_id": 9,
"name": "data-read-value",
"is_visible": true,
"used_count": 0
},
{
"service_id": 3,
"name": "data-list",
"is_visible": true,
"used_count": 0
},
{
"service_id": 4,
"name": "data-update",
"is_visible": true,
"used_count": 0
},
{
"service_id": 6,
"name": "data-delete",
"is_visible": true,
    "used_count": 0
  },
  {
    "service_id": 7,
    "name": "data-exists",
    "is_visible": true,
    "used_count": 0
  },
  {
    "service_id": 10,
    "name": "data-update-or-create",
    "is_visible": true,
    "used_count": 0
  },
  {
    "service_id": 11,
    "name": "data-update-entries",
    "is_visible": true,
    "used_count": 0
  },
  {
    "service_id": 12,
    "name": "data-create-entries",
    "is_visible": true,
    "used_count": 0
  },
  {
    "service_id": 13,
    "name": "data-update-or-create-entries",
    "is_visible": true,
    "used_count": 0
  }
],
"data_index_increment": 0,
"has_logo": false
}
```

Appendix 2- Example Schema with Foreign Keys

```
{
    "id": 185,
    "version": "1.0",
    "name": null,
    "description": null,
    "institution": null,
    "number_format": "{code}{indexNoByCode}",
    "schema": {
        "type": "object",
        "properties": {
            "ID": {
                "type": "string",
                "triggers": [{
                    "conditions": [{
                        "logic": "==",
                        "value": "",
                        "gate": "&&"
                    }],
                    "actions": [{
                            "type": "set-value",
                            "value": "{code}{indexNoByCode}",
                            "field_id": 1
                        },
                        {
                            "type": "upper-case",
                            "field_id": 1
                        }
                    ]
                }],
                "primaryKey": true,
                "readOnly": true,
        "$id": 1
      },
      "Client ID": {
        "type": "string",
        "$id": 5
      },
      "Client first name": {
        "type": "string",
        "$id": 6
      },
      "Client last name": {
        "type": "string",
        "$id": 7
      },
      "Client date of birth": {
        "type": "string",
        "format": "date",
        "$id": 8
      },
      "Prescriptions": {
        "type": "string",
        "foreignKeys": [{
          "databaseKey": "54",
          "values": [{
            "fieldKey": "6"
          }]
        }],
        "$id": 3
      }
    },
    "$incrementIndex": 13,
    "required": [
      "ID"
    ]
  }
```



MCTS Registry

ID	Text
Child	Block
ID	Text
Firstname	Text
Lastname	Text
Birthdate	Date
Address	Text
Birth_certificate	File
Citizenship	Text
Registration date	Date
Expiry date	Date
Caretaker	Block
ID	Text
Firstname	Text
Lastname	Text
Birthdate	Date
Phone	Text
Email	Text
Picture	File
Document ID	File

MCTS payment registry

ID	Text
MCTS ID	Text
Payment sum	Number
Payment method	Text
Payment time	Date
Payment initiator	Text
Payment approved by	Text
Payment receiver	Text
Payment subject ID	Text
Payment valid	Text

National ID Registry

ID	Text
Firstname	Text
Lastname	Text
Date of birth	Date
Nationality	Text
Gender	Text
Place of birth	Text
Address	Text
Birth certificate	File

Illustration 6. - Linking databases with foreign keys.