Enterprise Data Asset Management (EDAM)

EDAM Community Edition v0

2022-12-17 – Draft v0

Table of Contents

[1. Motivation 1](#_Toc121924013)

[2. Communicable Diseases Assets Management 2](#_Toc121924014)

[3. EDAM Features 2](#_Toc121924015)

[4. EDAM UI Screens and Forms 4](#_Toc121924016)

[4.1 EDAM App Main Screen 4](#_Toc121924017)

[4.2 EDAM Asset Dictionary 5](#_Toc121924018)

[4.3 EDAM Asset Data Output Options 6](#_Toc121924019)

[4.4 EDAM Asset Tree Structure 8](#_Toc121924020)

[4.5 EDAM Use Case – Mappings 9](#_Toc121924021)

[4.6 EDAM Use Case – Book – Booklets 10](#_Toc121924022)

[4.6.1 EDAM Use Case Mappings – Map Play (JSONata) 11](#_Toc121924023)

# 1. Motivation

After many years of struggling with spreadsheets, word documents and other forms of documentation while trying to work with Business Analysts (BA) and Architects and collaborative define an “Enterprise Data Model” it was clear to us that we should strive for a better way to capture and maintain Data Assets information.

There are excellent commercial tools to help is this area still we have not found any tool that offers a substantial set of features to do data assets management that we could rely on without some kind of financial burden to our customers and projects. Not every customer or project has the budget to manage the expenses of a workable “Data Asset Management” (DAM) and for this reason BA’s and Architects resolve to use spreadsheets, word documents or other forms of documentation to manage solutions data assets.

After many years of development without any budget and limited time only the hope that we could find a way to provide a tool with some level of usable DAM features to help in the documentation and generation of Data Assets artifacts Datovy is presenting its “Open Source” EDAM solution. With the possible contributions of the larger “Open Source” community we hope to find a better way to make this effort a suitable alternative to Commercial software specially for those projects with limited budget.

EDAM offers output that is always verifiable with schema definitions and its output to an Excel Workbook a guaranteed consistent with declared entities that is a substantial improvement over handmade Spreadsheets that can be easily break and extremely difficult to maintain.

This December 2022 EDAM is released in GITHUB with a set of limited functionality but enough to be able to quickly generate documentation, schemas, and other artifacts. The rest of this document will try to provide a glimpse into EDAM and existing features. This is not aimed to compete with Commercial offerings and is work in progress therefore expect incomplete functionality and documentation, by-hand configurations or work, bugs to be fixed and other hurdles that will be found while trying to use this product. We have been users of our own software and we think that those hurdles worth the trouble since we could ease the Business Analysis and Data Architecture by quickly generate useable documentation and artifacts. Hopefully we could find some additional contributors that could spend time with us and improve this product.

# 2. Communicable Diseases Assets Management

At the same time of this release Datovy is also providing another contribution to the “Open Source” community with the release of a database for “Communicable Diseases” or “Disease Surveillance” (DS) with limited and partial support for related CDC messaging.

The DS database is used in the first release of EDAM as a sample non-trivial collection of data entities and components that show case supported features.

# 3. EDAM Features

The list of supported features include:

|  |  |  |
| --- | --- | --- |
| **Feature** | **Description** | **See** |
| *EDAM Projects Support* | | |
| EDAM Projects | A project is composed of an expected minimum folder structure that should include an “Arguments” folder that define details about the Asset to be managed. | 4.1 |
| *Schema Reading Support* | | |
| XSD to EDAM | Convert a complex multi-namespace XSD into EDAM assets definition set. |  |
| JSON Schema to EDAM | Convert a complex multi-namespace JSON schema into EDAM assets definition set. |  |
| DDL Definitions to EDAM | Convert MS-SQL, MySql, or Oracle schema definitions into EDAM assets definition set. Support for multiple schemas and related namespace data elements collection sets. |  |
| EDI to EDAM | Convert an enhance EDI definition into EDAM assets definition set (partial support). |  |
| *Schema Writing Support* | | |
| EDAM to XSD | Output to XSD retaining all namespace information. |  |
| EDAM to JSON Schema | Output to most recent JSON Schema Draft. |  |
| EDAM to DDL | For the time being only MS-SQL, others later including multiple namespaces to schemas support. |  |
| EDAM to JSON-LD | Output to JSON-LD Components and definitions. |  |
| *Use Case Definitions and Mappings (on the works)* | | |
| EDAM to Use Case | Identify and document elements from an Asset that are part of a use case. | 4.5 |
| EDAM to Use Case Mappings | Identify and document elements involved in transformations and mappings. | 4.6 |
| *Other Supported Features* | | |
| EDAM to Data Dictionary | Excel generated file given an EDAM assets definition set. |  |

Samples for most of the above had been produced for the “Disease Surveillance” database (Visit the related “Datovy Communicable Disease” Open-Source community project).

# 4. EDAM UI Screens and Forms

This section contains a few screenshots of the EDAM UI Windows Application (App).

## 4.1 EDAM App Main Screen

The Main Screen display the list of projects (in the left-hand side) and 3 Tabs (Domains, File, and Asset) on the right-hand side (see Figure 4.1). The App projects path is configurable and can be set in the “appsettings.json” by setting the “AssetConsolePath” to the desired folder path.

Graphical user interface, text, application

Description automatically generated

Figure 4.1 Projects and Arguments files.

The App Visual Studio project has a default project configuration and files for the “Disease Surveillance” database that serves as an example and can be used to test existing functionality.

Within the “File” TAB 2 buttons are found, the first to “Save” updated to the selected Arguments file in display, and the second to execute the process to read the defined source data and generate the EDAM data dictionary (see 4.2 for more details).

## 4.2 EDAM Asset Dictionary

After the EDAM dictionary is generated, the results are displayed in the “Asset” TAB. The number of elements is displayed, and the “NAMESPACES” drop down list contains all found namespaces withing the source data (if any).

Graphical user interface, application, table

Description automatically generated

Figure 4.2. Asset Dictionary

The “ASSET” panel has 3 buttons and a dropdown box. The first button is related to the screen as shown in 4.2, the second will display the Asset data components in a Tree structure (see 4.4).

## 4.3 EDAM Asset Data Output Options

Within the “ASSET” panel (on the left) save options are provided (see Figure 4.3).

Graphical user interface, application, table

Description automatically generated

Figure 4.3. EDAM Asset Data Output Options.

The output options include:

|  |  |  |
| --- | --- | --- |
| **Option** | **Description** | **Status** |
| XSD | XML Schema | Available |
| JSON-Schema | Will Output as to latest Draft Specification. | Available |
| GRAPH-SQL (GQL) | TigerGraph GQL. | Testing |
| Database | Output to MS-SQL EDAM database. The connection string should be specify in the “appsettings.json” file. | Available |
| DDL | MS-SQL Schema definition. | Available |
| Excel | Data Dictionary as an Excel Workbook. | Available |
| Data Template File | Data Template file definition. | Testing |

The output is sent to the Projects “Document” folder and for the “Disease Surveillance” EDAM includes:

Graphical user interface, application

Description automatically generated

Figure 4.3.1. EDAM Output folders with generated documentation or schemas.

Once the “Database” output is selected the definitions are stored in the DataElement table (see 4.3.2).

Graphical user interface, application, table, Excel

Description automatically generated

Figure 4.3.2. EDAM Database showing the DataElement table with loaded definitions.

## 4.4 EDAM Asset Tree Structure

The EDAM Asset data set can be displayed in a Tree Structure format by choosing the “folder” icon to the right of the “ASSET” label (see Figure 4.4).

Graphical user interface, application, table

Description automatically generated

Figure 4.4. Asset as a Tree Structure.

Once, in this case the “Disease Surveillance” DDL database definition is read by the application the schemas and tables details are use to create a hierarchy representation of the data as shown in Figure 4.4 whose root element is “Disease\_Surveillance\_Document”. The “\_Document” is automatically appended to the root element.

## 4.5 EDAM Use Case – Mappings

While selecting the “ASSET” mappings option the “Use Cases” TAB is shown. Already defined Use Cases will be listed and can be selected to continue working on those.

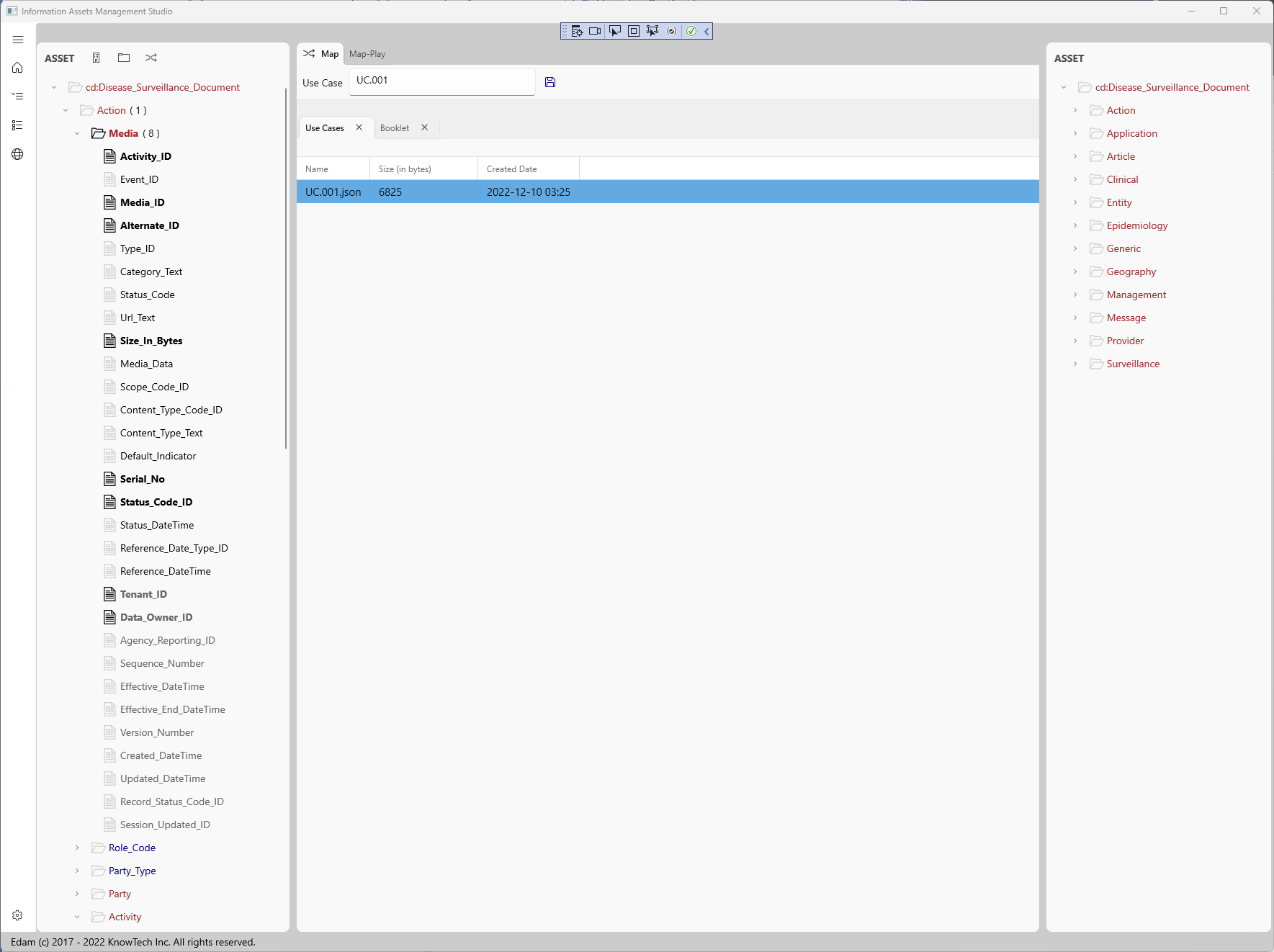


Figure 4.5. EDAM Use Case TAB and source (left) – target (right) Asset Trees.

While selecting items from the Source or Target the Map panels display them and Booklets definitions can be defined (see 4.6).

## 4.6 EDAM Use Case – Book – Booklets

As an example, the selection of a data-element on the source will display the mapping panel showing its path withing the document and any related Booklet associated with this reference data-element (see Figure 4.6).

Graphical user interface, text, application, email

Description automatically generated

Figure 4.6. Use Case – Book / Booklet.

Similar to Jupiter Notebooks the inner Use Case mapping Book can have a booklet per each mapped item allowing the definition of one or more Text or Code cells. In this example the Code cell uses “JSONata” as the language that the code will support. By clicking in the execute button the output will show beneath the code box.

### 4.6.1 EDAM Use Case Mappings – Map Play (JSONata)

The “Map Play” TAB offers the opportunity to execute code based on selected Language, for this version only JSONata is supported. Given an instance of a document for the source (left side) Tree a JSON sample is automatically generated by clicking the “Sample” label and display in the panel as shown in Figure 4.6.1.

Graphical user interface, text

Description automatically generated

Figure 4.6.1. Use Case Mappings Play using JSONata.

For these examples some code can be tested by placing it on the “Request” panel and executing it by pressing the “Execute” button. The output of the request is shown in the “Results” panel.