Automatic production of publications

Chapter 1 specifies the data model to represent publications and documents regarding information on governmental decision-making. The next step in achieving uniform publications is to identify how the data model can be validated against defined constraints. Also, the creation of a user-friendly tool to automatically turn input data into a validated JSON document is something that will be covered in this chapter. Before being able to validate the publications, constrains must be introduced.

Constraints

Using the same attributes on a national, or even bigger scale is the first step in improving the coordination and cooperability of governmental bodies. However, uniform agreements about the attribute's values have to be in place to prevent ambiguity. Three types of restrictions on the values can help in achieving this.

To avoid ambiguity, some attributes should only contain values in accordance with predefined standards. Dates can be represented in many ways. ISO 8601 is a unambiguous calendar format that is internationally understood (International Organization for Standardization, 2019). All dates should be conforming to this standard, representing dates in the format 'YYYY-MM-DD'. Similarly, ISO 639-3 (International Organization for Standardization, 2007) provided three-letter language code elements. Language values should follow this standard, writing languages in the form 'eng' or 'nld'. The mime type attribute already indicates the use of mime types, or media types as specified in IETF RFC 6838 (Internet Engineering Task Force, 2013). As most documents are PDF documents, mostly the mime type 'application/pdf' will be used. Each attribute should also be restricted to one or more valid JSON data types. These consist of strings, numbers, objects, arrays, booleans and null values. The validation of the use of the correct standards and data types will be covered in the next chapter on the automatic production of JSON documents.

To further constrain attribute values, some of the attributes should be a constrained choice. For publications, the type of publication should be a choice of two possible values, 'request' or 'active disclosure'. Both publications and documents have the 'valuation' attribute. This is a constrained choice of 'public', 'partially public', 'not public' or 'already public'. The document type is a choice of either 'request', 'decision', 'inventory list' or 'released document'.

To pressure the creators of the metadata, a few attributes are marked as required. These are attributes that should always be present and are always attributes that already are present in publicly available publications, but hard to extract for a computer. For publications, the following attributes are required: identifier, title, type, handledBy, decisionDate and valuation. If the publication is based on a request, the fileDate and adjourned attributes are also required. For documents, the required attributes are: identifier, title, date, fileName, fileExtension, mimeType and documentType. If the document is a released document, also annexType and valuation are required. If not completely public, the groundsOfRefusal are required and when the valuation is 'already public', the alreadyPublicLocation shall be provided.

Table 1. Value constraints on the defined attributes

Object – attribute	**Constraint**
Publication – fileDate	ISO 8601 date format
Publication – decisionDate	ISO 8601 date format
Publication – type	Constrained choice (‘request’, ‘active disclosure’)
Publication – valuation	Constrained choice (‘public’, ‘partially public’, ‘not public’, ‘already public’)
Document - date	ISO 8601 date format
Document - downloadDate	ISO 8601 date format
Document – language	ISO 639-3 language format
Document – mimeType	IETF RFC 6838 media type format
Document – documentType	Constrained choice (‘request’, ‘decision’, ‘inventory list’, ‘released document’)
Document - valuation	Constrained choice (‘public’, ‘partially public’, ‘not public’, ‘already public’)

Validation

The validation of publication JSON documents can be done by using JSON Schema (Droettboom, 2022). It is a powerful tool to validate JSON data. It enables users to annotate and validate JSON documents. This makes the schema a specification of the data with the ability to run a validator against it. The schema itself is stored in a JSON document and understands type constraints, such as the fact that ‘numberDocuments’ should be an integer. It also allows value constraints, such as a constrained choice for a ‘documentType’. Finally, regular expressions can lay constraints on values. This will be used to ensure the right date format. With the ‘required’ keyword, a list of required attributes can be provided. JSON Schema supports an if-else structure. This can be used to set additional required attributes if, for example, the type of a document is a ‘released document’. With a valid JSON Schema, a JSON document can be validated in almost every programming language. For validation of the publication data, the Python module ‘jsonschema’ will be used to automatically validate the data based on a pre-defined scheme. ** Small fraction of json schema **

Production

For the production of valid JSON documents containing governmental information, a user-friendly tool has been created to ease working with the in chapter 1 introduced data model. The tool is in the form of a webpage and is built on the Flask web framework. The tool functions in four stages. Firstly, users are able to fill in all earlier identified attributes relating to publications and documents. After filling in details on the whole publication, users can add as many documents as needed. The document can be uploaded as pdf file. If the user selects that the document is a released document, additional attributes are available to provide. Figure (?) and (?) respectively show the interface to add publication and document data. In the second step, the submitted information is turned into a JSON document following the attributes as specified in the first chapter of this paper. Additional data, which are too time-consuming to manually fill in or calculate, on the documents are extracted. These are simple extractable data like the filename, the file’s extension and mime type, but also more technical data like the number of pages, words and characters. These are extracted using a Python module like ‘pdfplumber’, which reads in a pdf document and turns pages into strings of text. After storing all the data in a JSON format, the constraints on the data model are checked against the created JSON Schema with the ‘jsonschema’ Python module. As a fourth step, the validation result is shown to the user. In the case of insufficient data, shortcomings are displayed. An example of this can be found in figure (?) where (?). When the validation succeeds, the option to download the JSON appears. Also, a

CSV file, listing all the documents and their attribute values is available to use as an inventory list. Figure (?) shows an example result page when the provided data are valid.

Example case: Zonneweide Jaagweg

In the light of the replacement of the Wob by the Woo, the province of North-Holland started a pilot on the active disclosure of information on the placement of solar panels along the Jaagweg. The pilot was started to experiment on how to publish actively disclosed documents. A step into the direction of more transparency, however the method of providing the information still remains the same, by just providing the pdf documents without relevant extra information. This does not meet the FAIR principles, as the lack of data impedes the findability and accessibility of the data. Without proper archiving and information management, it will be hard to reuse the data. As ministries, provinces and municipalities struggle to find the best way to publish Wob or Woo data, there is a call to increase interoperability between the governmental bodies. The recent monthly publications on the 'Zonneweide Jaagweg' serve as an example to increase the satisfaction of the FAIR principles by storing the data in the proposed data model and validating the data against the constraints.

The publication in the form of a JSON document is created by providing all information to the production tool introduced in the former paragraph. To increase findability, accessibility and reusability, a theme and theme identifier should be provided to group publications relating to the same project, such as the example of the Zonneweide Jaagweg. Table (?) shows example values of a monthly publication (table or JSON example….). These include all required attributes, next to the topic and topic identifier to indicate a project or theme.

Attribute	**Value**	**Source**
title	Zonneweide Jaagweg (sept, okt, nov, dec 2021)	Publication page
identifier	ZW8	Mock value
type	Actieve openbaarmaking	Derived (Woo pilot)
topic	Zonneweide Jaagweg	Derived (Woo pilot)
idTopic	ZW001	Mock value
decisionDate	2021-12-23	Decision document
handledBy	Provincie Noord-Holland	Derived
valuation	Deels openbaar	Decision document
numberDocuments	7	Publication page
sourceUrl	https://www.noord-holland.nl/Loket/…	Publication page