User-friendly Information

Tool on Urban and Regional Access Regulations Schemes

Contract: MOVE/B4/SER/2019-498/SI2.832125

Deliverable D2.5 - Technical documentation and user documentation of UVAR tool

Consortium:

Contents

Administrative section

Document bios

Document file name	Work package	Tasks
UVARBox_WP2_Deliverable2.5_08-2022_2.0	WP2	T2.2

Version history

Version	Date	Description of changes	Author	Partner
1.0	09.06.2022	Merge user manual, technical documentation and explanations into D2.5	Thomas Piribauer	PRISMA
1.1	22.06.2022	Review	Jon Harrod Booth	Harrod Booth Consulting
1.2	18.07.2022	Adaptations based on review	Thomas Piribauer	PRISMA
1.3	29.08.2022	Content inputs	Andreas Höbausz	AustriaTech
2.0	31.08.2022	Final revision and adaptations	Thomas Piribauer, Sónia Soares	PRISMA, ARMIS

Disclaimer

The views and opinions expressed in this document are the sole responsibility of the author(s) and do not necessarily reflect the views of the European Commission.

About this document

The purpose of this deliverable is to provide a user manual which guides the user of the UVAR Box tool through the different use cases and functionalities of the tool and on the other hand to provide technical documentation for administrators which guides through setting up and configuration of the UVAR Box tool. It is closely related to other deliverables of WP2, especially D2.2 Functional specification of the tool and D2.3 UVAR tool package.

User manual

1 User interface and basic functions

1.1 Login

The UVAR Box tool is developed as a web-application and hence can be used with standard web-browsers and does not need extra installation by the user. Concerning the web-browser, Google Chrome is recommended.

A prerequisite for using the system is a valid login. This serves both to identify authorized users and to determine the role that the logged-in user has in the system. In addition to the functionalities and permissions of a user, administrators have the ability to manage all users of the system.

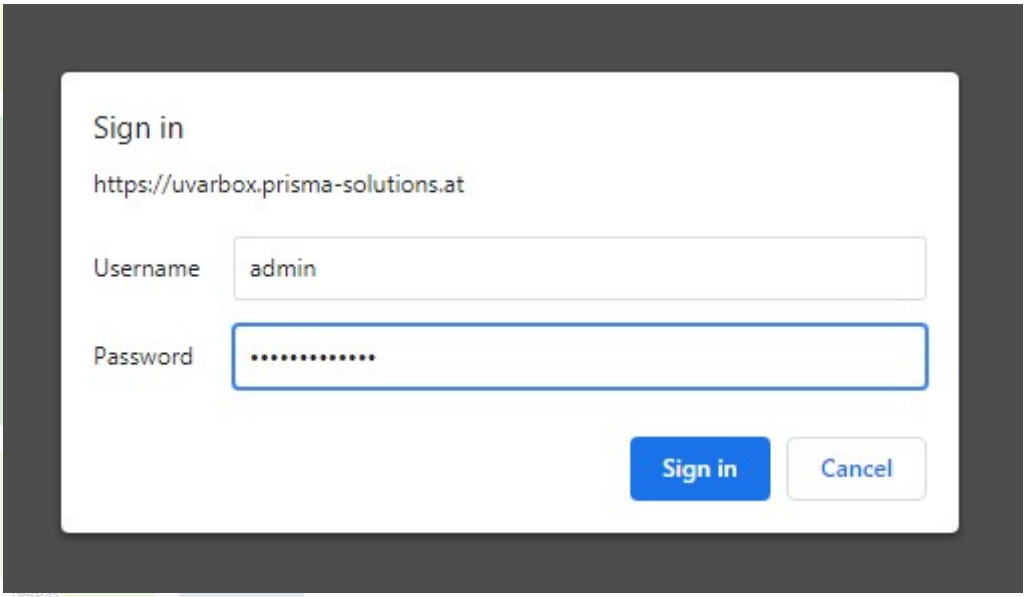

Figure 1: Login

1.2 User interface

After the successful login, the start page is loaded. The following figure shows an overview of the basic elements of the user interface on the start page.

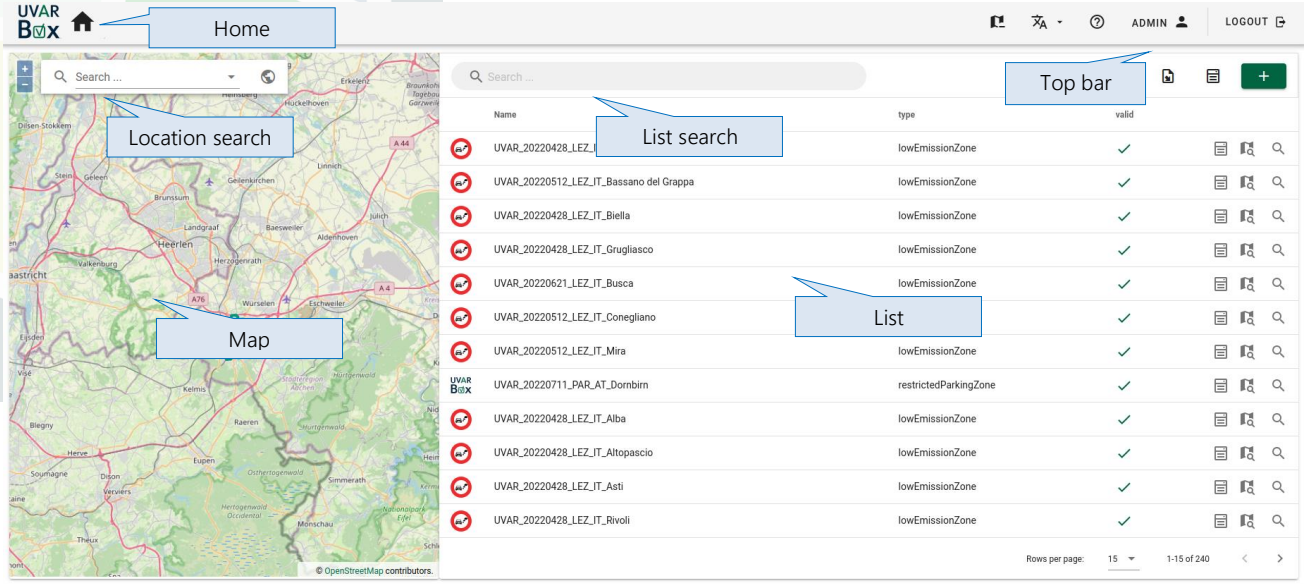

Figure 2: User interface

In the top bar basic information (e.g. user name) is displayed. On the left side a map and on the right side a list is shown.

1.3 Home

Figure 3: Home

- UVAR Box (1): image of UVAR Box Logo
- Home (2): Clicking on the home button reloads the page.

1.4 Language selection, user manual and Logout

The following information and configuration tools are available through the top bar:

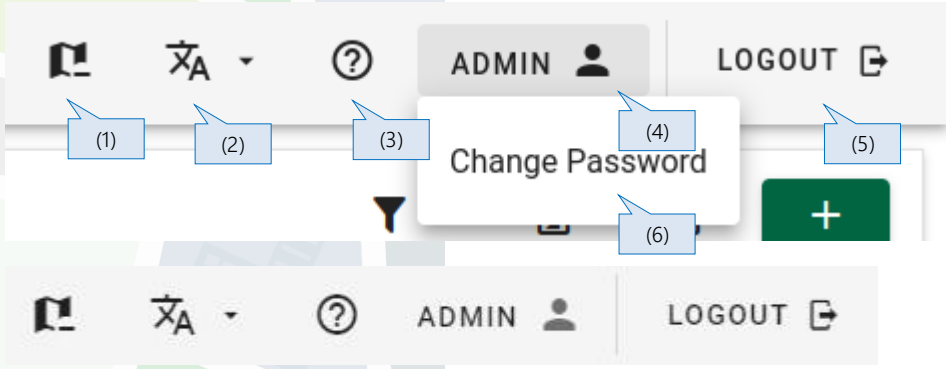

Figure 4: Top bar

- Hide map (1): Clicking on the map button hides/shows the map
- Language (2): To change the language the user can pick one of the languages from the drop-down list. Language selection affects most of the elements of the user interface, e.g. captions of buttons. It does not affect user-specific input like text in the form.
- User manual (3): Clicking on the user manual button opens/downloads the user manual. The user manual covers all use cases of the UVAR Box tool.
- Current user (4): Show the current user.
- Logout (5): To trigger logging out of the application the user can click on Logout and close the browser tab.
- Change Password (6): To change the password the user must click on "Change Password" in drop-down list and a dialog will appear, figure 5.

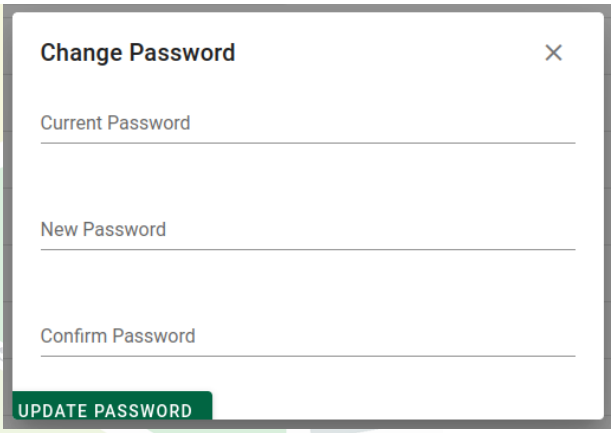

Figure 5: Change password dialog

1.5 List

On the right side of the user interface the list is displayed. The list provides an overview of all UVARs. Depending on the user's permissions and the UVAR properties different icons/buttons are available per UVAR.

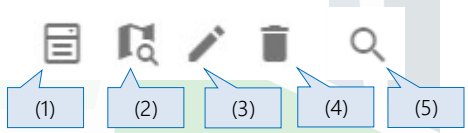

Figure 6: List

- Clone as template (1): Clone UVAR and save as a template, see 4.1 Save UVAR as template for details.
- Zoom to geometry (2): Zoom the map to the geometry of the UVAR.
- Edit UVAR (3): Open the form with UVAR details in edit mode.
- Delete UVAR (4): Delete UVAR.
- Open UVAR (5): Open the form with UVAR details in "read-only" mode, see Figure 7.

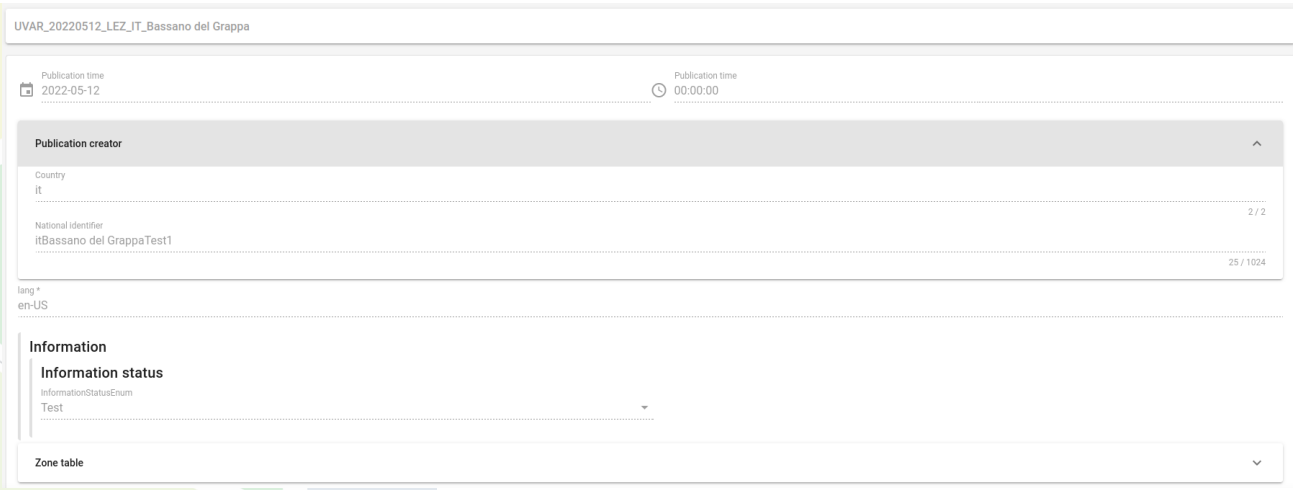

Figure 7: Open UVAR Form

List Search: On top of the list, the list search bar is available.

Figure 8: List search

List search enables the user to search UVARs for specific values. The entered search term filters the list of items to the UVARs matching the search.

Figure 9: List/map filter

List/map filter (6): Open the filter dialog to filter list and map

List and map can be filtered by selecting/deselecting UVAR types

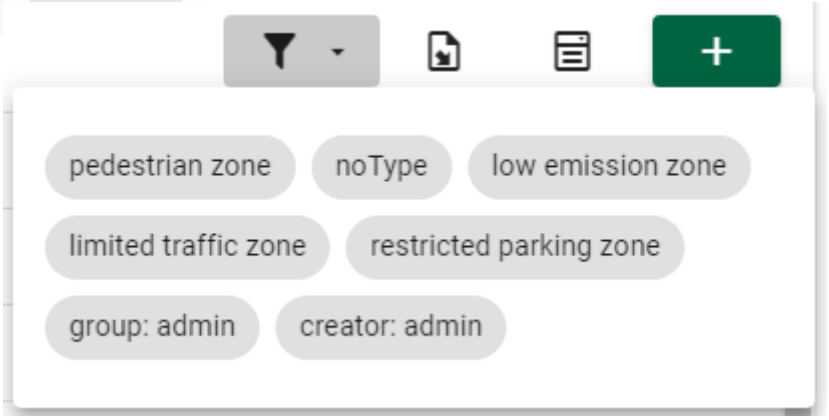

Figure 10: Filter dialog

1.6 Map

On the left side of the user interface the map is displayed. It gives an overview of all UVARs and their locations. It allows to navigate the map using the respective buttons and mouse. In the search box the user can search the map for specific locations.

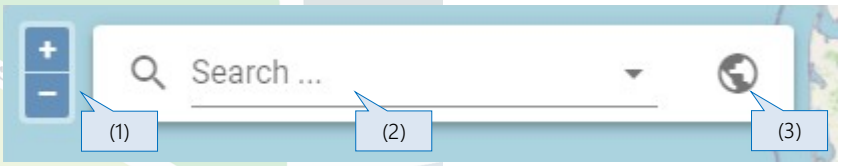

Figure 11: Map

- Zoom in/out (1): navigate map using zoom buttons and mouse.
- Search map (2): search map for specific locations. Map will zoom to the selected result.
- Zoom to default extent (3): Zoom/pan to default extent.

1.7 Form

The UVAR details are shown in the form. The form can be opened by pressing the Edit UVAR or Open UVAR button, see 1.5 List.

Figure 12: Extended editing mode

Extended editing mode (1): switch between simplified and extended editing mode

The form can be rendered in a simplified mode or an extended editing mode, if the user has the privilege to edit the UVAR. The user can switch between these two modes by pressing the pen button (1) at the bottom of the form. The extended editing mode allows to add new elements to the UVAR by pressing the corresponding +buttons. Disabling the extended editing mode gives a view on the UVAR without the functionality to add new elements.

2 UVAR creation

2.1 Import and export

2.1.1 Import of a fully digitized UVAR (DATEX II)

A prerequisite for this use case import is a fully digitized UVAR in the correct data structure according to the DATEX II UVAR profile. The import/export dialog can be triggered by clicking on the import icon above the list:

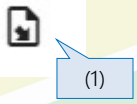

Figure 13: Import icon

- Import/export (1): Open the import and export dialog.

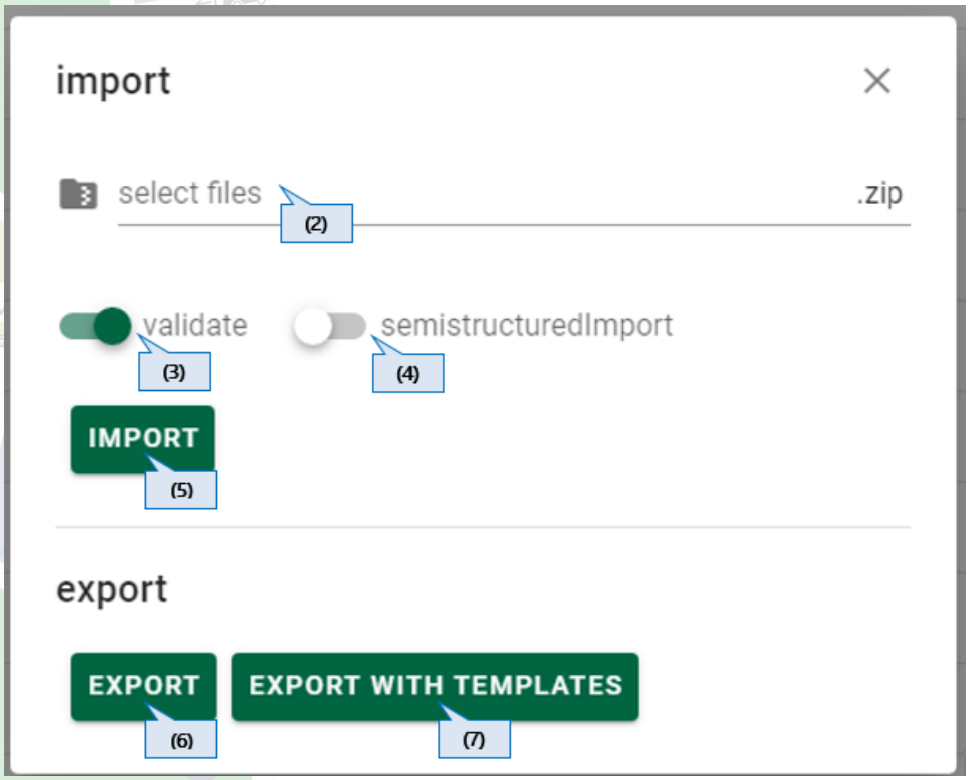

Figure 14: Import/export dialog

- Select file (2): select file for import. Only ZIP-files are allowed, ZIP-file can contain one or more separate UVAR files (XML).
- Validation (3): UVARs should always be validated when importing. All UVARs inside the selected ZIP-file will be rejected if a single validation fails.
- Semi-structured data import (4): trigger to use the import of semi-structured data
- Import (5): start import process. When the import process is finished the user gets feedback if the import was successful. If validation fails, the details can be displayed.
- The import function also serves as an update mechanism. Importing a file with the same XML-file-name updates the corresponding UVAR on import. This can be used for Backup/Restore.

After the successful import the UVAR is stored in the database and the UVAR will appear in the list. The user can now use the zoom to geometry icon to navigate the map to the location of the imported UVAR or open the form of the UVAR by pressing the Edit UVAR or Open UVAR button.

2.1.2 Import of semi-structured data

A prerequisite for this use case is data in a predefined structure and format. Hence, UVAR data containing geometry and attributes (stored in a Shapefile) and an import template (stored as XML) are needed. The import template defines the mapping of the attributes of the Shapefile to the DATEX II UVAR structure. This import template is basically an UVAR template (DATEX II), but contains placeholders to fill in the attributes of the Shapefile. The import can be triggered via the UVAR Box tool or by calling the respective import service call.

After the successful import, the UVAR can be edited in the UVAR Box tool and attributes can be edited as needed.

2.1.3 Export of UVARs

An export of an existing UVAR can be triggered with the following tools.

- Export (5): Generates a ZIP-file containing all UVARs.
- Export with templates (6): Generates a ZIP-file containing all UVARs and templates.

2.2 Creation of a new UVAR

In addition to importing an already existing UVAR, it is possible to create a new UVAR on the tool. For this purpose, a template is necessary and has to be selected after clicking the create UVAR button.

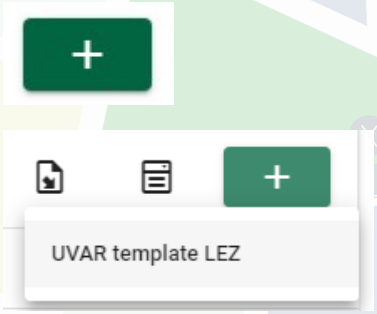

Figure 15: Create new UVAR

A template refers to a specific UVAR type with conditions, attributes and values prefilled. As a result, the form for the definition of the UVAR is generated based on the template. This is used to simplify the process of defining a UVAR because only the city or regional specific characteristics have to be filled in (e.g. issuing authority, geometry). Hence having well defined templates as a basis in the tool eases the process of creating an UVAR.

Remark: As the elements displayed in the form are generated based on the template, deviations to the screenshots and described elements are likely.

As an example UVAR, the basic structure of a Low Emission Zone is shown subsequently. The different elements of a Low Emission Zone are displayed as groups of elements, tabs etc. to give guidance

regarding the nested structure of the data. With + buttons in the form additional elements can be added to the UVAR, if extended editing mode is enabled (see 1.7 Form).

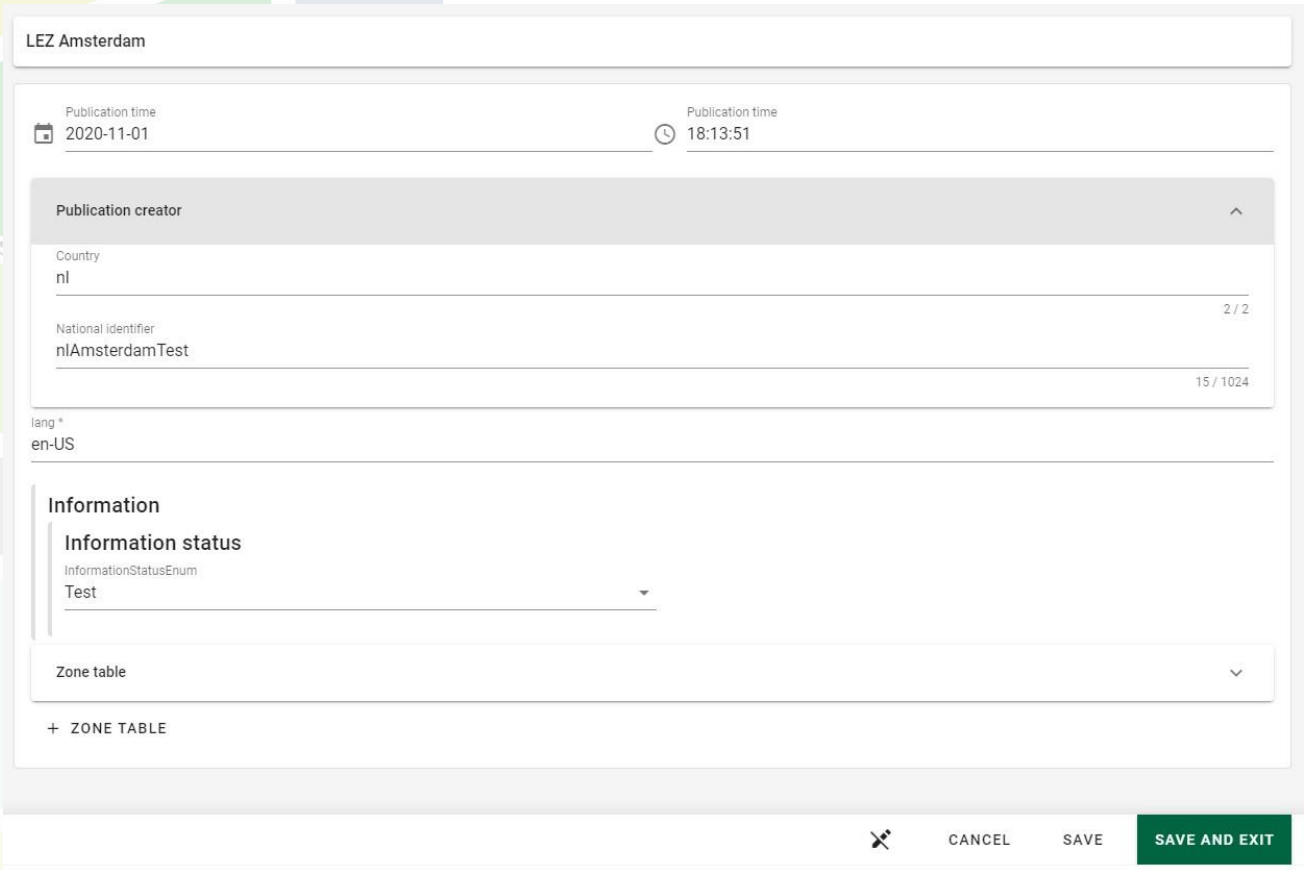

Figure 16: UVAR form

As a user you fill in the basic attributes - like different time attributes or the publication creator. Regarding the structure a UVAR is an entry in a zone table element of the form. The UVAR has some basic attributes like the name, the zone type, status or summary that need to be filled in by the user.

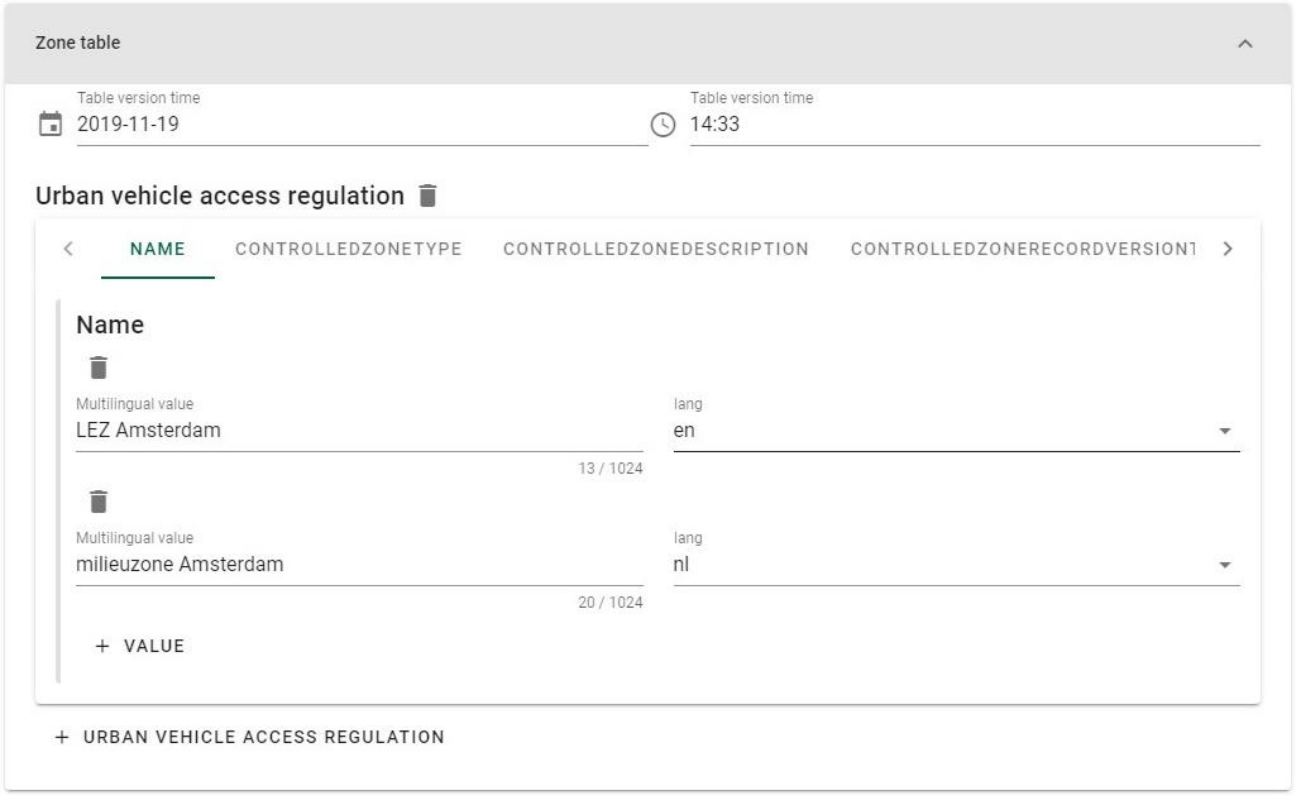

Figure 17: UVAR form - zone table – name of UVAR

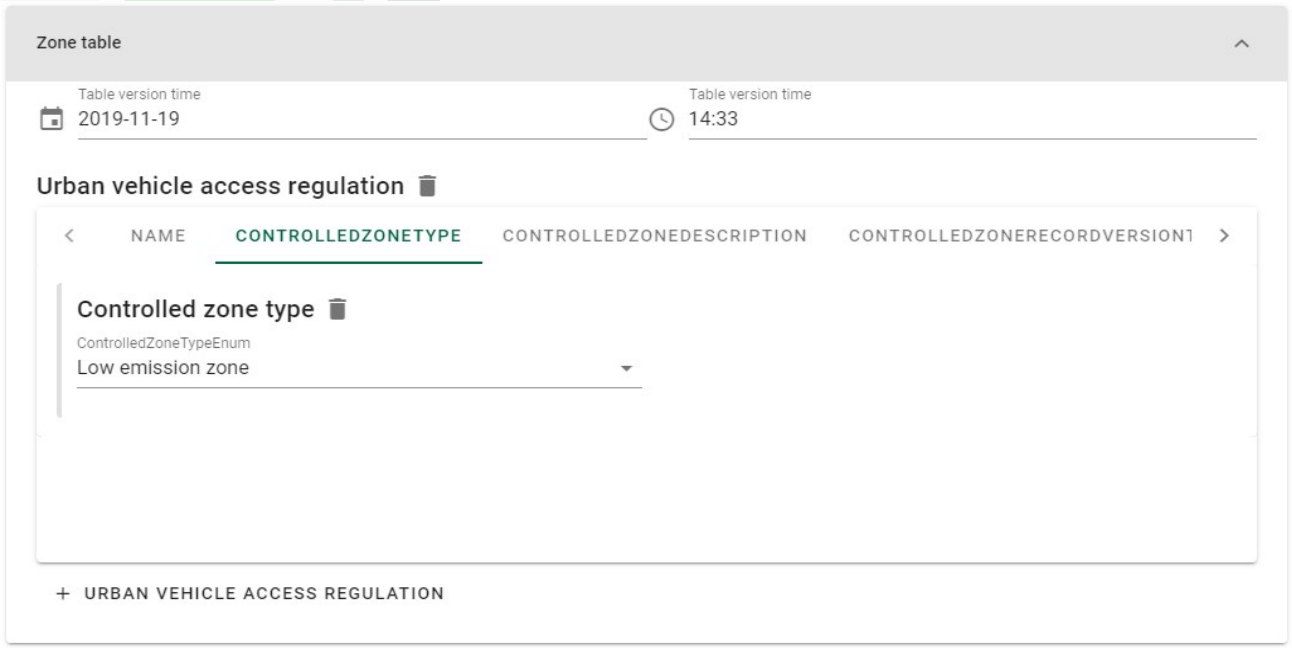

Figure 18: UVAR form - zone table – type of UVAR

The implementation of the UVAR is performed by issuing one or more traffic regulation orders, which have basic attributes like issuing authority or regulation id.

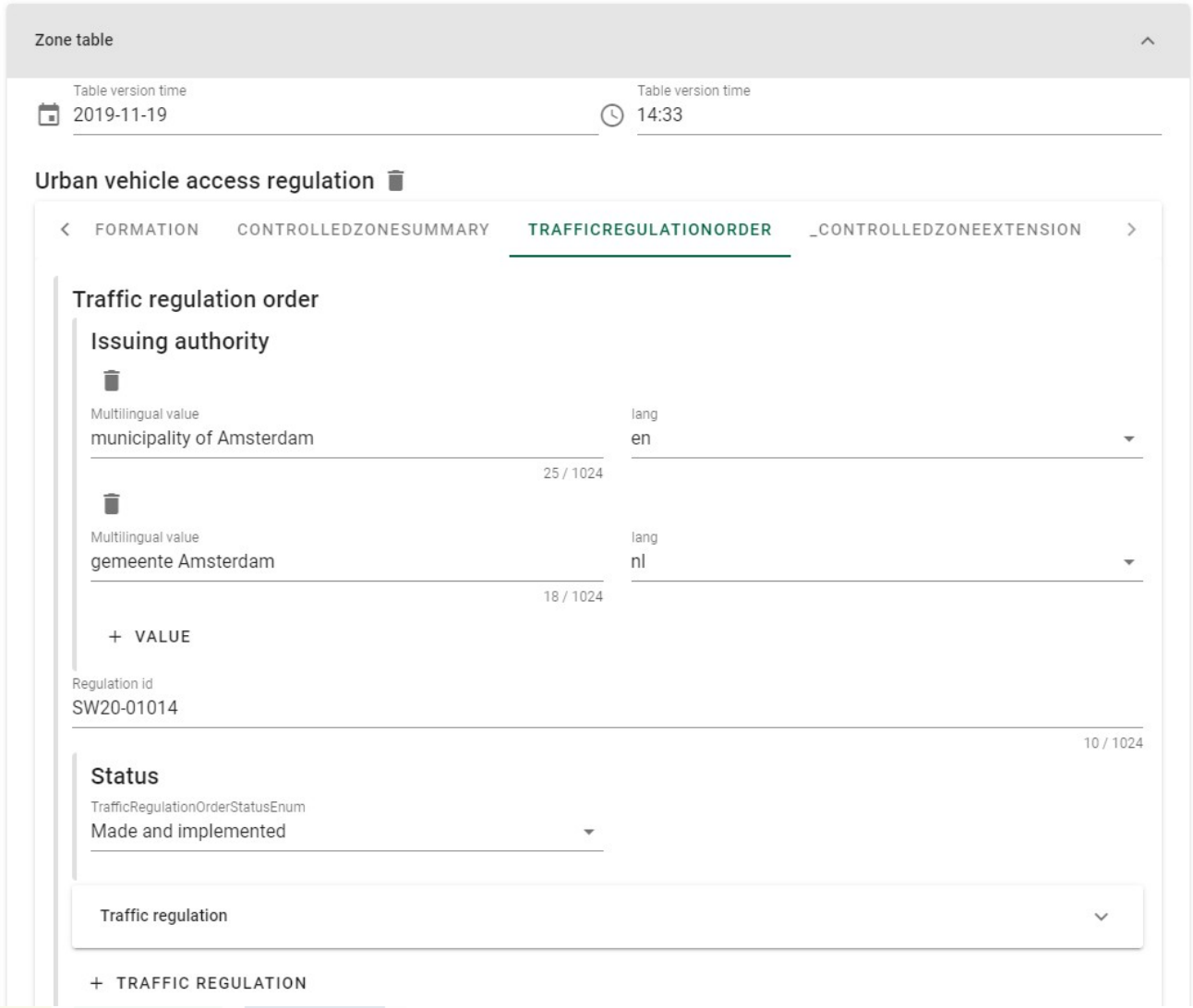

Figure 19: UVAR form - zone table – traffic regulation order

The regulations that are issued via the traffic regulation order are defined in one or more traffic regulations. The traffic regulation has a type of regulation, which in this example is an access restriction type "No entry" - i.e. a ban (for certain vehicles). A Low Emission Zone can, for example, consist of road closures for a whole area/zone with certain exemptions (e.g. vehicles that meet the emissions standards).

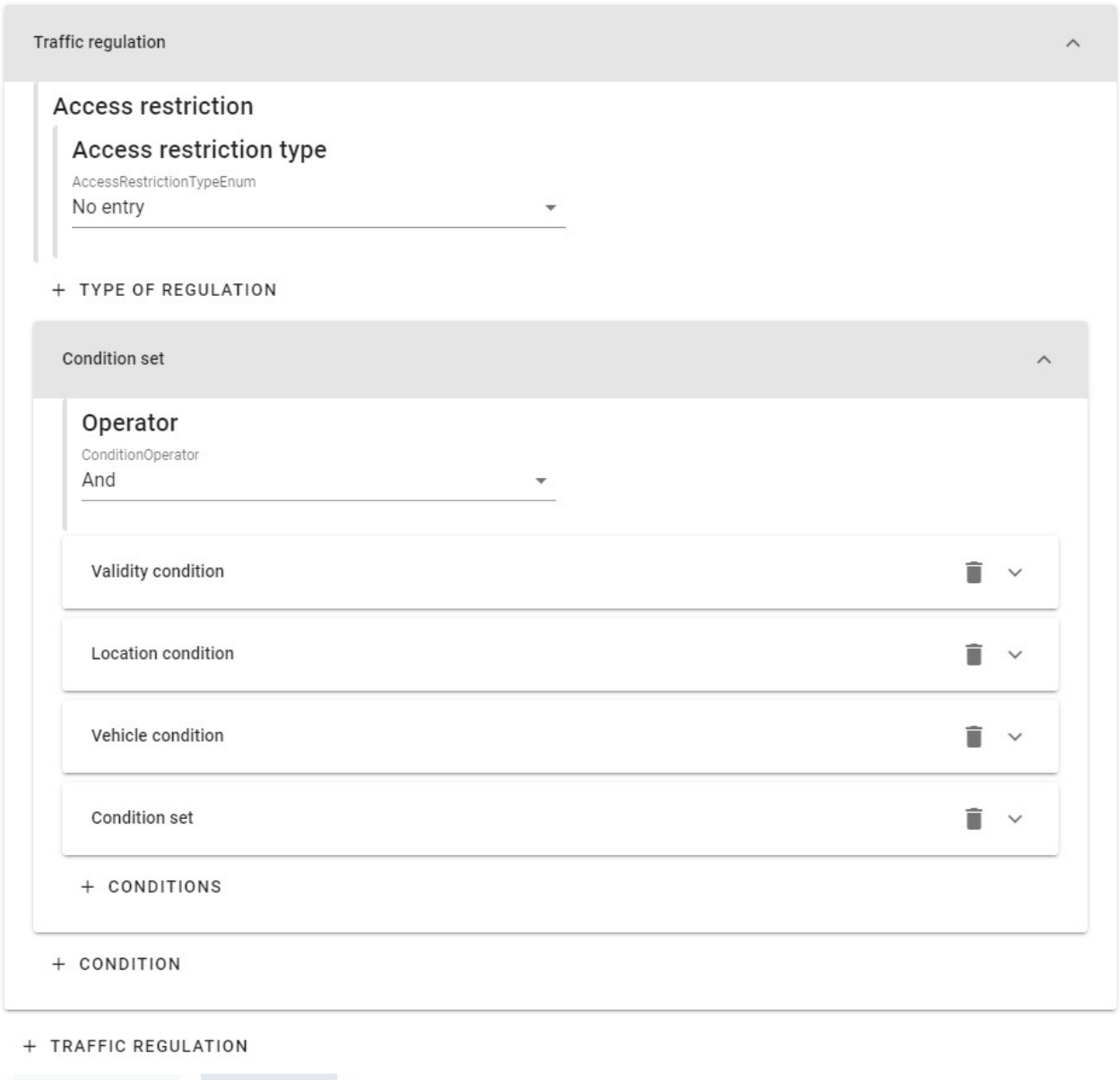

Figure 20: UVAR form - zone table – traffic regulation

The applicability is checked against a set of conditions that ALL need to be met for the restriction to become applicable. Hence the conditions are represented by a set of conditions that are combined by an AND operator. The following figure gives an overview of the condition model structure for UVARs.

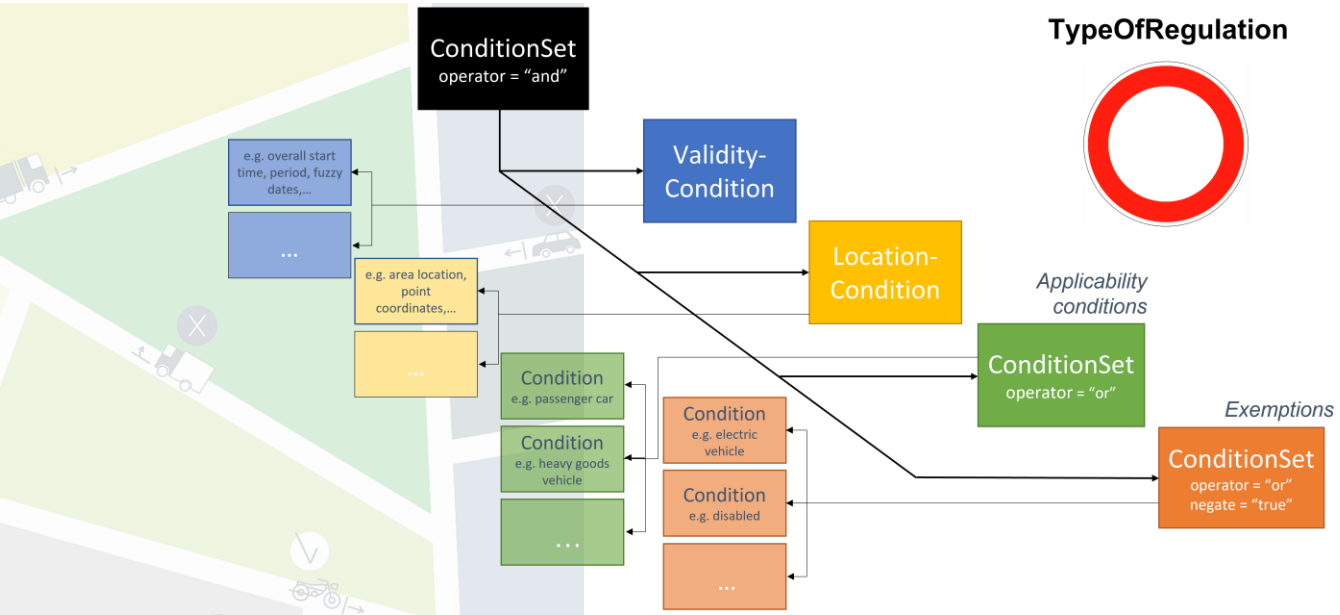

Figure 21: The "Condition" model structure for UVARs

In the condition set:

- the Validity condition defines the time validity of the traffic regulation
- the Location condition defines the location of the traffic regulation, see 2.3 Geometry/location
- the Applicability conditions define all conditions under which the traffic regulation applies (e.g. a Vehicle condition to define a specific vehicle type)
- a condition/condition set which includes all exemptions of the traffic regulations. Here the *negate* attribute is set to *"true"* and the *operator* attribute to *"or"*. This means that if one of the exemptions holds, the overall *ConditionSet* is *"false"* and therefore the conjunction in the topmost *ConditionSet* becomes *"false"* as well. Thus, the traffic regulation does not apply.

The user has to set at least the details of the validity and location condition, provided that the applicability conditions and exemptions are already set by the underlying template. If special permits or conditions that differ from the template apply, specific conditions or information about permits can be deleted, edited or added.

If the template used has more gaps because there are more variations possible, then more data is needed to be filled by the user.

When all necessary information is filled in the user can save the UVAR. For this purpose, the buttons "Save" and "Save and exit" are available. By pressing "Save and exit" the form will be closed after saving and the list will be shown. Pressing the "Cancel" button closes the form without saving.

Figure 22: Buttons "cancel", "save", and "save and exit"

When saving, a validation according to the DATEX II UVAR profile takes place. In case of a negative validation a dialog pops up on top of the list. By pressing the arrow button besides the invalid field names the user can jump to the corresponding elements to add the missing/invalid information.

Figure 23: validation message

Even if the validation fails, the data entries can still be saved to allow the user to add missing information at a later stage by choosing "Save anyway".

The validation status is saved with the UVAR. As a result of the use case the new UVAR is displayed in the list and on the map.

2.3 Geometry/location

In the UVAR Box tool location information can be defined in the Location Condition of a traffic regulation, see 2.2 Creation of a new UVAR. The spatial extent can be defined as an area (polygon). For this purpose, the following use cases are supported by the UVAR Box tool:

- Manually digitize the geometry
- Add a geometry from an existing Shapefile
- Edit an existing geometry

Figure 24: Geometry-related icons

- Zoom to geometry (1): Zoom the map to the geometry.
- Draw geometry (2): Draw/Edit a geometry: The geometry can be manually digitized and edited by the user. To digitize a geometry the user has to click on the map. Each click on the map adds a new vertex to the geometry. At least three vertices are needed to define a polygon. The last vertex has to be added by a double-click on the map. This ends the edit geometry mode. To insert a new vertex the user has to click on the connecting line between two existing vertices,

when editing mode is active. The position of a vertex can be changed by drag-and-drop of the vertex. To delete a vertex the user has to click on the vertex while pressing the Alt-Key.

- Add geometry from file (3): Add a geometry from an existing Shapefile (containing polygons)

Additional information about Shapefiles: Shapefile is a data format for storing geographical information and be can edited/created using standard GIS tools, e.g. QGIS. Various datasets containing administrative boundaries are available on EU/country/regional level, e.g. published by the respective authorities.

2.3.1 Add geometry from file

The "Add geometry from file" dialog can be triggered by clicking the respective icon. A prerequisite for this use case is an existing Shapefile (stored in a ZIP-file), defining geometries as polygons .

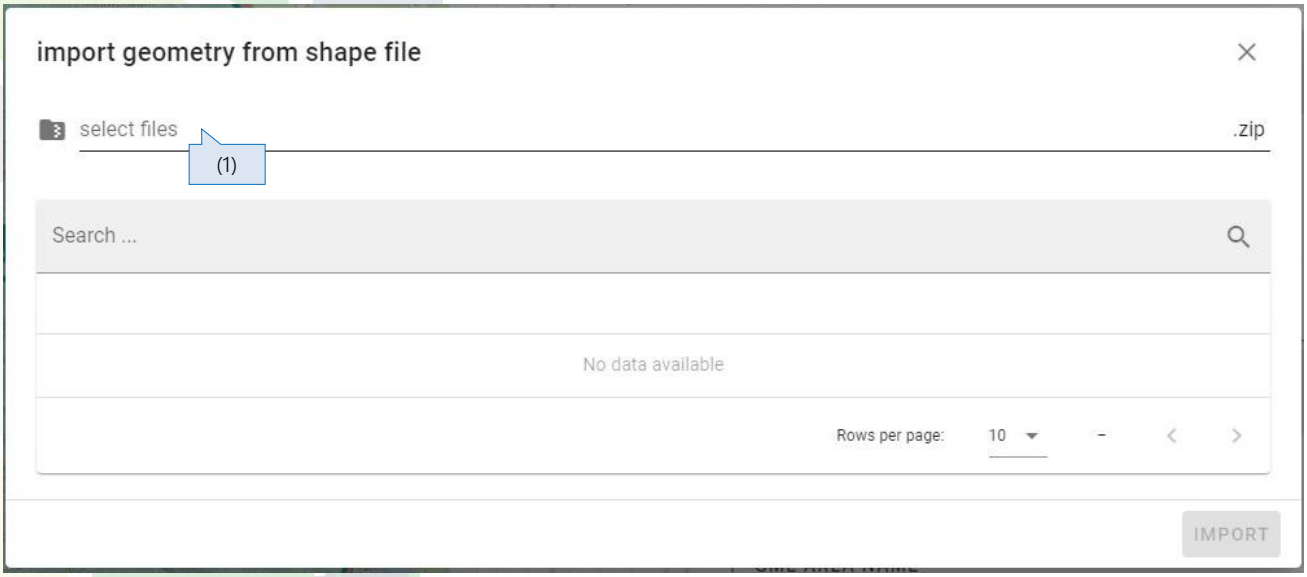

Figure 25: Add geometry from file – select file

- Select file (1): select file to use. Only ZIP-files are allowed, ZIP-file must contain at least files with the following filename extensions: *.shp, *.shx, *.dbf and *.prj

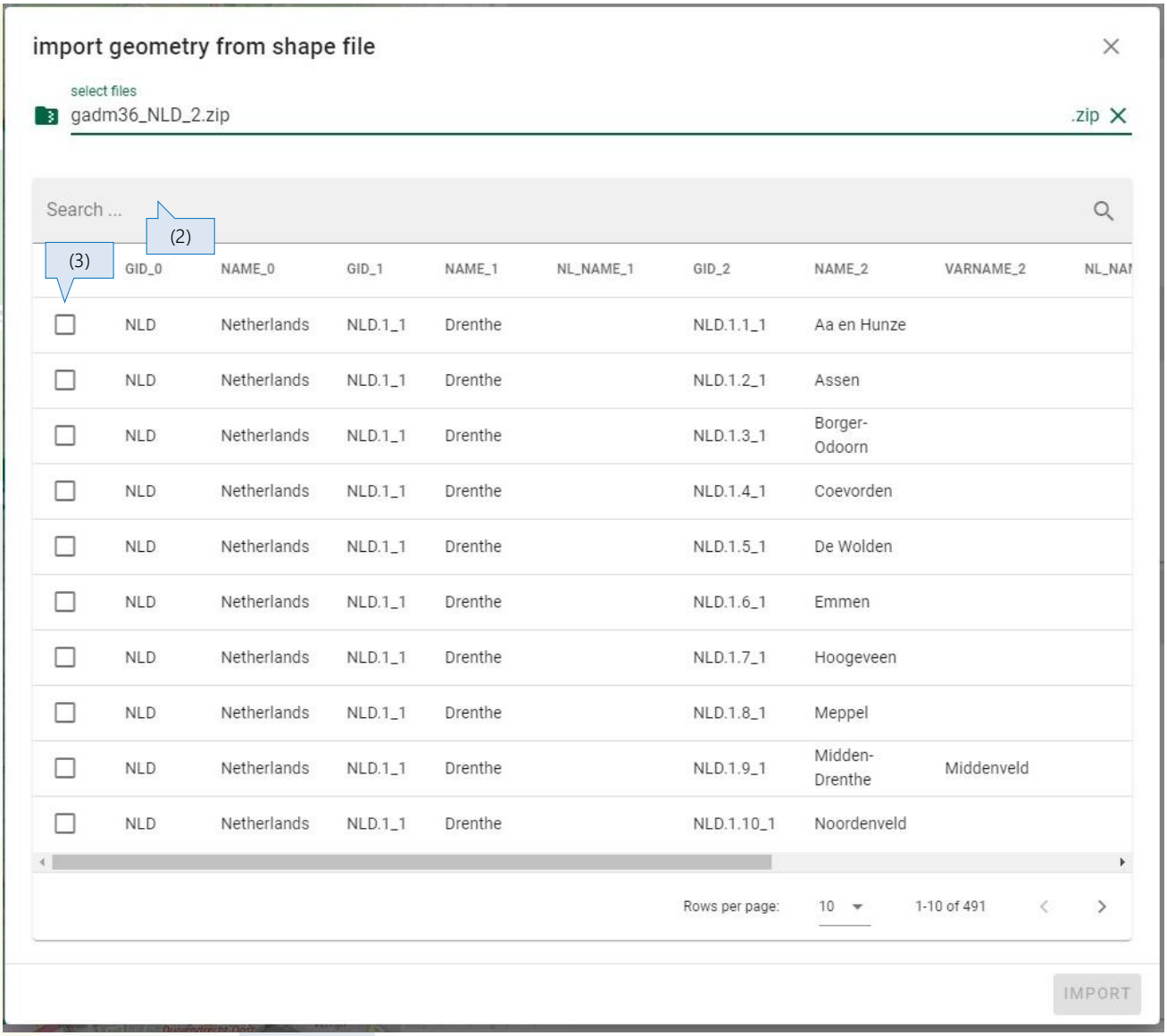

Figure 26: Add geometry from file – select feature

- Search features (2): search features for specific attribute values (e.g. name of city/region). The search enables the user to search the list of features for specific values. The entered search term filters the list of items to the features matching the search.
- Select feature (3): select feature/geometry to be imported
- Pressing the "Import" button adds the geometry of the selected feature and zooms/pans to it.

3 Edit an existing UVAR

Editing of an existing UVAR (either imported or created on the tool) is possible. The form of the UVAR can be entered by pressing the Edit UVAR icon. The form opens and the UVAR can be edited as seen in the previous use cases.

4 Templates

A template refers to a specific UVAR type with conditions, attributes and values prefilled. Regarding templates the following use cases are supported by the UVAR Box tool:

- Create an UVAR (based on existing template)
- Save an existing UVAR as a template: template is saved with the attributes of the UVAR, but stripped of from its geometry, contacts, institutions and dates.
- Manage and edit templates
- Clone templates

Basically, there are three ways to create a template. It can be generated by using an existing UVAR with the "Save UVAR as template" functionality, by cloning an existing template or it can be generated outside of the UVAR Box tool as an XML-document and imported using the backend interface of the UVAR Box tool.

Figure 27: Template-related icon

4.1 Save UVAR as template

To create a new template on the tool the user role has to have the privilege for this functionality. An icon "save as template" is shown in the list in this case. After pressing the icon, the user is prompted to input the name of the template. On confirming the given name, the UVAR is saved as a template with the attributes of the UVAR, but stripped of the information regarding dates, geometry and authorities.

4.2 Manage and edit template

To manage templates the user has to press the "edit templates" icon besides the "+" button for creating a new UVAR. The list of UVARs switches to a list of templates. Editing a template works in the same way as editing an UVAR. A well-defined template always has to have a specific UVAR type (Controlled zone type) attached to it.

4.3 Clone template

To clone an existing template the user has to press the "save as template" button in the template list.

5 Administrative functions

Administrative functions regarding user management, management of layout, client/server configuration, DATEX II profiles for validation etc. are described in the technical documentation of the UVAR Box tool.

6 Application Design

UVAR Box tool is designed as a Client/Server Web-application which communicates via REST Interfaces.

The client is developed as single-page-web application in the Javascript-Framework "VUE.js", using Openlayers as Map component.

Frameworks/libraries:

- Vue.js - JavaScript Framework
- Vuetify - Component/Design Framework
- OpenLayers – JavaScript map library
- OpenStreetMap – WMS - basemap

The server is a J2EE Application, using Hibernate as Database Layer. The server provides an administrative UI as html-pages and forms.

Frameworks/libraries:

- Modular java-application
- compiled and packaged with Maven
- Java-EE server (Wildfly) – application server
- JAX-RS (Rest services)
- Jackson JSON processing
- Java API for XML Processing
- JTS Java Topology Suite
- GeoTools – Java GIS toolkit
- Hibernate Spatial

As storage PostgreSQL with PostGIS extension is used.

The documentation of the REST Interfaces is autogenerated from Java Code and can be viewed via the administrative UI.

https://server/UVAR/serverinfo/ -> Swagger-UI

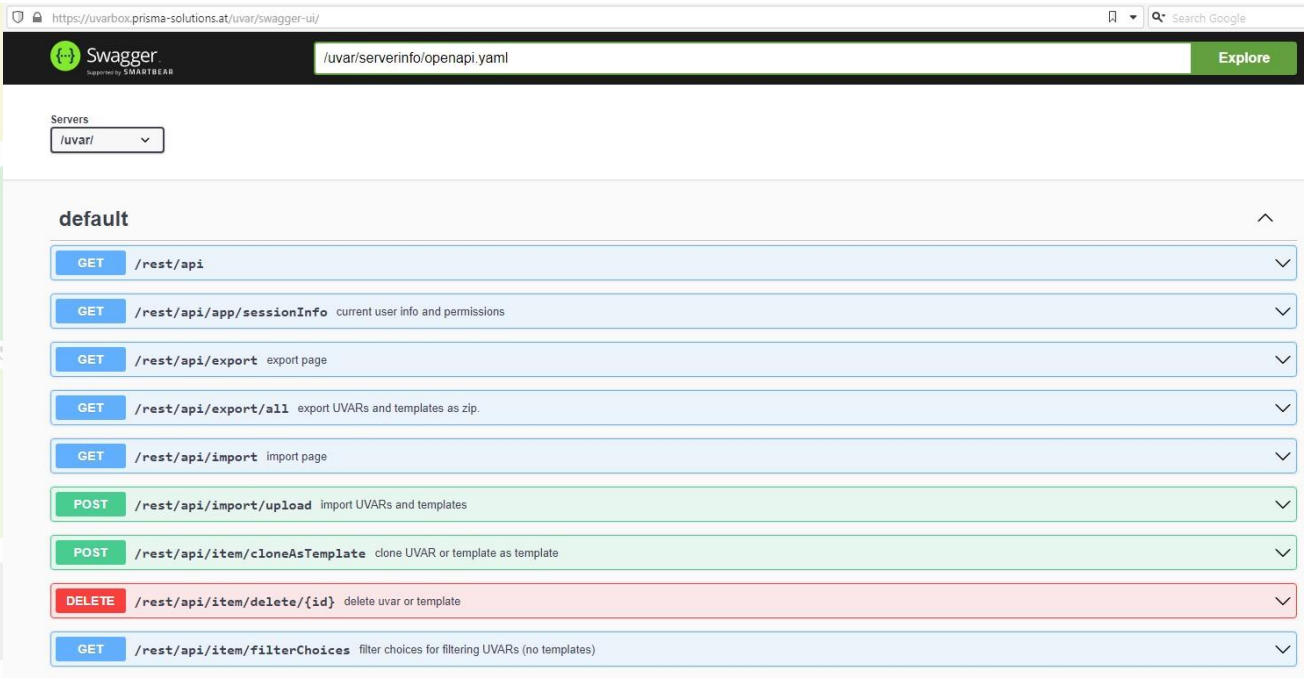

Figure 28: API documentation - Swagger UI

6.1 Technical Requirements for UVAR Box Server

For hosting the UVAR Box tool the following technical requirements to apply.

Docker and Database Host:

- Linux, 6-8 GB Ram, 15 GB HD
  a. Docker version 20.10.12
  b. docker-compose version 1.29.2
- „UVAR" user (ssh), in the „docker" group with sudo permissions.

Database:

- PostgreSQL 12.9 - possibly on the same vm, otherwise Linux with 2GB Ram (5GB HD)

6.2 Building and Deployment

The Software relies on the "Maven" build tool. Scripts to compile the software and to build the docker-container are provided with the sources:

cd build

./compile_and_build_image.sh

For launching the container an example for docker-compose is provided with the sources:

UVAR/docker/sample-compose/docker-compose.yaml

7 Database Design

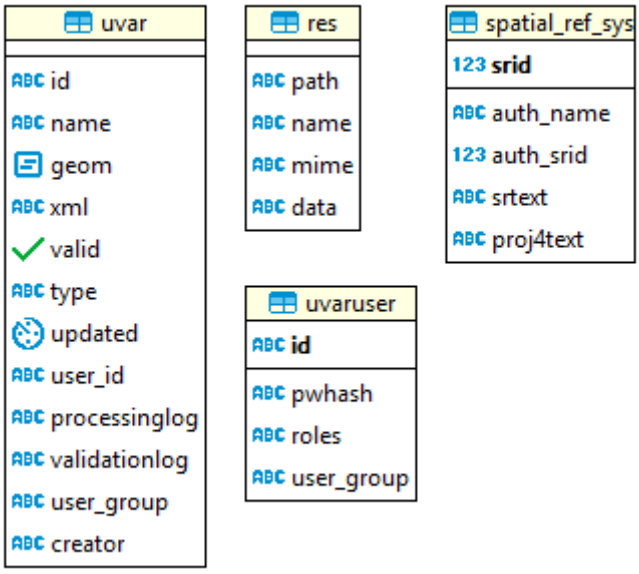

Figure 29: Database tables

7.1 "UVAR" Table for UVARs and Templates

The UVAR Table stores both UVARs-Documents as well as UVAR-Templates. UVARs-Documents can be valid regarding the Reference XSD Profile or invalid (incomplete). Templates are usually incomplete.

There are two types of Templates:

- Those which appear in the List to create new UVARs ("+" button)
- Those which only appear in the "edit Templates" mode and are used as Basis for other Templates. They are the "hidden" templates. A typical "hidden" Template is "Template 0", which technically is an empty d2:payload XML node.

The Fields which have "derived" content are marked with a "*".

Table 1: UVAR table

Column	Datatype	Description
ID	String	Primary Key. Can be any text. Templates have the prefix "template::" or

		"template_hidden::" for hidden Templates
TYPE*	String	Type of the UVAR - derived from the DATEX II XML, using XPath
NAME*	String	Title of the UVAR or Template. For UVARs the Name is derived from the DATEX II XML, using XPath. For Templates the user can edit the name (it is not derived).
GEOM*	Polygon	Geometric location of the UVAR. Derived from the XML using XPath – Union of all Geometries.
XML	String	DATEX II Document (complete and available for validation)
VALID*	Boolean	UVAR Document is validated against XSD Profile when storing or importing it.
UPDATED*	DATE	Timestamp, this Object has been changed.
USER_ID	Foreign key (String)	Refers to the User-ID (see UVARusers-table) of this UVAR who did the last editing.
PROCESSINGLOG*	String	Logging Output of the last attempt of extracting Data from the XML (to fill the NAME and GEOM column).
VALIDATIONLOG*	String	Logging Output of the last XSD-Scheme Validation

An example for a base template to derive other templates is "Template 0" (with the id "template_hidden::0"):

```
<?xml version="1.0" encoding="UTF-8"?>
<d2:payload xmlns:cz="http://levelC/schema/3/controlledZone"
        xmlns:com="http://levelC/schema/3/common"
xmlns:tro="http://levelC/schema/3/trafficRegulation"
        xmlns:loc="http://levelC/schema/3/locationReferencing"
        xmlns:comx="http://levelC/schema/3/commonExtension"
        xmlns:locx="http://levelC/schema/3/locationExtension"
        xmlns:d2="http://levelC/schema/3/d2Payload"
xmlns:sit="http://levelC/schema/3/situation"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
        xsi:schemaLocation="http://levelC/schema/3/d2Payload
LevelC_3_D2Payload.xsd"
        xsi:type="cz:ControlledZoneTablePublication"            lang="en-US"
modelBaseVersion="3">
</d2:payload>
```

7.2 "UVARUSER" Table defining Users and their Roles

Table 2: User table

Column	Datatype	Description
ID	String	Unique Identifier of the User. Email-Address, for instance.
PWHASH	String	MD5 encoded password.
ROLES	String	Comma "," separated list of Roles (as described below)
USER_GROUP	String	group name of users, who share permissions on group of items

7.3 "RES" Table for Schema and Configuration-Files

The Resource Table contains XSD-Schema files and Configuration Files for the Client and the Backend. The Files are organized in a "one level" directory structure.

Table 3: Resources table

Column	Datatype	Description
PATH	String	Path (could also be seen as Namespace)
NAME	String	Name
MIME	String	Mime-Type (text/xml or application/json)
DATA	String	File Data. UTF-8 encoded Text. No binary files at the moment.

7.4 Required Configuration Files

Sample documents are provided with the sources.

Table 4: Configuration files

Path	Name	Dedication
client	config	Client specific Configuration • default Map-Extent

		• Documentation URL • Nominatim URL
server	config	Server specific Configuration: • specific service XSD schemes • Xpaths to extract information from XMLs: ◦ caption, type and geometry • Xpath(s) for stripping of information when creating templates from UVARs.
layout	standard	Layout definitions for building forms from the schemes.
locale	en	Internationalization English. Key/Value notation in JSON. Text Labels for XSD Classes, attributes (Class.attribute) , Enums and values (Enum.value).
locale	xx	Internationalization other languages
Xxx	xxx.xsd	Packages of XSD Schema Files, as linked in the server.config.

client.config (JSON)

The client configuration parameterizes client-side components like the map.

```
{
  "defaultExtent": [642485.7685442179, 6531845.468531775, 708748.3442912741,
6641531.17902371],
  "documentationUrl":
"https://drive.google.com/file/d/1VypiawZrLJ2_lWD9NXB3W2iy1Ezi8RmG/view?usp=sha
ring",
  "nominatimUrl": "https://nominatim.openstreetmap.org/search"
}
```

server.config (JSON)

Parametrizes the server side components. Affects the client also - by providing the appropriate information to the client for building the form for a particular UVAR.

The section "uvarTypes" defines which set of XSDs and resource files are used for building up the form and doing the validation (validation is done client- and server-side).

The section "extractXMLInformation" defines

- how the server extracts information from the UVAR XMLs like the caption, the type and the geometry, particularly for the list-view.
- Which information to strip of the UVAR-xml when creating templates.

```
{
	"uvarTypes": {
		"*": {  // base uvar profile (used as fallback)
			"xsdResPath": "xsd",
			"xsdResNames": [
				"LevelC_3_D2Payload.xsd",
				"LevelC_3_Common.xsd",
				"LevelC_3_CommonExtension.xsd",
				"LevelC_3_ControlledZone.xsd",
				"LevelC_3_LocationExtension.xsd",
				"LevelC_3_LocationReferencing.xsd",
				"LevelC_3_TrafficRegulation.xsd"
			],
			"layoutResPath": "layout",
			"layoutResName": "standard",
			"localeResPath": "locale",
			"localeResName": "en"
		},
		"lowEmissionZone": { // type specific editing profile (set of xsds)
			"xsdResPath": "LEZ",
			"xsdResNames": [
				"LevelC_3_D2Payload.xsd",
				...
			],
			"layoutResPath": "layout",
			"layoutResName": "standard",
			"localeResPath": "locale",
			"localeResName": "en"
		},
		"limittedTrafficZone": {
			...
		},
		"restrictedParkingZone": {
			...
		},
		...
	},

	"extractXMLInformation": {
		"xpathPrefixMap":  {    // prefix mapping required for the xpath expressions
			"xsi": "http://www.w3.org/2001/XMLSchema-instance",
			"d2": "http://levelC/schema/3/d2Payload",
			"cz": "http://levelC/schema/3/controlledZone",
			"com": "http://levelC/schema/3/common",
			"tro": "http://levelC/schema/3/trafficRegulation",
			...
		},
		"captionXPaths": [// xpath to determine uvar caption (fallback chain, the first successful xpath wins)

	"/d2:payload/cz:controlledZoneTable/cz:urbanVehicleAccessRegulation/cz:name/com:values/com:value[@lang='en']",

"/d2:payload/cz:controlledZoneTable/cz:urbanVehicleAccessRegulation/cz:name/com:values/com:value[@lang='it']",
```

```
                    ...
            ],
            "typeXPaths": [ // xpath to determine uvar-type (fallback chain, the
first successful xpath wins)

      "/d2:payload/cz:controlledZoneTable/cz:urbanVehicleAccessRegulation/cz:co
ntrolledZoneType",

      "/d2:payload/cz:controlledZoneTable/cz:urbanVehicleAccessRegulation/cz:tr
afficRegulationOrder/tro:trafficRegulation/tro:typeOfRegulation/tro:accessRestr
ictionType"
            ],
            "geomXPath":    "//d2:payload/descendant::loc:gmlMultiPolygon",    //
xpath collect polygons to form compound multipolygon geometry
            "templateRemoveXPaths": [ // information to strip of the UVAR-xml
when creating templates

      "/d2:payload/cz:controlledZoneTable/cz:urbanVehicleAccessRegulation/cz:na
me",
                        ...
                ]
      }
}
```

locale.xx

The "locale.xx" files are used for providing text labels for the user-interface. Example: locale.en.

They are a flat JSON key/value structure.

- Titles, labels and tooltips of standard user interface elements
- Labels of data elements as defined in the XSDs:
    - Complex types (containers)
    - Simple types (widgets for text, numeric, date entry)
    - Enumeration values

      XML elements are linked with their Complex-Type plus element name or just element name

      Enumerations are linked with "EnumerationType.enumValue".

```
{
  "hideMap": "hide map",
  "showMap": "show map",
  ...
  "AccessConditionTypeEnum.accessOnly": "Access only",
  "AccessConditionTypeEnum.destinationTraffic": "Destination traffic",
  ...
}
```

layout.standard

The form component is built from the XSDs schemes. The layout configuration enables more fine-grained control which goes beyond the standard behaviour:

- Hiding specific elements.
- If containers are organized as “container”, “tabs” or “extension panels”.

```
{
  "_ExtensionType": {
    "hidden": true
  },
  "_vehicleCharacteristicsExtension": {
    "hidden": false
  },
  "TrafficRegulation": {
    "displayAs": "expansionPanels"
  },
  "ControlledZone": {
    "displayAs": "tabs"
  },
  "id": {
    "hidden": true
  },
  "version": {
    "hidden": false
  },
  "targetClass": {
    "hidden": true
  },
  ...
}
```

8 Authentication and Permission-System

UVAR Box uses the “Basic HTTP authentication” mechanism. Everything below the application root context (/UVAR/*) is protected, except for users which send the appropriate username/password combination listed in the UVARUSER table.

The stateless “Basic HTTP authentication” was chosen for simplicity and for making access from 3rd party systems (import/export) easy.

Specific Server-API access from the client or 3rd party systems requires the user to have roles defined in UVARUSER table as described below:

8.1 Description of Roles

Table 5: Description of roles

Role-Name	Description	Affected Endpoints

Read	Can List UVARs and open the Form in Read-Only Mode	/rest/api/app/* /rest/api/item/* /rest/api/res/*
Write	Can open the UVAR Form in Edit-Mode, validate and save them. Can create/manage templates and edit them.	/rest/api/item/*
Template	Can manage Templates and create them from UVARs	/rest/api/item/*
Import	Can use the Import Interface	/rest/api/import/*
Export	Can use the Export Interface	/rest/api/export/*
Admin	Can use the Administrative API/UI for managing Users, Resources and UVARs. Can Edit, Delete "hidden" Templates	/rest/api/itemmgr/* /rest/api/resmgr/* /rest/api/usermgr/*
Edit_all	Can edit UVARs and templates of all groups	/rest/api/item/* /rest/api/import/*

8.2 Sharing permissions on group of items

Each user is assigned to a group (table UVARUSER, attribute USER_GROUP). When a UVAR/template is created/imported/cloned, the creator (= logged-in user) and the group of the creator are stored with the UVAR/template (table UVAR, attributes CREATOR, GROUP). The UVAR/template can be edited by users of the same group. Assigning the Edit_all role to a user allows the user to edit UVARs of all groups (independent of the user's user_group)

9 Import/Export Interfaces

9.1 UVARs

UVARs can be uploaded or downloaded from the UVAR Box tool as Zip-Files.

9.2 Download

https://server/UVAR/rest/api/export/all?templatesAlso=false

Exports all UVARs. In the Zip-File the UVAR-Ids are used as Filenames. Templates can also be exported, but not their names.

(Export role required)

9.3 Upload

9.3.1 Import of a fully digitised UVAR (DATEX II)

(as multipart form-data)

https://server/UVAR/rest/api/import/upload

When uploading UVARs you can choose to validate them first – all UVARs inside the zip will be rejected if a single validation fails.

As the UVAR-Ids are the filenames in the ZIP, UVARs with the same ids will be updated on import. This can be used for Backup/Restore.

(Import role required)

9.3.2 Import of semi-structured data

https://server/UVAR/rest/api/semistructimport/upload

Data in the following structure is needed as input (stored in a zip-file):

- UVAR data (Shapefile): containing geometry and attributes
- import template (XML): defining the mapping of the attributes of the Shapefile to the DATEX II UVAR structure. This import template is basically an UVAR template (DATEX II), but contains placeholders to fill in the attributes of the Shapefile.

Placeholders are defined by ${shape_attribute} and are usually mapped as datatype String. Other datatypes supported: date, dateTime, e.g.: ${dateTime:shape_attribute}.

The result of the use case is an imported UVAR based on the import template with placeholders being replaced by the attributes of the Shapefile.

When uploading UVARs you can choose to validate them first – all UVARs inside the zip will be rejected if a single validation fails.

After importing, the UVAR can be edited in the UVAR Box tool and attributes can be edited as needed. For update the service call for the import of a fully digitised UVAR (DATEX II) can be used.

This import concept follows a generic approach which allows

- to import different data sources, as long as they are preprocessed to the specified input format
- the user to define the import template (i.e., mapping to DATEX II) based on the structure and availability of existing data

(Import role required)

9.4 Read

For 3rd party systems, which need an Overview of the UVAR-Documents inside the UVAR-Box as JSON, the REST-API can be used:

`https://server/UVAR/rest/api/item/listAll`
(Read role required)

For an overview of the UVAR Box REST API see the OpenApi/Swagger Docs:

https://server/UVAR/serverinfo/ → Swagger-UI

9.5 Geoserver

Geoserver WMS or WFS can be used to display the features in the "UVAR" table in external GIS systems. For setting up Geoserver with docker-compose see:

https://gitlab.utu.fi/geonode/utu-geonode/-/tree/master/docker/geoserver

Geoserver is not required for the UVAR Box Tool to function.

10 Administration interface

Besides the front end the UVAR Box tool offers an administration and configuration interface, which allows to customize the tool for different deployment scenarios. This involves administration use cases like configuration of the underlying DATEX II data schemas (XSD) per UVAR type, the appearance of the form in the tool, the supported languages and their configuration, user management etc.

The administration interface is available at https://server/UVAR/serverinfo/

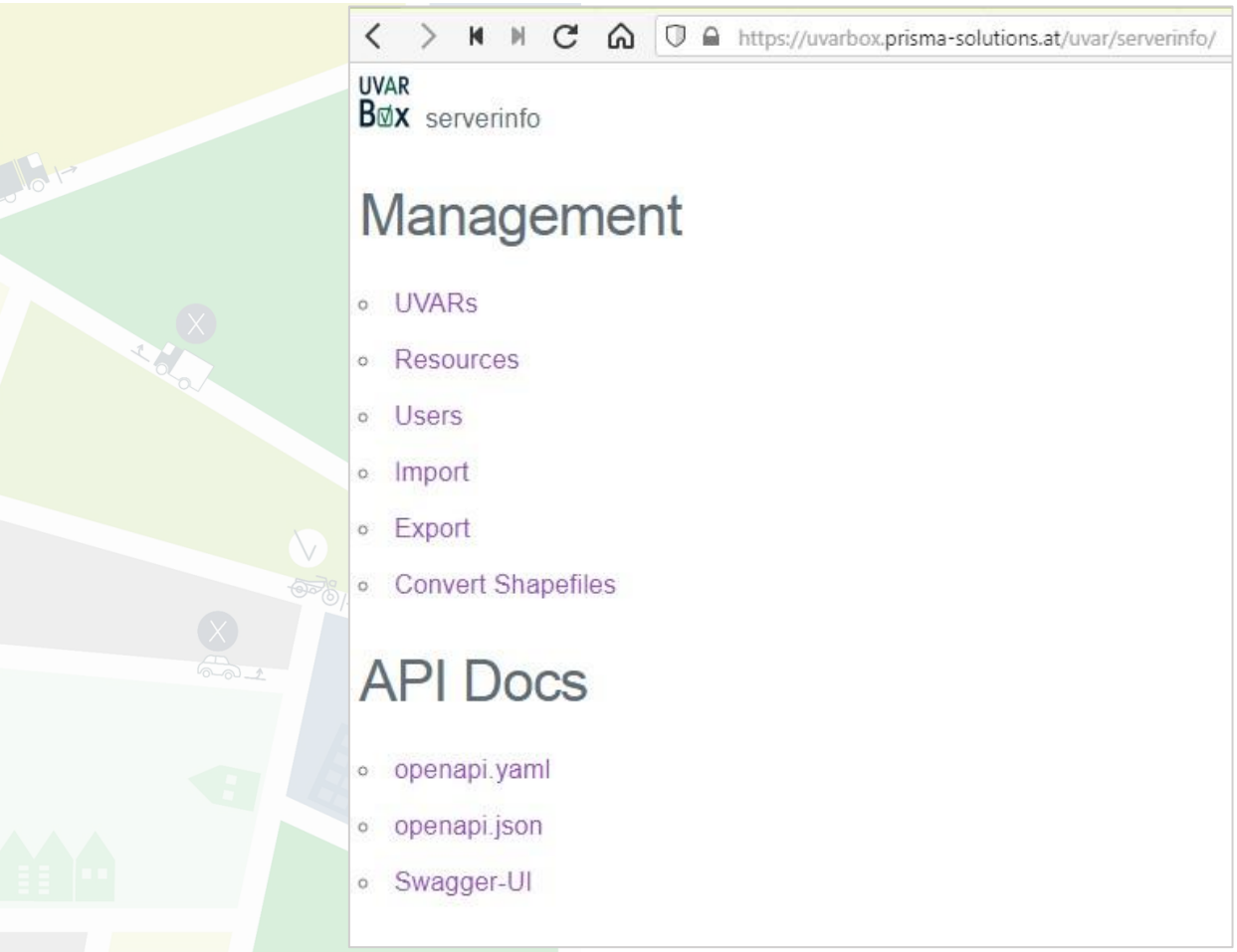

Figure 30: Administration interface

11 Explanations regarding functionality of UVAR Box tool

11.1 Location of an UVAR

DATEX II offers plenty of options to define the location of an object. Details about the location methods for UVARs and the corresponding DATEX II elements are described in detail in D1.2 and D1.3.

In the UVAR Box tool location information can be defined in the Location Condition of a traffic regulation. The spatial extent can be defined as an area (polygon). For this purpose, the following use cases are supported by the UVAR Box tool:

- Manually digitize the geometry
- Import a geometry from an existing Shapefile (containing polygons)
- Edit an existing geometry (either manually digitized or imported)

Additional information about Shapefiles: Shapefile is a data format for storing geographical information and be can edited/created using standard GIS tools, e.g. QGIS. Various datasets containing administrative boundaries are available on EU/country/regional level, e.g. published by the respective authorities.

11.2 Usage of DATEX II profiles and templates

Regarding the usage of DATEX II profiles and templates in the UVAR Box project the following terms have to be distinguished.

- DATEX II standard
- DATEX II UVAR reference profile (set of XML Schema definition files): specified for all UVAR types based on DATEX II standard
- DATEX II UVAR service profiles (set of XML Schema definition files): specified per UVAR type and/or per country/region
- UVAR template (XML): specified corresponding a certain DATEX II UVAR service profile with attributes and values prefilled
- UVAR: specified on basis of a specific template (by cloning the template)

For having different service profiles for different templates there are three alternatives to link them:

- Configuring service profiles on an UVAR-Type basis (current method, UVAR type is derived from the UVAR-Template-XML document*)). Requires having separate UVAR Box instances per country/region if service-profiles are country/region specific.
- Configuring service profiles on an UVAR-Type and Country basis (both properties derived from the UVAR-XML document**)). Doesn't work for region specific service-profiles.
- Storing the reference of the service profile within the UVAR-Template-XML. For instance using a new "UVAR Box" header extension. This reference is cloned when creating a new UVAR.

*) determine UVAR Type using XPath:
/d2:payload/cz:controlledZoneTable/cz:urbanVehicleAccessRegulation/cz:controlledZoneType
**) determine Country using XPath: /d2:payload/com:publicationCreator/com:country

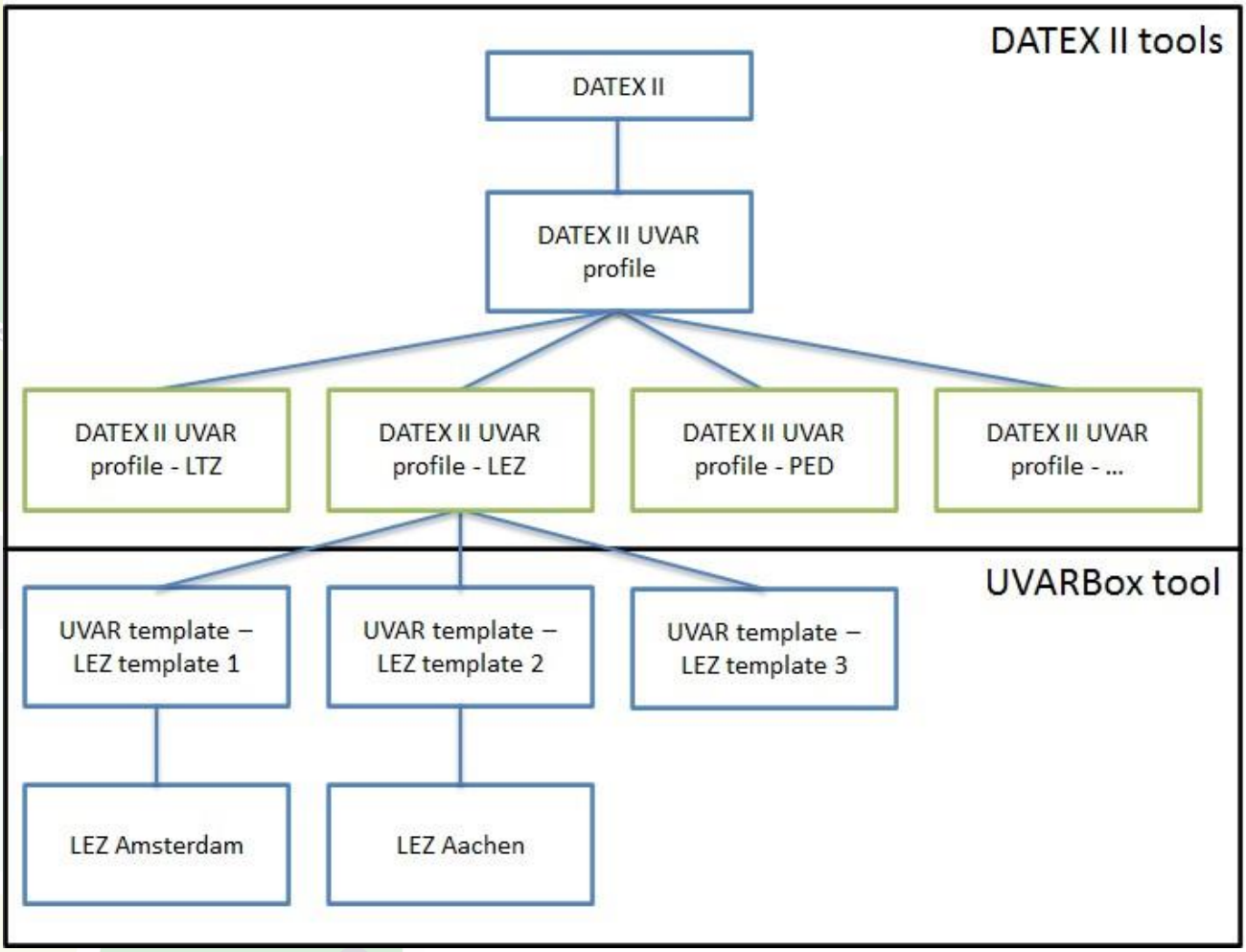

Figure 31: DATEX II profiles and templates

Regarding templates the following use cases are supported by the UVAR Box tool:

- Create an UVAR (based on existing template)
- Save an existing UVAR as a template: template is saved with the attributes of the UVAR, but stripped of from its geometry, contacts, institutions and dates.
- Manage and edit templates
- Clone templates

Basically there are three ways to create a template. It can be generated by using an existing UVAR with the "Save UVAR as template" functionality, by cloning an existing template or it can be generated outside of the UVAR Box tool as a XML-document and imported using the backend interface of the UVAR Box tool.

Creating DATEX II UVAR model and reference/service profiles is not part of the UVAR Box tool as DATEX II offers tools for these purposes.

12 Create a new DATEX II-based Profile for the UVAR Box Tool

12.1 Creating a Profile for the UVAR Box Tool

To create a new profile for the UVAR Box Tool, the DATEX II Wizard (https://webtool.DATEX2.eu/wizard/) must be used. Before using it, the data from the UVARs must first be collected and categorised and mapped to the DATEX II model. In the DATEX II model, the subcategories " PayloadPublication/ ControlledZoneTablePublication/ urbanVehicleAccessRegulation/ ControlledZone " and the sub-subcategory " Trafficregulation " are particularly important for the UVARs and contain the majority of the required attributes.

12.1.1 Source

The current UVAR-DATEX model must be used as source file – Figure 32.

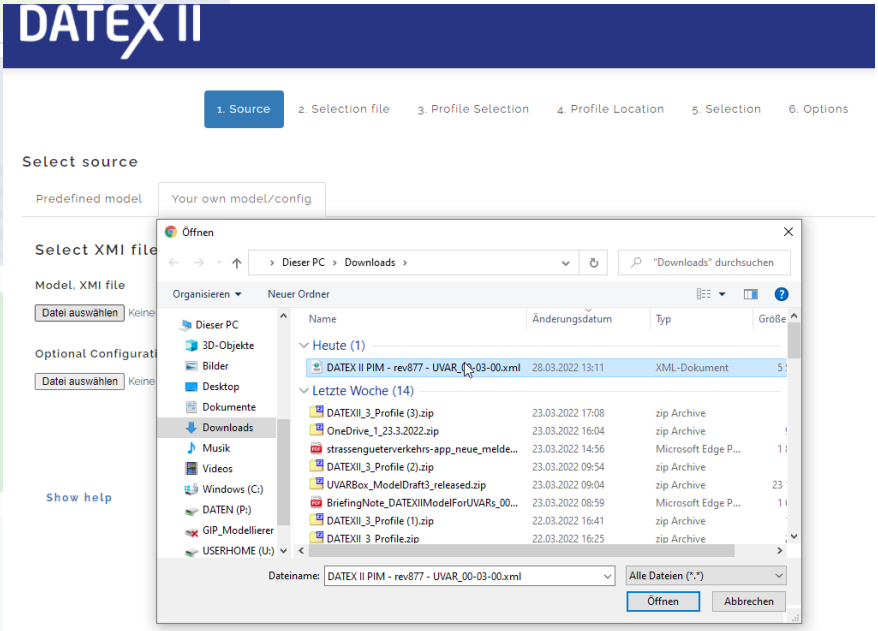

Figure 32: DATEX II Wizard - Source

12.1.2 Selection File

An existing profile (selection.sel) can be used as selection file or it is possible to proceed without a file – Figure 33.

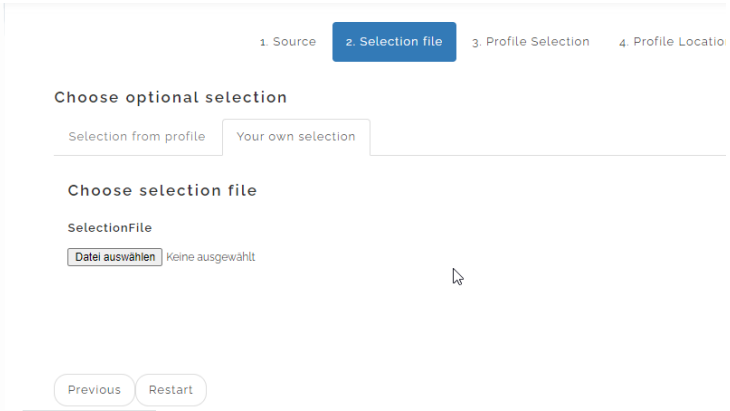

Figure 33: DATEX II Wizard – Selection File

12.1.3 Profile Selection

At this point, nothing should be selected – Figure 34.

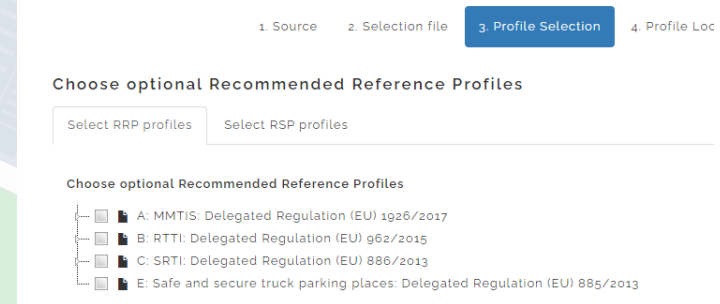

Figure 34: DATEX II Wizard – Profile Selection

12.1.4 Profile Location

The GML multipolygon should be selected as Profile Location – Figure 35.

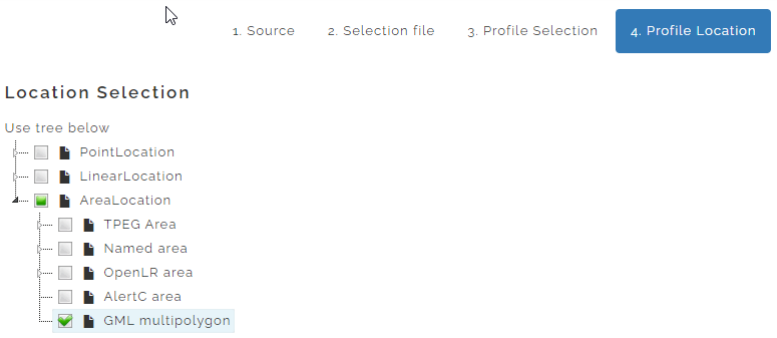

Figure 35: DATEX II Wizard – Profile Location

12.1.5 Selection

Going to the Subcategory "PayloadPublication/ ControlledZoneTablePublication/ urbanVehicleAccessRegulation/ ControlledZone" all the necessary Categories for describing the UVAR must be selected – Figure 36.

All required categories are in "ControlledZone" or "TrafficRegulation". Here it is important to activate "Condition Set" and its attributes "Active" and "Negate". All conditions that are activated under "Condition Set" are also activated under "Conditions". It is also important to activate the option "gmlMultiPolygon" in "LocationReference/ Location/ AreaLocation" for the geometry. Also under "Location/ NetworkLocation/ gmlLineString/ gmlLineString/" all options must be activated, this is needed so that one can draw the location in the UVAR Box Tool.

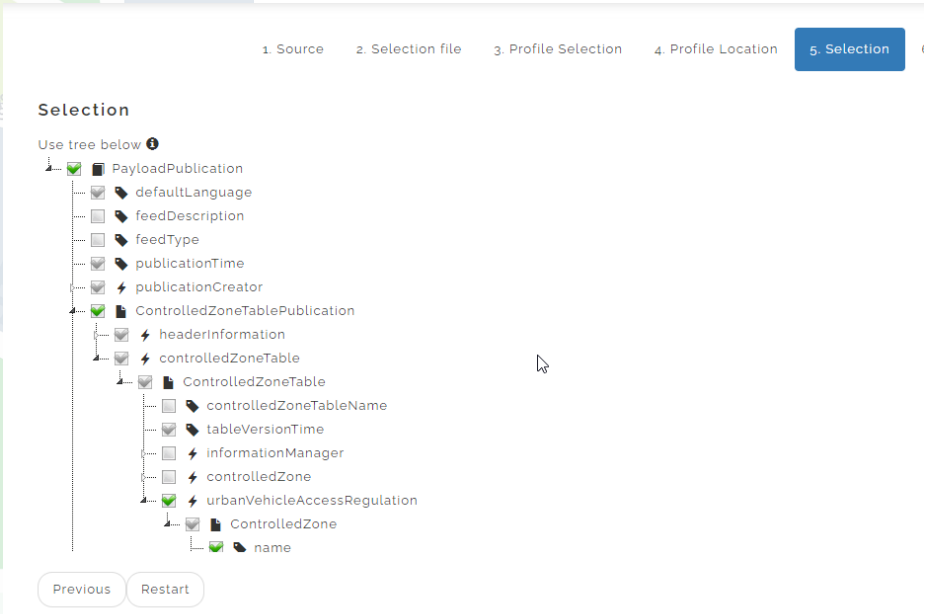

Figure 36: DATEX II Wizard - Selection

12.1.6 Options

Every Option should be selected and continue – Figure 37.

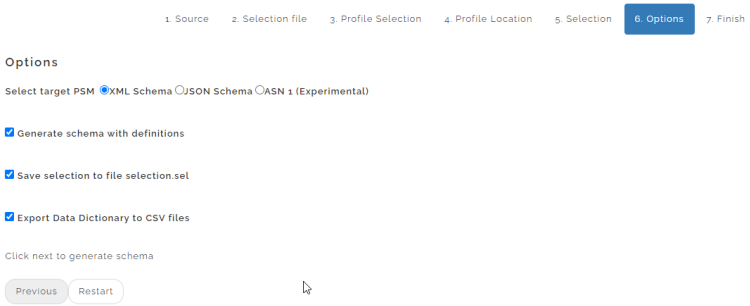

Figure 37: DATEX II Wizard - Options

12.1.7 Finish

The File can be saved and the package downloaded.

12.2 Experience Report - Creation of a DATEX II Profile

In order to create a template for the UVAR Box Tool, an underlying profile is required. Since there was no template and profile for the PARK UVAR yet, this had to be created by AustriaTech so that PARK UVARs could be included in the tool. The profile was created without DATEX II experts, only with the documentation of DATEX II and with PRISMA solutions' support.

Creating a UVAR profile takes some time and effort. There is a very detailed documentation for the DATEX II Wizard (https://webtool.DATEX2.eu/wizard/) and the model. However, there is no step-by-step guide to create a profile. This is probably due to the size and scope of DATEX II and the profiles for the UVAR Box Tool only represent a small part of it.

Therefore, it is necessary to be familiar with documentation and the structure of the model. Then the attributes needed for the UVAR must be found. After this, the profile can be created in the DATEX II Wizard. This may take several tries. as you create profiles, then try to create the existing data as a UVAR and realise that attributes are missing or not configured correctly can be helpful for this iterative process. Creating the profiles was done in close cooperation with PRISMA solutions, which provided valuable input and tips.

After intensive work with the DATEX II Model and Wizard, a working PARK profile was created. For this purpose, it was helpful to create a table in which the attributes required for the regulation were mapped and referenced to the attributes contained in the DATEX II Model. Based on this table, it was possible to decide which attributes should be integrated into the PARK profile. It was also helpful to first collect a variation of different PARK UVARs and to summarise them, since not all partners provided the same set of data and they varied in scope. This way, a wide range of cases could be covered right from the beginning.

To create a new profile from scratch, the DATEX II Wizard, a good understanding of the underlying regulations, a good understanding of the DATEX II Model and which subcategories to work in are needed, as well as time and patience.

Glossary

Term	Definition
API	Application Programming Interface
Docker	set of platform as a service products used to deliver software in packages
GIS	Geographic Information System
JSON	JavaScript Object Notation, open standard file format and data interchange format
MD5 algorithm	hash function producing a 128-bit hash value
MIME type	two-part identifier for file formats and format contents
Nominatim	tool to search OSM data by name and address
PostgreSQL	free and open-source relational database management system
QGIS	free and open-source cross-platform desktop geographic information system application
REST	Representational state transfer
Shapefile	data format for storing geographical information
Swagger	suite of tools for API developers
UI	User Interface
UTF-8	character encoding
UVAR	Urban Vehicle Access Regulation
UVAR Box Tool	Tool to enable the digitisation of UVARs
WFS	Web Feature Service
WMS	Web Map Service
XML	Extensible Markup Language, markup language and file format.
XPath	expression language for querying XML documents
XSD	XML Schema Definition
ZIP-file	Archive file containing one or more files

Interconnections with other work packages

13 Related themes described in other work packages

For details regarding the data model used as basis for the UVAR Box tool see deliverables D1.2 and D1.3. For details regarding the processes of defining UVARs in the different countries and the functional specification of the UVAR Box tool see D2.1 and D2.2. For finding the UVAR Box tool sources see D2.3 UVAR tool package + development report. For specification of UVAR related data-flows for data collection and data maintenance see D2.4.

Template version and print date

Template version used	1.0 01-09-2020
Print date	30-3-2023 7:04