# HUMBOLDT Alignment Editor Manual 2.0.1 (2010/12/14)

Thorsten Reitz (*thorsten.reitz@igd.fraunhofer.de*)

Simon Templer (*simon.templer@igd.fraunhofer.de*)

## Table of Contents

# Introduction to the Project

HUMBOLDT, a European project within the 6th Framework Programme, focuses on the facilitation and support of cross-national harmonisation of spatial data.

Within the HUMBOLDT project, the following goals were used to guide the scientific and implementation work:

1. Capturing domain knowledge / application-specific knowledge required for data harmonisation, including an enhanced formalization of the transformation between two conceptual schemas

2. Supporting the definition of the information product (target schema, SRS, spatial extent, …) to which the processed data needs to be transformed

3. Handling of transformation needs as part of the overall processing of an information request

4. **Enhancement of the automation of the data harmonisation processes** (depending on the possibility of capturing the required knowledge and the availability of this knowledge)

As already indicated in the list of goals, a main aspect is the involvement and formalisation of human knowledge for the geodata harmonisation process. Mainly the knowledge of GI data integrators as well as GIS developers has to be captured. The creation of a target data model (e.g. from an INSPIRE data theme specification) and the analysis of available data sources have to involve application experts' knowledge about data interpretation and GIS developers' know-how about data processing. The expertise of GI data integrators is required again when the mapping between source and target is specified.

For this purpose, three main components were designed and implemented in HUMBOLDT to support the acquisition of users' knowledge for data modelling and harmonisation as well as the subsequent execution of conceptual schema transformation:

– HUMBOLDT GeoModel Editor

– HUMBOLDT Alignment Editor

– HUMBOLDT Conceptual Schema Transformer

While the HUMBOLDT GeoModel Editor supports the specification of source and target models, the HUMBOLDT Alignment Editor facilitates the specification of the mapping between source and target, the HUMBOLDT service called Conceptual Schema Transformer is responsible for the execution of the transformation.

## Getting and Providing Feedback

If you find that there is any information missing from this handbook, you are invited to visit the HUMBOLDT project's community website:

*http://community.esdi-humboldt.eu*

This website provides you with additional documentation as well as several options of getting support or feedback on an issue you are encountering. You can use the website to do the following:

– Read additional specifications

– Register to get notifications for any updates on HUMBOLDT software

– Discuss issues and ideas with the HUM-BOLDT developers on a mailing list (*dev@esdi-humboldt.eu*)

– Create and follow new feature requests and bug reports, as well as support requests.

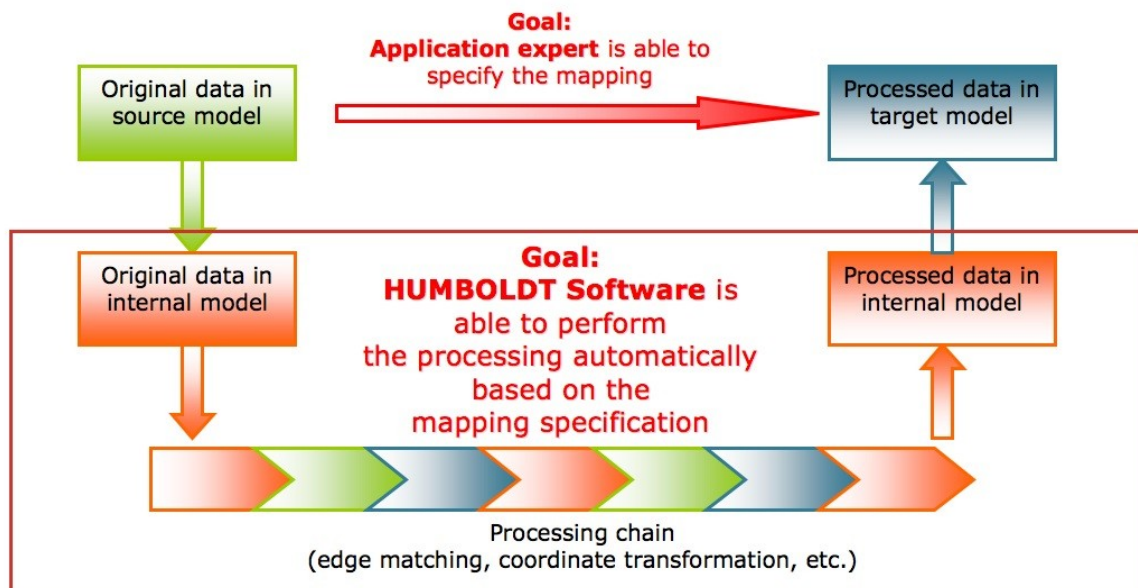The community website furthermore collects news and development activity for each software component.



Figure 1: Schematic overview on the goals of HUMBOLDT concerning data harmonisation

# Introduction to the HUMBOLDT Alignment Editor

## What can the HUMBOLDT Alignment Editor do for you?

The HUMBOLDT Alignment Editor, short HALE, is a rich graphical user interface for defining mappings between concepts in conceptual schemas (application schemas created with the HUMBOLDT Model Editor or other modelling applications), as well as for defining transformations between attributes of these schemas. These mappings are expressed in a high-level language called Ontology Mapping Language (OML) and can later be used by the Conceptual Schema Transformer processing component to generate an executable form of transformation, such as an XSLT for XML input/output.
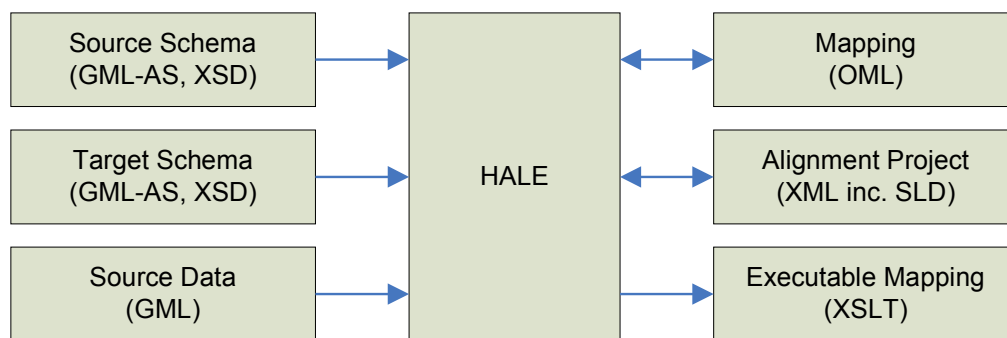
*Supporting Data harmonisation*

The HUMBOLDT Alignment Editor is a core part of the HUMBOLDT Harmonisation toolkit. HALE (and the component executing your specified mappings, called the Conceptual Schema Translation Service CST) allows you to resolve several different interoperability issues:

- Application schema and Terminology
  ◦ Mapping rules for the classes and attributes of a source to a target conceptual schema



*Figure 2: Main input and output of HALE*

To make the mapping process more accessible to a domain expert and to increase the quality of transformations, *HALE allows working with sample source geodata for interactive visualization and validation.* Furthermore, *a task-based system* as it is often used in programming *supports users in the creation of a mapping*.

HALE also provides other innovative capabilities, such as the *real-time evaluation of the mappings* you create in terms of their quality. Finally, HALE also contains a system that can be used to *document the validity of mappings*.

- Metadata
  ◦ Mapping rules for metadata elements
- Spatial and temporal aspects
  ◦ Definition of functions for transformation of geometric types
- Multiple representations
  ◦ Definition of rules for handling other representations of the same object, i.e. under which circumstances which precedence should be used.

*Main Workflow*

When working with HALE, you typically follow these main steps. Each of these steps is explained in greater detail in the remainder of this handbook.

1. Open the schema of the source data you want to map

2. Open the target schema

3. Open a source data set (optional)

4. Configure the source and target Styled Layer Descriptors (optional) to get interactive feedback on the progress of your mapping project in the map viewer

5. Set up initial analysis and quality definition

6. Go through the created task list until the mapping is of sufficient quality for your application

7. Start final mapping analysis (optional)

8. If no problems pop up, save/publish created mapping.

*Terminology*

The term *alignment* refers to the complete set of relationships defined between two conceptual schemas. A *mapping* is an individual relationship between entities of the source and the target schema.

A *conceptual schema* is the model used for a data set and can be expressed in a different conceptual schema languages, such as a GML Application Schema, as an XML Schema (XSD), as a UML model or as a SQL DDL. Even shapefiles contain a simple model for attributive data.

Please note that throughout this documentation, the term *FeatureType* is used synonymously to Spatial Object Type (INSPIRE wording) and Concept (Ontology Engineering).

*Attribute* and *Property* are used synonymously. Both refer to elements of a type definition, sometimes also called fields. Usually, a Property represents a complex field, i.e. one that is not of a primitive type such as a String or number.

# Installation of the HUMBOLDT Alignment Editor

**Windows (32 bit and 64 bit)**

Prerequisites for the HALE installation:

– Installed Java Runtime Environment (1.6 required) or Java Development Kit; can be downloaded from *http://www.java.com*

– Windows XP, Windows Vista or Windows 7

For Windows operating systems, an executable installer is provided. This installer contains all program files. Double-clicking the file (of type *.MSI) will start

an installation wizards guiding you through the important steps, such as selecting the location where HALE should be installed. To start the application, go to the folder where you have installed HALE and double-click on the file called hale.exe.

It should be noted that currently, you can run multiple versions of HALE in parallel.

To uninstall HALE, just delete the folder in which you have installed the software.

**Linux and MacOS**

Prerequisites for the HALE installation:

– Installed Java Runtime Environment (1.6 required) or Java Development Kit; can be downloaded from *http://www.java.com*

For Linux and MacOS, HALE distributions are delivered as a single zip file. This file just needs to be uncompressed. After this step, HALE can be started by going to it's installation folder.

**Building and running from source code**

If you would like to build HALE from the source code, you need to have the following environment set up:

– an installation of a 1.6 JDK, which can be found here:

*http://java.sun.com/javase/downloads/index.jsp*

– an installation of Eclipse for RCP Development, version 3.5, which can be found here:

*http://www.eclipse.org/downloads/download.php?file=/technology/epp/downloads/release/ganymede/SR2/eclipse-rcp-ganymede-SR2-win32.zip*

Furthermore, you need to have a Subersion client installed, such as Subversive or Subclipse for Eclipe, TortoiseSVN or any other one. When these prerequisities are fulfilled, proceed as follows to check out any HALE version later than 1.0.0-RC1:

1. First, start eclipse in a new workspace.

2. Second, check out the target platform project for HALE from this location:

*http://svn.esdi-humboldt.eu/repo/humboldt2/trunk/hale/hale-platform/*

3. After checking out, double click the hale-platform.target file and wait until eclipse has fully resolved all dependencies. This can take a while since some of the de-

pendencies have to be downloaded. When this is finished, press the button labelled "Set as target platform" in the upper right corner.

4. Now, check out the actual HALE and CST modules as projects:

   1. eu.esdihumboldt.hale: *http://svn.es-di-humboldt.eu/repo/humboldt2/trunk/hale/eu.esdihumboldt.hale/*

   2. eu.esdihumboldt.hale.gmlparser: *http://svn.esdi-humboldt.eu/repo/humboldt2/trunk/hale/eu.esdihumboldt.hale.gmlparser/*

   3. eu.esdihumboldt.hale.logback.config: *http://svn.esdi-humboldt.eu/repo/humboldt2/trunk/hale/eu.esdihumboldt.hale.logback.config/*

   4. eu.esdihumboldt.hale.rcp.wizards.-functions.core: *http://svn.esdi-humboldt.eu/repo/humboldt2/trunk/hale/eu.esdihumboldt.hale.rcp.wizards.functions.core/*

   5. eu.esdihumboldt.hale.schemaprovider: *http://svn.esdi-humboldt.eu/repo/humboldt2/trunk/hale/eu.esdihumboldt.hale.schemaprovider/*

   6. eu.esdihumboldt.hale.htmlexporter: *http://svn.esdi-humboldt.eu/repo/humboldt2/trunk/hale/eu.esdihumboldt.hale.htmlexporter/*

   7. eu.esdihumboldt.hale.instancepro-

vider: *http://svn.esdi-humboldt.eu/repo/humboldt2/trunk/hale/eu.esdihumboldt.hale.instanceprovider/*

   8. eu.esdihumboldt.gmlhandler: *http://svn.esdi-humboldt.eu/repo/humboldt2/trunk/gmlhandler/*

   9. eu.esdihumboldt.transformer.cst: *http://svn.esdi-humboldt.eu/repo/humboldt2/trunk/cst/eu.esdihumboldt.cst/*

   10. eu.esdihumboldt.cst.corefunctions: *http://svn.esdi-humboldt.eu/repo/humboldt2/trunk/cst/eu.esdihumboldt.cst.-corefunctions/*

   11. com.onespatial.jrc.tns.oml.rif.translator: *http://svn.esdi-humboldt.eu/repo/humboldt2/trunk/hale/com.onespatial.jrc.tns.oml.rif.translator /*

   12. com.onespatial.jrc.tns.rif.exporter: *http://svn.esdi-humboldt.eu/repo/humboldt2/trunk/hale/com.onespatial.jrc.tns.rif.exporter/*

In the eu.esdihumboldt.hale project, you should now be able to double-click on the HALE.product file and launch the application from there.

To create a new product based on your developments, also double-click on the HALE.product file and then click the "Export an Eclipse product" button in the upper left corner.

Please note that checking out can be done anonymously, but for checking in you first need to obtain an account, which you can get from *thorsten.reitz@igd.-fraunhofer.de*.

# Using the HUMBOLDT Alignment Editor

## Configuring your workbench

When you start HALE the first time, it will present you with a user interface that contains several different parts. These parts, called views, each have a specific purpose. In the current release, the following views are included:

have defined on the elements currently selected in the Schema Explorer.

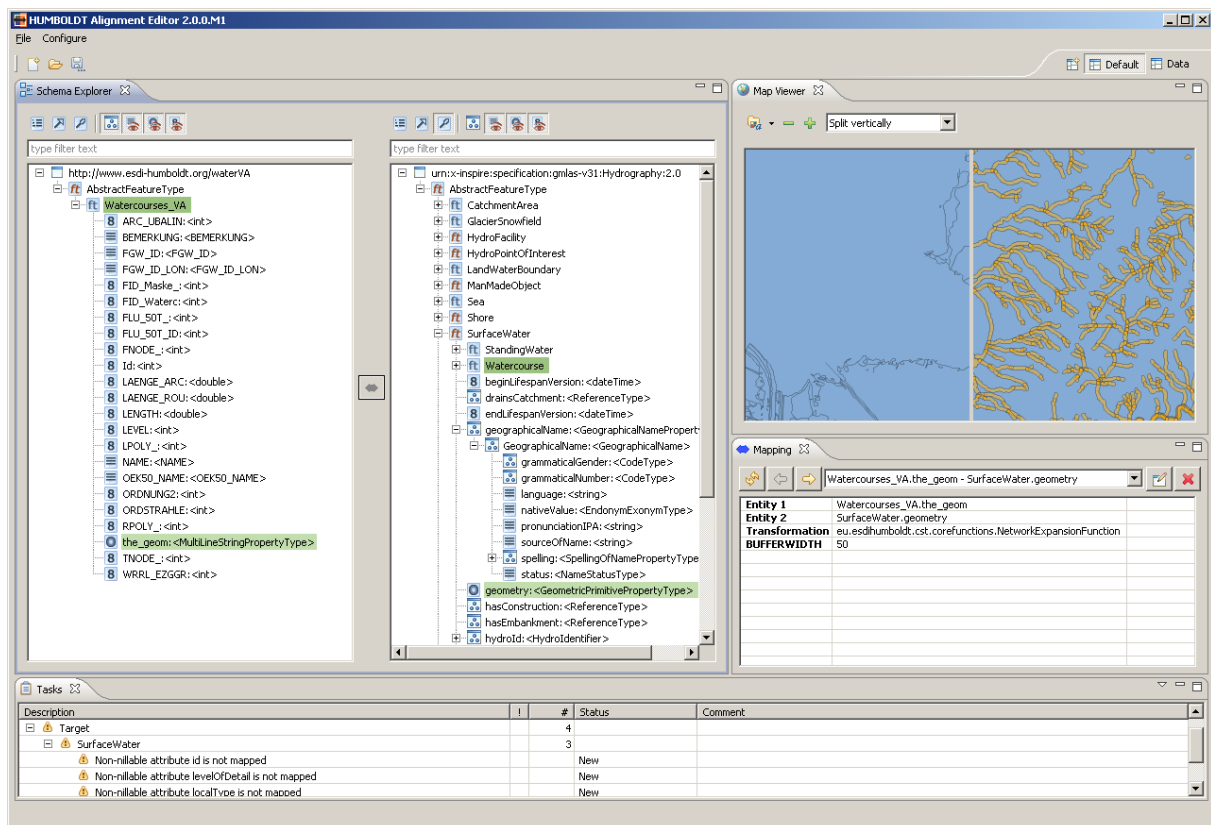– *Map View*: This view provides you with a cartographic representation of the reference



*Figure 3: The default view configuration of HALE, including the Schema Explorer, the Map Viewer, the Mapping view and the Tasks view.*

– *Schema Explorer*: The Schema Explorer allows you to view the structure of the source (left) and the target (right) schema in various ways and enables the definition of mappings between the elements of the schemas.

– *Mapping View*: The mapping view provides information on the schema mappings you

data for the source schema and the transformed data alongside each other, when you have loaded such data. It can be styled and navigated interactively.

– *Reference Data:* This view provides a tabular representation of the reference data for the source schema when you have loaded any.

– *Transformed Data*: Like the Reference Data

view, this view provides you with the possibility to inspect transformed data.

– *Tasks*: The Task View provides you with information on tasks that have to be completed to create a complete and correct mapping.

– *Mapping Graph View*: This view gives you a graphical overview of the mapping created so far.

– *Error Log:* Gives you insight into the program's log messages. They can be exported as log files to submit with a bug report

Each of these views can be resized by grabbing its border and then dragging the mouse cursor. Furthermore, every view can be maximized by double clicking on the "Folder" part of the view on top or by clicking the bigger window icon on the right side of the view's top. To set a view back to normal, you can again double-click on its tab or click on the small window icon.

It is also possible to rearrange views. Just click on

the tab and drag the mouse cursor to the place where you would like the view to be. You will see that it is possible to stack views over each other, but also to arrange them next to each other. In addition, you can also open and close views, To close a view, just click the "X" next to it's tab. To open it again, go to the "Configure" menu, pick "Views"->"Other" and then select the view you would like to see again.

When you find a view configuration that you like particularly well, you can save that configuration. Again, go to the Configure menu, pick "Save Perspective as", and enter the name under which you want to store your configuration. The configuration's name will now appear in the upper right corner to indicate it is active and stored.

By default, there are two perspectives already pre-configured for you. One is called the "Default" perspective and includes the Schema Explorer, the Map View, the Mapping View and the Task View, the other one is called the "Data" perspective and contains the Mapping View, the Map view, the Reference Data view and the Transformed Data view.

**Loading and browsing schemas**

A first step to map schemas is to load them. In the current version of HALE, you can load any XML Schema, including metadata schemas, GML Application Schemas and others. However, the schema import is optimized for GML Application Schemas and supports the following versions:

– GML 2.1

– GML 3.1

– GML 3.2

You can also load a schema from a shapefile (*.shp). Schemas can either be loaded from a file or from a

Web Feature Service's GetCapabilities and DescribeFeatureType operations. Please note that using GML Application Schemas with types extending *AbstractFeatureType* is the recommended method for HALE in the current version. Only when doing so, all transformation functions can be used.

Such schemas can be created in multiple ways, e.g. from UML models by using Shapechange[1] or from databases and shape files using Geoserver[2]. It is

---

[1] Please refer to *www.interactive-instruments.de/ugas*
[2] GeoServer can be used to create GML and Schemas from Shapefiles and Oracle databases, among other sources. Please refer to *http://geoserver.org/*.

also possible to use Safe's FME for this purpose.

To load a schema, go to the "File" menu and click "Open Schema". In the dialogue that appears now, choose whether you want to import a schema to use as a source or as a target schema. Then, choose from which source to import by clicking the radio buttons in front of the "... file" and "...WFS" options.

When you chose to import from a file, click the Button labelled "Browse..." and select the file you would like to import. Please note that only files with an extension that is either *.xsd, *.xml or *.gml will be shown. After doing so, press "Open" and then "Finish".

Importing from a WFS works in a similar way. Press the "Change..." button and enter your server's Get-Capabilities request URL into the text field at the top of the appearing window. The correctness of the URL is automatically validated. If the validation is successful, you can continue by pressing the "Finish" button. All types will now be loaded and shown in a new wizard page. If your WFS offers `Feature-Types` from more than one namespace, you are now also required to pick one of the namespaces – HALE can always just work with one "main" namespace for the schema. Press "Use this WFS" and then again "Finish".

When everything worked as it should have, you will now see the namespace of your schema in the left part of the Schema Explorer if you loaded a source schema, or in the right part if you selected to load a target schema.

*Using multiple source schemas*

HALE allows you to load exactly one source and one target schema. However, the schemas that you load are actually allowed to have multiple imported schemas. So, if you have to use multiple source schemas in your mapping, using joins between these, you can create a new schema file and import the schemas you want to join. Importing schemas is done following this pattern:

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema" xmlns:import1="http://yournamespace.com/import1"
xmlns:import2="http://yournamespace.com/import2"
 elementFormDefault="qualified" targetNamespace="http://yournamespace.com/combined">
  <import namespace="http://yournamespace.com/import1" schemaLocation="http://schemas.yourserver.com/YourSchemaImport1.xsd"/>
  <import namespace="http://yournamespace.com/import2" schemaLocation="http://schemas.yourserver.com/YourSchemaImport2.xsd"/>
</schema>
```

*Troubleshooting Schema Loading*

In case an error message pops up during schema loading or a schema doesn't appear after loading, some problem occurred in parsing it. Most of the time, these issues are due to referenced schemas not being accessible. This is e.g. the case when your schema references online resources, e.g. from *http://schemas.opengis.net*, but you currently don't have a connection. Also, you might have relative paths in the schema locations used that are not valid on your machine.

Another issue that frequently prevents loading of schemas is the presence of a **proxy server.** If your network requires the usage of one, please configure the proxy server setting by going to the Configure Menu and clicking "Network Settings". Enter your HTTP proxy host and port and click "Save settings".

*Exploring a loaded schema*

HALE comes with a powerful schema explorer view. This view presents the content of the loaded schem-

as in a tree view. In the default representation, the schema elements that have been loaded are organized by their extension (or inheritance) hierarchy. In the case of a GML Application Schema, this means that directly under the namespace element, an entry for `AbstractGmlObject` and/or `AbstractFeatureType` is created, depending on the GML version you are using. Below this entry, you'll find your types extending `AbstractFeatureType`.

If your schema contains complex element declarations that do not extend *AbstractFeatureType*, these are interpreted as so-called *PropertyTypes*.

When you expand a Feature or Property Type by clicking on the + or arrow (depending on your operating system) in front of its name, its subtypes and attributes will be shown to you. For the attributes additionally to the property name also their type name is shown, e.g. *<string>*. A type name that includes a question mark (e.g. *<?>* or *<? extends boolean>*) states that this is an anonymous type declared especially for the given attribute.

Some attributes represented in the schema tree are marked with a special sign in addition to their classification symbol. The default geometry of a feature type is marked with a green triangle ( ), items that represent attributes in XML (in contrast to elements) are marked with a brown arrow (e.g. ) and mandatory attributes are marked with a red asterisk (e.g. ).

You can also deactivate the organisation by inheritance by clicking the first icon on the top of the respective schema explorer. Your types will then be shown in a list instead of a tree. Pressing the button again will restore the tree presentation.

There are two additional ways of extending the tree

or the list:

– *Expand complex properties:* In the case of complex application schemas, attributes are often complex types themselves. Enabling this option will expand these properties, so that you can inspect their structure.

– *Repeat inherited properties:* With this option enabled, all properties that a type inherits are also displayed under it again, not just under the type that declared the properties. This is a helpful option if you want to define specific mapping rules for each subtype instead of defining them on the supertype.

In addition to extending the representation, there are also several filters you can apply to limit the information presented to you. The most obvious one is the filtering by element name. Just enter a string corresponding to the name of the element you are looking for, and the tree will be updated to show only the elements matching your query. To remove this filter, press the eraser icon on the right side of the filter text field.

Furthermore, you can filter the properties that are displayed by their type:

– *Hide Abstract Types*: This will hide types that are declared as abstract.

– *Hide Property Types:* This will hide/show types that are no FeatureTypes.

– *Hide String Types*: This will hide/show any property with a String binding.

– *Hide Numeric Types:* This will hide/show any property with a numeric (integer, floating point) type of binding.

– *Hide Geometry Types*: This will hide/show any property that has a Geometry binding.

You can also get detailed information about any element of a schema by right-clicking on it in the schema explorer and then selecting "Properties". The dialogue appearing then contains any annotations that have been defined on the type, provides information on the binding and also on other properties, such as its cardinality. It also lists allowed values in case of an enumeration type of property.

**Mapping schema elements**

The main purpose of HALE is of course not to just inspect schemas, but to map the elements of these schemas with the goal of transforming corresponding geodata sets.

To map types or attributes from the source schema to type or attributes, you first have to select them in the Schema Explorer. You can do any complex selection in the Schema Explorer, such as selecting three elements on the source side and one element on the target schema side. To select multiple elements, press the CTRL button while clicking on the items. Using the CTRL button will also allow you to deselect a schema item.

After selecting the source and target schema elements, you can press the Arrow button in between Source and Target Schema and then select an appropriate function.

If you want to connect one feature type to another, use "Retype Feature", which effectively allows to assign instances of one type of the source schema to one in the target schema. In addition, you can define so-called instance splits and merges in this function.

After defining a Retype Feature mapping, you can also apply a filter to it. Without changing the selection to which you applied a transformation, click on the "Mapping" button again and choose the CQL Filter

All options above can also be combined. As an example, selecting the "List" organisation mode together with "repeat inherited properties" and "Hide Property types" will give you a list of the abstract and concrete FeatureTypes, each one with their full attribute set. Use Hide Abstract Types in addition and you'll get a list of only the concrete types, each one with all its properties.

Wizard. Configure the expression as you would like to have it, click "Finish", and you will now see a Filter added in the Mapping View.

*Available attributive transformation functions*

The functions that are available perform transformations on properties. These include the following:

1.  A simple *Attribute Rename Function* that can copy any alphanumeric attribute and that can also map the following geometric types: `LineString <-> MultiPoint`, `Polygon <-> MultiPoint`, `MultiPolygon <-> MultiPoint`, `MultiLineString <-> MultiPoint`, `Polygon <-> LineString`, `Polygon <-> MultiLineString`, `MultiPolygon <-> LineString`

2.  A *Generic Math Function* where you can use source attributes as variables in an arbitrary mathematical expression,

3.  A *Bounding Box Function* and a *Centroid Function* to derive additional geometric attributes,

4.  A *Buffer Function* that will buffer any Line or point-type geometry to become polygons,

5.  A *Classification Mapping Function* that al-

lows to transform code lists and classification values.

6. An *Ordinates to Point Geometry* function that accepts two numerical values as input and will create a point geometry on that base.

7. An *Attribute Concatenation Function* that allows to concatenate attribute values and user-defined strings;

8. A *Date Extraction* Function that allows you to extract a date object by specifying the format of that date and to set it in another format.

9. A *Create GML Reference* Function that enables you to create a reference to another element in the same data set or a different data set.

10. An *INSPIRE Identifier* Function that enables you to create a `IdentifierProperty-Type` easily;

11. An INSPIRE Geographic Name Function that does the same for `Geographic-NamePropertyTypes.`

12. And finally, a *User Defined* Function that serves as a placeholder to document required transformation functions that are not yet available in HALE or CST or within your own extensions.

There are also so-called *Augmentation* functions that will be applied only on the target feature but don't use data in source features. An example for this is the "*NilReason Function*" or the "*Attribute Default Value*" function.

*Using the Mapping View*

After you have created a mapping, also called a "Cell", the mapping can be seen and edited in the Mapping View. The Mapping View, by default, allows you to step through all mappings that you have defined so far. You can step through the mappings either using the two arrow buttons on top or by selecting the mapping you want to view or edit from the drop-down box in the middle of the Mapping view's icons bar. If you have many mappings, it might be a good idea to synchronize the content of the Mapping view with your selection in the Schema Explorer. You can do this by pressing the "Synchronize with Schema Explorer Selection" button. After doing so, the drop-down box will contain only those mappings defined on the currently selected types and their subtypes or properties.

To edit a mapping, click the "Edit Cell" button on the top right part of the view. You will then be presented with the wizard matching the function used in the mapping you want to edit.

It is also possible to delete a cell. Just click the button with the red "X", and – after your confirmation – the currently selected cell will be removed.

*Examples for typical mapping cases*

A typical case is a **one-to-one type mapping**. In this scenario, you have one type in the source schema which you would like to map to one type in the target schema. This can be done with the following steps:

1. Select the Feature Type from the source schema you would like to map

2. Select the Feature type from the target schema that shall be mapped

3. Pick the Retype Feature function

4. Optionally, define a split condition (more on

this later)

5. Per default, only the identifier and the default geometry will be transformed. If you want to transform additional attributes from the source to the target, you'll need to add mappings for those attributes now.

HALE also supports more complex mappings where **one FeatureType from the source schema is related to multiple FeatureTypes from the target schema or vice versa**. This is the process for "splitting" a Feature Type into multiple types:

1. Select the source type and the first target type you want to map.

2. Apply a Retype Feature function, as above.

3. Now, apply a filter to the Retype function that you have just defined by clicking on the mapping arrow again and by selecting any of the two "CQL-based Filter" options.

4. Define your filter on the basis of the attributes of the source type. Such a filter can e.g. pick only specific individual values out of an enumeration of values or a codelist.

5. Now, select the second target type that you want to map.

6. Again, apply a Retype Feature function.

7. Apply a corresponding CQL filter to this second Retype Feature function as well.

You can do this as many times as you want, you just should be careful that your filters are mutually exclusive, otherwise you'll end up with double Features being created in the later transformation process.

For "merging" Feature Types, the process is simpler:

1. Select the first source type and the target type you want to map.

2. Apply a Retype Feature function.

3. Select the second source type that you want to map.

4. Again, apply a Retype Feature function.

This process can also be repeated as many times as necessary.

*Renaming Attributes*

Renaming attributes is the act of copying the value of an attribute in the source schema to an attribute of one of the types in the target schema. This is likely one of the most common operations you'll need, so HALE tries to make it as simple as possible. The Rename Attribute function in HALE can convert automatically between alphanumeric types as applicable (double, integer, long integers, float, strings) as well as between geometric types.

To use the Attribute Rename function, click on the attribute in the source schema you'd like to rename, then select the element in the target schema that the attribute should be copied to.

HALE also enables you to rename attributes that are not directly attributes of the Feature Type you are mapping, but are rather nested properties. An example for such a nested property is the GeographicName property that many spatial object types have in the INSPIRE application schemas. This property is a complex property, i.e. it does not consist of a single value, but rather of a full structure of properties and sub-properties. If you now want to set such a sub-property (or nested property) directly, proceed as follows:

1. Select the source property you want to rename;

2. Select the nested property, where the value of the source property should actually be

copied;

3. Click on the mapping arrow and select the Rename Attribute Function;

4. In the dialogue shown, the field Traget Property is now filled in with a string such as `geographicName::Geographic-NameType::spelling;`

5. After confirmation, the value is copied.

6. Optionally, you can apply a filter on the transformation.

*Reclassification*

One of the most important transformation functions that HALE offers is the **Classification Mapping** function. This function allows you to map values of a source attribute to fixed values of a target attribute. The relation is always a many-to-one relation, and each code from the source schema can only be mapped once.

If the schema you have loaded contain information on the allowable values, this information will be used by the Classification mapping Wizard to pre-fill the target values list (see Figure 4).
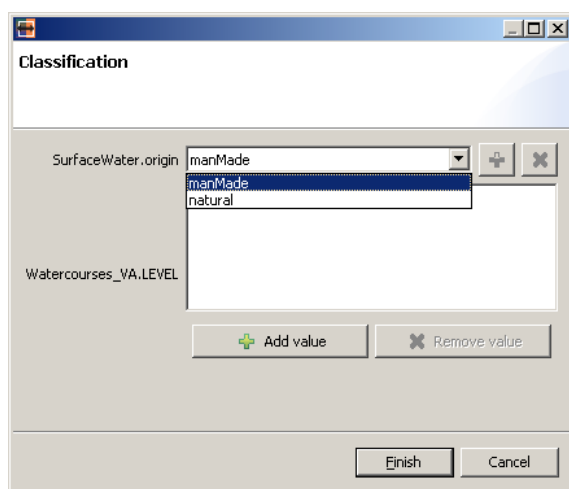


*Figure 4: The Reclassification wizard with pre-filled in target values from the schema*

*Using Code Lists*

In HALE, you can also load Code Lists and use them appropriate, e.g. in the *Classification Mapping* or in the *Attribute Default Value* function.

To load a code list, go to the Configure Menu and pick "Other settings...". In the dialogue that opens, select "Code lists". There you can add folders which contain your code list files. After adding your folder, press OK to save your settings and to close the dialogue.

To use a Code List, create a mapping where at least one of the source and target attributes is a Code Type. Currently, exactly two functions make use of code lists, the aforementioned *Classification Mapping* and the *Attribute Default Value* function. In both, you can now assign a code list to be used. Actually, HALE will try to find a matching one itself, but this does not always succeed.
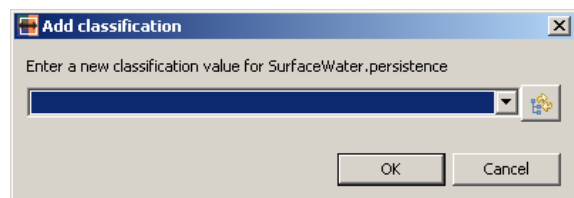


*Figure 5: Adding a classification value using a Code List*

As shown in Figure 5, there is a button to the right of the classification combo box. Pressing this button gives you the opportunity to change the preselected codelist or to initially assign one should HALE not have been successful in automatically determining one.

In the Assign Code Lists dialogue, you have two options: You can either pick one of the code lists found on the search path (using the "from search path" tab) or you can directly load a file containing the definition (using the "from file or URL" tab). Pick the code list you want to use, and you can now use the code list values as you would use them when the schema directly contained enumerations.
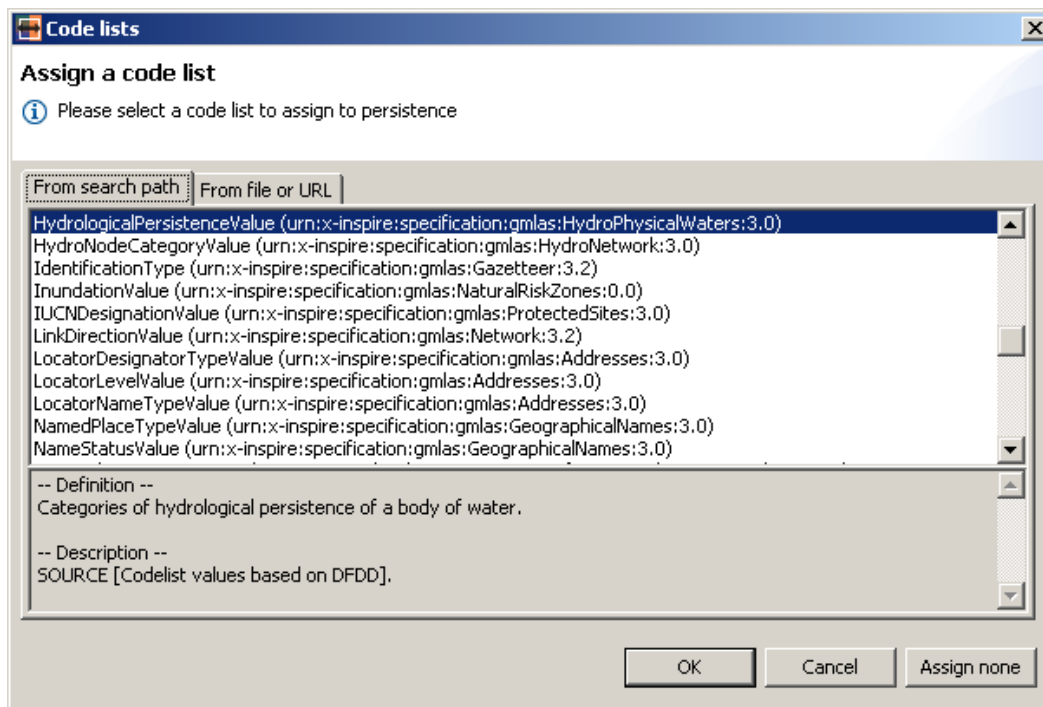


*Figure 6: The dialogue for selecting a code list to use*
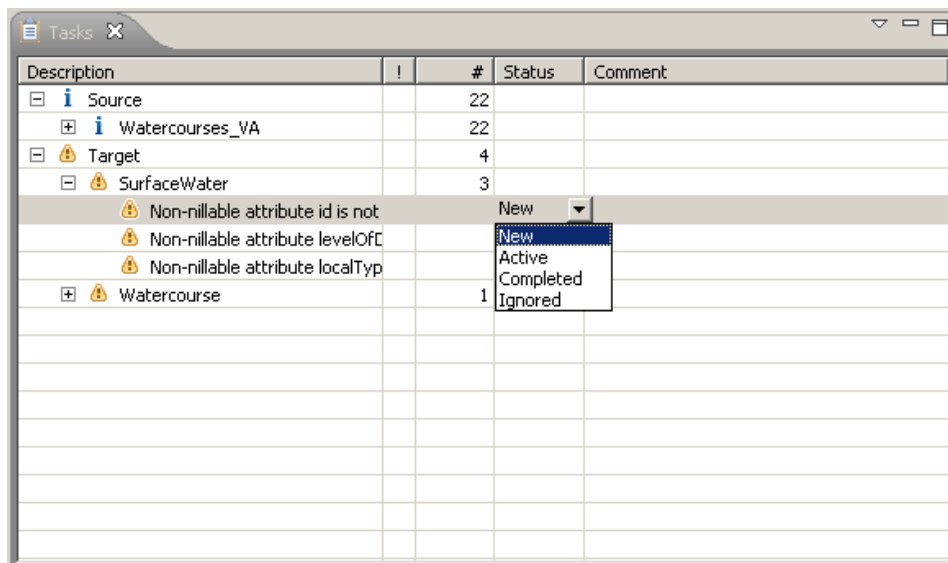
**Working with tasks**



*Figure 7: The task view of HALE*

Mapping two complex schemas can be quite a daunting task that requires the creation of many dozens of individual mappings. It is hard to evaluate one's progress and can even be frustrating. To make this process more transparent and efficient for you, HALE provides you with the option of working in a task-based process. For this purpose, HALE continuously evaluates the current mappings and creates "tasks", "warnings" and "errors" on the basis of this evaluation. These different work items are displayed in the Task View and mean different things:

– *Task*: A normal work item that suggests what to do, e.g. to map any given element of the source schema or the target schema. A normal task is visually indicated by the blue "i" icon in front of it's description.

– *Warning*: There is likely an issue with the mapping that needs to be amended, such as a non-nillable attribute not yet having a mapping assigned for it. A warning is visually identified by the yellow warning triangle icon in front of its description.

– *Error*: The mapping you created is not valid, e.g. it overwrites values you have already set or contains consistence errors. An Error is visually identified by the red circle with a white X in it.

Each task's properties are given in one line of the table in the task view. The first column indicates the type of task and gives a short description. The second column, headed by an exclamation mark (!), gives an indication of the value of each task. Tasks with a higher-than-average value are marked with a red exclamation mark. The next column provides you with information on the number of total tasks, warnings and errors under a given item. The Status column indicates the current status of the task (more on this below). The final column displays any comments you have added to the specific task.

A task can be activated by double-clicking it. The schema items that are connected to this task are then automatically selected to help you to quickly resolve the task.

You can also change the status of a task. This is

done by double-clicking on the task's status in the status column of the task view. You can set the status to any of the following:

– *New*: A new task on which no work has been done yet.

– *Active*: A task on which you are currently working.

– *Completed*: You have completed work on this task.

– *Ignored*: You know that this task is not important and therefore you can ignore it.

It is also possible to add comments to each task, such as limitations of the mapping you created for that task or how far you have come with solving the task. Adding a comment is simply done by double-clicking in the comments column in the row of the task you want to add a comment to.

*Configuring Task Creation*

You can also configure when HALE should create tasks for you. Go to the "Configure" menu and click Tasks. You will then have the option to activate or deactivate individual task generators, Currently, there are three task providers delivered with HALE:

– *Source Schema Mapping*: This task generator creates a task for every type of the source schema, and adds tasks for its attributes when it is mapped.

– *Target Schema Mapping*: This task generator creates a task for every type of the target schema, and adds tasks for its attributes when it is mapped.

– *Nil Attribute Warnings*: This task generator creates warning when a non-nillable attribute in a target type doesn't have a mapping defined on it.
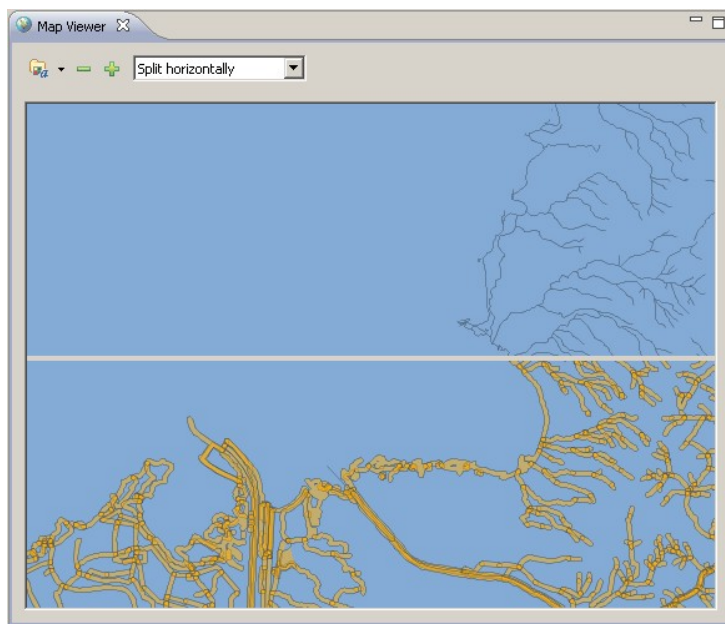
**Working with a source data set**



*Figure 8: HALE's Map Viewer, with source data shown on top and transformed data on bottom*

The term "source data" or "instance data" always refers to an actual geographic data set whose entries represent objects and are modelled using one of the loaded schemas, be it target or source.

To load an source data set , go to "File" -> "Open Source Data". In the wizard which opens, you can again choose whether to import data from a WFS or from a local GML file (*.shp support will be added soon). If you would like to load from a local GML file, click on "... file" and press the "Browse..." button. Select the file you want to use, press "Open" and then "Finish". Loading the file usually takes several

seconds (typically around 3-6 per Megabyte).

There are different parser configurations available for loading GML data. Select the appropriate configuration (GML 2, GML 3 or GML 3.2) for your data set.

To import source data from a WFS, pick the option "WFS GetFeature". Enter the GetCapabilities URL of your WFS in the entry field at the top of the form that appears. When HALE succeeds in parsing the capabilities, the "Next" button is enabled. Press the "Next" button and pick a namespace as well as the FeatureType of the Features that you would like to import. Note that the namespace of the loaded source schema must match the namespace of the loaded source data, otherwise you will not be able to continue. Finally, you can also define filters on the source data, such as spatial filters or filters on alpha-numerical properties. The exact filters you can define depend on the capabilities of the WFS you are working with.

If the system cannot clearly identify the used SRS from the data, it will request the user to provide either the EPSG code or the Well-Known Text for the used SRS.

### Troubleshooting source data loading

If you have problems loading GML data, try using another parser configuration. If you get an error message that complains about a Spatial Reference System that is not known to the application, use the *Additional CRS* preference page (available through *Configure→Other settings*) to register the Well Known Text of the missing reference system.

HALE currently supports loading of GML 2.1 and GML 3.1/3.2 source data. Furthermore, GML which uses Application Schemas that redefine core elements such as `FeatureCollection` also often

give issues in loading. If you have issues with loading a GML file, please provide us with feedback on the issue.

Also please note that many existing WFS implementations have a few peculiarities. HALE has been tested with the following products:

– GeoServer – OK

– MapServer – OK

– CubeWerx WFS – OK

– ESRI – OK when Schema References are repaired (by default they are often broken

– GeoMedia – Fails because of coordinate formatting issues and redefinition of FeatureCollection

### The Map Viewer

After this step, the content of the file will now be displayed in the Map Viewer. In the Map Viewer, you can pan and zoom in this map view using the mouse and mouse wheel or the buttons at the top of the view. Furthermore, you can select one of several overlay modes between the reference data and the transformed data, including horizontal and vertical splitting, diagonal splitting and a direct overlay.

Left-Clicking in the map enables you to pick features. This selection can be synchronized with the source data and the transformed data views.

### Styling of your data

On the top left corner of the Map View you can find the settings for controlling the map rendering. Specifically, you can configure so-called Styled Layer Descriptors there, using a rich GUI. Click on the Styles button, select the `FeatureType` for which you want to define a styling, and then configure your styling rule. You can define styles for points, lines

and polygons, and you can define complex styles with filter rules. It's also possible to define a style for selected Features.

It is also possible to load a styling from a XML Styled layer Descriptor file. Using this feature to load a pre-defined styling for the transformed data allows to get a live impression in the Map Viewer of the progress of your mapping endeavour.

*The Data Views*

Another option for analysing the source data and the transformed data is using the *"Source Data"* and *"Transformed Data"* views.

They display the attribute values of sample features that are picked from the source data. The features in the "Transformed Data" view are already trans-formed using the alignment mapping. To select

sample features of a certain type you have to select the corresponding feature type in the list. Additionally you can define a filter in the text field. The buttons next to the filter field on the right provide support for defining your filter (Inserting attribute names/Create filter using a form).

To analyse features and their transformed counter-parts you can synchronize both views by clicking on the little table icon in the "Transformed Data" View. If the views are synchronized, in the "Transformed Data" only the transformed features of the features displayed in the "Reference Data" view will be avail-able. You can filter the transformed features by their feature type.

You can also synchronize the "Transformed Data" View to the selection made in the Map View, or to just show manually filtered source features.

**Saving and loading projects and alignments**

*Exporting your mapping*

Exporting your created Mapping is very straightfor-ward: Go to the "File" menu, pick "Save Mapping" and chose a location and file name.

*Saving and Loading of Alignment Projects*

To save your entire project configuration, go to the "File" menu , and pick "Save Alignment Project as...". You have to pick a location where to save the project

to, and can optionally also provide some metadata on the project. This will create two files: A project configuration file and an alignment file with the same file name and an additional extension (*.goml).

To load a project, just go to the "File" menu, and pick "Open Alignment Project". Pick the project file and click "Finish". Schemas, the mapping, selected source instances, the SLD configuration and the state of your tasks will then be loaded.

## Point of Contact

Fraunhofer Institute for Computer Graphics Research

Thorsten Reitz

Fraunhoferstraße 5

64283 Darmstadt

Phone: +49 6151 155 416

Fax: +49 6151 155 444

Email: thorsten.reitz@igd.fraunhofer.de

http://www.igd.fraunhofer.de