

# ABuDaT - Technical System Documentation

---

EIS / SemWeb Lab 2016

Topic: Semantic Lifting of Budget Data

## Mentor

Fathoni Musyaffa, [musyaffa@iai.uni-bonn.de](mailto:musyaffa@iai.uni-bonn.de)

## Team members

- Florian Weile, [weile@cs.uni-bonn.de](mailto:weile@cs.uni-bonn.de) (team leader)
- Tatiana Novikova, [s6tanovi@uni-bonn.de](mailto:s6tanovi@uni-bonn.de)
- Aberham Gebreyohannes, [s6abgebr@uni-bonn.de](mailto:s6abgebr@uni-bonn.de)
- Samuel Y. Ayele, [ayele@informatik.uni-bonn.de](mailto:ayele@informatik.uni-bonn.de)

---

## Table of Contents

<b>1. Introduction</b>	<b>3</b>
<b>2. Purpose</b>	<b>3</b>
2.1 Limitations	3
<b>3. System Overview</b>	<b>4</b>
3.1 User Input / Form Validation	4
3.2 RDF Data-Cube Validation	4
<b>4. System Architecture</b>	<b>5</b>
4.1 Major Components	5
4.2 Used Technology	5
<b>5. Installation Guide</b>	<b>6</b>
5.1 Prerequisites	6
5.2 ABuDaT Installation	6
5.3 Database Setup	7
5.4 Apache Jena Fuseki	8
5.5 LinkedPipes ETL	8
<b>6. ABuDaT Testing Strategy</b>	<b>11</b>
6.1 Unit Tests	11
6.2 Integration Tests	11
6.3 Frontend Tests	11
6.4 Test Coverage Report	12
<b>7. Updating the LinkedPipes ETL dependencies</b>	<b>13</b>

---

# 1. Introduction

This documentation document has been created to outline the system design for *ABuDaT* - *Administrative Budget Data Transformer*. *ABuDaT* is intended to help users, without experience in the fields of ETL, RDF and SPARQL, to transform fiscal data into the [OpenBudgets.eu](https://openbudgets.eu) data model.

## 2. Purpose

The purpose of this technical system documentation is to provide a description of how *ABuDaT* is constructed - its system architecture and database design. Furthermore it provides insight about *ABuDaT*'s testing strategy, specifies requirements to the production environment and provides an installation guide.

### 2.1 Limitations

#### Limitations on the input datasets

Due to the heterogenous nature of fiscal data some limitations on the input data format are made. At the time of writing this documentation, *ABuDaT* supports datasets in CSV and XLS format. The data may contain a header row that is analyzed by *ABuDaT* in order to make the transformation workflow easier for the user.

However a user might prepare the fiscal data he want's to transform prior to using *ABuDaT*.

Examples of valid input data can be found on the github repository of the OpenBudgets.eu project. For example for the ESIF and Aragon datasets:

- <https://github.com/openbudgets/datasets/tree/master/ESIF/2014/raw>
- <https://github.com/openbudgets/datasets/tree/master/Aragon/2016/raw>

#### Limitations on the data validation

For fiscal data in the OpenBudgets.eu data format, there are two possible validations that can be performed. On the one hand, there is the RDF data cube validation, on the other

---

hand there is the OpenBudgets.eu data validation. At the time of writing this documentation, ABuDaT only supports the RDF data cube validation.

### 3. System Overview

Transforming fiscal data to the OpenBudgets.eu data model is a challenging task. For citizens without knowledge in RDF and SPARQL, in the process extract - transform - load, it might even be close to impossible to do so. ABuDaT was introduced to help transforming fiscal data by providing an easy-to-use interface.

ABuDaT is designed as a web application and is thus platform independent. In the backend ABuDaT utilizes existing ETL software to transform datasets. It uses a database to store created transformation definitions and the executions.

Besides transformation of fiscal data, ABuDaT enables its users to verify that transformation results meet the integrity constraints of the RDF Data Cube Vocabulary specified in <https://www.w3.org/TR/vocab-data-cube/>.

Transformed data can be uploaded to an RDF triple store.

#### 3.1 User Input / Form Validation

In order to prevent users from transforming erroneous input data. ABuDaT uses html form validation to check if all relevant data is provided and if the data has the correct form. For example, ABuDaT validates that the value of a URI field does really contain a valid URI. If errors are found in the input fields, the user is redirected to the form and the fields containing errors are highlighted. A error message is presented to the user giving hints how to fix them. ABuDaT is Spring® powered for the purpose of form validation: <http://docs.spring.io/spring/docs/current/spring-framework-reference/html/validation.html#validation-mvc>

#### 3.2 RDF Data-Cube Validation

ABuDaT uses a modified version of the NoSPA RDF data cube validator <https://github.com/yyz1989/NoSPA-RDF-Data-Cube-Validator>. The original

---

implementation is pure logger-based. As this is not suitable for ABuDaT a modified version that outputs the validation report as a Java String was introduced.

## 4. System Architecture

ABuDaT is designed as a web application. It consists of a frontend and a backend part. The system architecture is described in the following sections.

### 4.1 Major Components

- The frontend, accessed using a web browser, provides ways for the user to input the relevant information about the budget data to transform.
- The database is a central repository for all transformation definitions and the executions of the respective transformations.
- The data cube validator, validates transformed budget data against the integrity constraints IC\_1 to IC\_21 defined in <https://www.w3.org/TR/vocab-data-cube/>.

### 4.2 Used Technology

This section introduces the technology stack used in ABuDaT.

#### Spring® (Boot)

ABuDaT makes heavy use of the Java application framework **Spring** and it's inversion of control principles as well as it's model-view-controller and testing support functionality. Using Spring Boot, Spring's convention-over-configuration solution for creating stand-alone applications, ABuDaT comes with an integrated Apache Tomcat web server.

#### Gradle®

ABuDaT relies on the build tool **Gradle**. Spring boot comes with pre-defined Gradle tasks for deploying an application to the integrated Apache Tomcat web server and starting the server. Gradle offers a nice dependency management and is configured in a single build file.

---

## Jacoco

To keep track of the quality of ABuDaT's tests, we rely on **Jacoco** to measure the code coverage.

## Junit & Mockito

ABuDaT uses **Junit** as a unit test framework. It is also used for integration test. For-non integration tests we use **Mockito** to mock functionality of other units.

## Hibernate

The object-relational mapping framewok **Hibernate** makes it easy for ABuDaT to persist entities to a mysql database through the java persisitence API, JPA.

## Thymeleaf

At the view layer of MVC, ABuDaT uses the HTML templating engine **Thymeleaf**.

## JavaScript & jQuery

For a better user experience, we rely on JavaScript and jQuery.

# 5. Installation Guide

## 5.1 Prerequisites

- \* Have git installed.
- \* Have a Oracle JDK version 8 installed.
- \* Have mysql installed.
- \* Have apache jena Fuseki installed.

## 5.2 ABuDaT Installation

### Clone the sources

```
$ git clone https://cowclaw@bitbucket.org/cowclaw/semweblab2016.git
```

### Running ABuDaT standalone (Tomcat included)

```
$ cd ABuDaT
$ ./gradlew bootRun
```

---

ABuDaT should now be up and running on `http://localhost:9000/`

### ABuDaT configuration file

The configuration file is located under `ABuDaT/src/main/resources/application.properties*`

Default values are:

```
linkedpipes.etl.host=localhost
linkedpipes.etl.port=8080

abudat.output-dir=/tmp/abudat

fuseki.data.endpoint=http://localhost:3030/ds/data

server.address=localhost
server.port=${port:9000}
spring.thymeleaf.cache=false

spring.jpa.database=mysql
spring.datasource.url=
jdbc:mysql://localhost/abudatdata?autoReconnect=true&useSSL=false
spring.datasource.username=abudat
spring.datasource.password=abudat
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.jpa.hibernate.ddl-auto = update
spring.jpa.properties.hibernate.dialect =
org.hibernate.dialect.MySQL5Dialect
```

**Note:** Due to limitations and security concerns, linkedpipes etl needs to run on the same host as abudat. Otherwise the functionality for download of data and upload of data to a triple store will not work.

### Running in a dedicated Tomcat

It is possible to run ABuDaT in a dedicated tomcat server. Therefore simply assemble a war file:

```
$ cd semweblab2016/ABuDaT
$ ./gradlew war
```

You can find it the war file at `ABuDaT/build/libs/ABuDaT.war`. It can then be copied to Tomcat's `webapp` folder.

**Note:** In this case the `server.port=${port:9000}` setting will have no effect.

## 5.3 Database Setup

Install mysql-server

---

```
$ sudo apt-get install mysql-server
```

## Create the database

```
$ mysql -u root -p

mysql> create database abudatdata default character set utf8 default
collate utf8_bin;
mysql> create user 'abudat'@'localhost' identified by 'abudat';
mysql> grant all on abudatdata.* to 'abudat'@'localhost';
mysql> flush privileges;
mysql> quit;
```

## Troubleshooting

If during container startup, you get exceptions regarding timezone, please check:

```
$ mysql_tzinfo_to_sql /usr/share/zoneinfo | mysql -u root mysql -p
```

Then add the default time zone to `/etc/mysql/mysql.conf.d/mysqld.cnf`, e.g.:

```
[mysqld]
...
default-time-zone='Europe/Berlin'
...
```

see: <http://dev.mysql.com/doc/refman/5.7/en/time-zone-support.html>

see: <http://stackoverflow.com/a/32736024/4098376>

## 5.4 Apache Jena Fuseki

This section explains how to install Apache Jena Fuseki on Ubuntu Linux.

Download Apache Jena Fuseki 2 from <https://jena.apache.org/download/>.

Unpack the downloaded archive. On linux, cd into the unpacked directory. Then run:

```
$ chmod +x fuseki-server
```

to make the server executable. To start the server run:

```
$ ./fuseki-server --update
```

To configure Fuseki, e.g. to create datasets visit: <http://localhost:3030>.

For details see <https://jena.apache.org/documentation/fuseki2/#getting-started-with-fuseki>



---

## 5.5 LinkedPipes ETL

This section explains how to install LinkedPipes ETL on Ubuntu Linux.

### Prerequisites

- \* Have maven installed
- \* Have nodejs-legacy, npm and nodejs installed

Create a Linkedpipes ETL data and working dir root and make the owner the current user:

```
$ sudo mkdir /usr/local/linkedpipes_etl
$ sudo mkdir /usr/local/linkedpipes_etl/working
$ sudo mkdir /usr/local/linkedpipes_etl/pipelines
$ sudo chown -R $USER:$USER /usr/local/linkedpipes_etl
```

Clone the sources:

```
$ cd /usr/local/linkedpipes_etl
$ git clone https://github.com/linkedpipes/etl.git
```

### Deploy LinkedPipes

```
$ cd /usr/local/linkedpipes_etl/etl
$ mvn install
```

Create a configuration file:

```
$ cd deploy
$ touch configuration.properties
```

Create a *configuration.properties* under */usr/local/linkedpipes\_etl/etl/deploy*. It may look like this:

```
executor.webserver.port = 8085
executor.webserver.uri = http://localhost:8085

executor.execution.working_directory =
/usr/local/linkedpipes_etl/working
executor.execution.uriPrefix =
http://localhost:8080/resources/executions/

executor.log.directory = /var/log
executor.log.core.level = DEBUG

executor.osgi.lib.directory = /usr/local/linkedpipes_etl/etl/deploy/osgi
executor.osgi.working.directory = .felix/

executor-monitor.webserver.port = 8081
executor-monitor.webserver.uri = http://localhost:8081/api/v1/
executor-monitor.log.directory = /var/log
executor-monitor.log.core.level = DEBUG
```

---

```
executor-monitor.ftp.command_port = 2221
executor-monitor.ftp.data_ports_interval.start = 2222
executor-monitor.ftp.data_ports_interval.end = 2225
executor-monitor.ftp.uri = ftp://localhost:2221

frontend.webserver.port = 8080

storage.components.directory =
/usr/local/linkedpipes_etl/etl/deploy/components
storage.components.path.prefix = file://
storage.pipelines.directory = /usr/local/linkedpipes_etl/pipelines

domain.uri = http://localhost:8080

external.fuseki.path =
external.working =
external.port.start = 3300
external.port.end = 3400
```

## LinkedPipes ETL Start / Stop script

Save this bash script as `/usr/local/bin/linkedpipes_etl.sh`

```
#!/bin/bash

linkedpipes_etl_path="/usr/local/linkedpipes_etl/etl/deploy"
usage="Usage: linkedpipes_etl.sh [start|stop]"

if [ $# -eq 0 ]
then
    echo "No arguments given, "$usage
    exit
fi

if [ $1 == "start" ]
then
    cd $linkedpipes_etl_path

    echo Running executor
    ./executor.sh >> /tmp/lp_executor.log &

    echo Running executor-monitor
    ./executor-monitor.sh >> /tmp/lp_executor-monitor.log &

    echo Running frontend

    ./frontend.sh >> /tmp/lp_frontend.log &
elif [ $1 == "stop" ]
then
    echo Killing Executor
    kill `ps ax | grep /executor.jar | grep -v grep | awk '{print $1}'`

    echo Killing Executor-monitor
```

---

```
kill `ps ax | grep /executor-monitor.jar | grep -v grep | awk '{print $1}'`  
  
echo Killing Executor-view  
kill `ps ax | grep node | grep -v grep | awk '{print $1}'`  
else  
echo "Unknown argument \"$1\" \"$usage"  
fi
```

Make it executable

```
$ sudo chmod +x /usr/local/bin/linkedpipes_etl.sh
```

Now you can start LinkedPipes ETL with:

```
$ linkedpipes_etl.sh start
```

To stop it run:

```
$ linkedpipes_etl.sh stop
```

## 6. ABuDaT Testing Strategy

This section explains the testing strategy for ABuDaT.

### 6.1 Unit Tests

ABuDaT uses JUnit as a unit testing framework. The tests are located at *ABuDaT/src/test/*. ABuDaT uses validation file based testing whenever appropriate. The benefit over assert-based tests is that they are more meaningful and that they provide quick insight in what is why a test breaks.

#### Testing MVC

ABuDaT uses Spring's WebMVC testing support. Tests ensuring the correct functionality of the controllers are in: *ABuDaT/src/test/java/de/uni/bonn/iai/eis/web/controller*

### 6.2 Integration Tests

ABuDat includes a number of Tests to ensure correct integration with LinkedPipes, Fuseki and MySQL. The tests are located at *ABuDaT/src/integration-test/*.

### 6.3 Frontend Tests

ABuDaT relies on Selenium frontend tests. Some assumptions are made:

- \* Assumes Firefox Browser Already installed,
- \* Assumes ABuDat running on <http://localhost:9000/>

---

\* Assumes Linkedpipes\_etl running on http://localhost:8080/

## **Install Selenium IDE**

1. Using Firefox, download Latest Version of Selenium IDE Here <https://addons.mozilla.org/en-US/firefox/addon/selenium-ide/>
2. Select Install Now. Firefox Add-ons window pops up, when the download is complete then
3. Restart Firefox

## **Open Selenium IDE**

To run the Selenium-IDE, simply select it from the Firefox Tools menu

## **Running Tests:**

\* File menu -> open a Test Case, from ABuDaT/ABuDaT\_Test/

\* Selenium will start testing

## **6.4 Test Coverage Report**












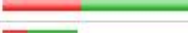























Using Jacoco ABuDaT can create reports for Test coverage. To create a report enter the ABuDaT folder and execute:

```
$ ./gradlew coverageReport
```

This will create a test coverage report for the java part of ABuDaT. Included is unit test as well as integration test coverage.

At the time of writing this documentation, ABuDaT has a combined test coverage of 92%.

## ABuDaT

Element	Missed Instructions	Cov.	Missed Branches	Cov.
 <a href="#">de.uni.bonn.iai.eis.etl.linkedpipes</a>		89%		69%
 <a href="#">de.uni.bonn.iai.eis.web</a>		86%		65%
 <a href="#">de.uni.bonn.iai.eis.web.controller</a>		88%		42%
 <a href="#">de.uni.bonn.iai.eis.web.model</a>		94%		66%
 <a href="#">de.uni.bonn.iai.eis</a>		90%		66%
 <a href="#">de.uni.bonn.iai.eis.rdf</a>		84%		n/a
 <a href="#">de.uni.bonn.iai.eis.rdf.obeu</a>		92%		n/a
 <a href="#">de.uni.bonn.iai.eis.etl.component</a>		99%		100%
 <a href="#">de.uni.bonn.iai.eis.web.model.example</a>		100%		n/a
 <a href="#">de.uni.bonn.iai.eis.etl</a>		100%		100%
 <a href="#">de.uni.bonn.iai.eis.web.model.mapping</a>		100%		75%
 <a href="#">de.uni.bonn.iai.eis.web.controller.example</a>		100%		n/a
 <a href="#">de.uni.bonn.iai.eis.rdf.obeu.classification</a>		100%		n/a
 <a href="#">de.uni.bonn.iai.eis.rdf.obeu.measure</a>		100%		n/a
Total	783 of 9.438	92%	126 of 362	65%

## 7. Updating the LinkedPipes ETL dependencies

As ABuDaT re-uses some functionality exposed in libraries provided by LinkedPipes ETL, which are not deployed anywhere on the internet. These dependencies should be updated when a new LinkedPipes ETL version was deployed.

To update the dependencies, enter the LinkedPipes ETL directory, deploy it as explained in section 5. Then from the etl base directory run the following script:

```
#!/bin/bash

mkdir dependencies

cp etl/api-component-v1/target/api-component-v1-*.jar dependencies
cp etl/api-commons/target/api-commons-*.jar dependencies
cp etl/api-component-v1-impl/target/api-component-v1-impl-*.jar \
dependencies
cp etl/api-executor-v1/target/api-executor-v1-*.jar dependencies
cp etl/plugins/e-httpGetFile/target/e-httpGetFile-*.jar dependencies
```

---

```
cp etl/plugins/e-textHolder/target/e-textHolder-*.jar dependencies
cp etl/plugins/l-filesToLocal/target/l-filesToLocal-*.jar dependencies
cp etl/plugins/t-filesToRdf/target/t-filesToRdf-*.jar dependencies
cp etl/plugins/t-graphMerger/target/t-graphMerger-*.jar dependencies
cp etl/plugins/t-rdfToFile/target/t-rdfToFile-*.jar dependencies
cp etl/plugins/t-singleGraphUnion/target/t-singleGraphUnion-*.jar \
dependencies
cp etl/plugins/t-sparqlUpdate/target/t-sparqlUpdate-*.jar dependencies
cp etl/plugins/t-tabularUv/target/t-tabularUv-*.jar dependencies
cp etl/dataunit-system/target/dataunit-system-*.jar dependencies
cp etl/dataunit-sesame/target/dataunit-sesame-*.jar dependencies
```

This will create a new directory *dependencies* and copy all relevant JAR's to it. Now copy all the JAR's from the *dependencies* directory to */ABuDaT/lib/* directory.