ELI Pillar4 **Helper specifications**

Version : 1

Date : 31/03/2022

**Table des matières**

[​ Introduction 1](#__RefHeading___Toc1661_1262260857)

[​ Prerequisites 2](#__RefHeading___Toc1663_1262260857)

[​ Functional specifications 2](#__RefHeading___Toc1665_1262260857)

[​ Atom generation from sitemap 2](#__RefHeading___Toc1667_1262260857)

[​ Command 2](#__RefHeading___Toc1669_1262260857)

[​ Output 3](#__RefHeading___Toc1671_1262260857)

[​ Sitemap + Atom generation from CSV 3](#__RefHeading___Toc1673_1262260857)

[​ Command 3](#__RefHeading___Toc1675_1262260857)

[​ Output 4](#__RefHeading___Toc1677_1262260857)

[​ CSV format 4](#__RefHeading___Toc1679_1262260857)

[​ Atom header generation 5](#__RefHeading___Toc1681_1262260857)

[​ Command 5](#__RefHeading___Toc1683_1262260857)

[​ Output 5](#__RefHeading___Toc1685_1262260857)

[​ Annex : query to generate the CSV of ELIs from Cellar 6](#__RefHeading___Toc1688_1262260857)

# Introduction

ELI Pillar 4 Helper is a tool that shall help ELI publishers implementing the Pillar 4 specification. The ELI 4th pillar defines a protocol to publish the exhaustive list of ELI URIs in a sitemap file combined with the latest update on ELI URIs in an Atom file.

The Pillar 4 helper application allows ELI publishers to either :

* generate a Pillar-4-conformant Atom feed from a sitemap file
* or generate both a sitemap and an Atom feed from a tabular CSV file

The intended usage of the tool is through automatically-scheduled tasks.

# Prerequisites

ELI Pillar 4 Helper is a command-line Java application. It requires a Java Runtime Environment, version 8 or above.

# Functional specifications

## Atom generation from sitemap

### Command

The Atom generation process shall be run the following way :

java jar pillar4helper-app.jar sitemap2atom [options]

With the following possible options :

Mandatory parameters :

* --sitemapInput :
  + Path to an input sitemap file or directory containing a set of sitemap files
* --sitemapBaseUrl :
  + base URL that is the beginning of all ELIs in the input CSV file. All ELIs must start with the same base URL
* --atomOutput :
  + path to the file where the Atom feed will be written
* --atomHeader :
  + Path to an input base Atom file containing a header information to be used for Atom feed generation (see below)

Other optional parameters :

* --atomDays
  + Number of days that the Atom feed should contain. Defaults to 60.

### Output

The sitemap content is parsed, entries of less than 60 days (or another number of days if specified) are extracted, and inserted into the Atom feed with the provided header information. The ELI URI is used as the title to insert in the feed.

## Sitemap + Atom generation from CSV

### Command

The sitemap+Atom generation process shall be run the following way :

java jar pillar4helper-app.jar csv2pillar4 [options]

With the following possible options :

Mandatory parameters :

* --input :
  + path to the input CSV file
* --sitemapOutput:
  + path to the output directory where to generate the sitemap files
* --atomOutput:
  + path to the file where the Atom feed will be written
* --sitemapBaseUrl
  + base URL that is the beginning of all ELIs in the input CSV file. All ELIs must start with the same base URL
* --atomHeader
  + Path to an input base Atom file containing a header information to be used for Atom feed generation (see below)

Other optional parameters :

* --feedUrl
  + final target URL of the Atom feed. If set, a « dct:relation » attribute will be set in the created sitemap file
* --atomDays
  + Number of days that the Atom feed should contain. Defaults to 60.

### Output

The command will :

1. Parse the input CSV file
2. Generate the sitemap in the provided output directory, and handle the splitting of sitemap files with the 50000 limit
3. Generate the Atom feed at the provided location, by inserting entries of less than the provided number of days to include. If provided, titles will be inserted in the Atom feed. Otherwise, the ELI URI will be used as title.

### CSV format

The input CSV file shall have the following structure :

"ELI","date"

"http://data.europa.eu/eli/dec/1998/538/oj",2016-05-04T11:32:59

"http://data.europa.eu/eli/reg/1976/2948/oj",2017-03-13T19:37:16

"http://data.europa.eu/eli/dec/2001/588/oj",2020-09-23T04:33:36

* the first line of the file will contain column names and will be ignored
* The rest of the file shall contain 2 columns :
  + the first column is the ELI URI
  + the second column is the update date of the ELI ;
    - it can have either the format yyyy-MM-dd or the format yyyy-MM-ddThh:mm:ss
* An optional third column can contain the title of the corresponding ELI
* Column content may use quotes as delimiters

The goal is that this CSV file can be easily generated from a query in a database (or SPARQL endpoint – see the corresponding SPARQL query to extract the CSV from OPEU Cellar SPARQL service).

## **Atom header generation**

### Command

Both the Atom feed generation command and the sitemap+Atom generation from CSV command can take as an input a base Atom file. This base Atom file is an Atom header, with no entries, containing only the header information, in which the entries will be inserted.

This third command allows to generate a basic conformant Atom header file, that can then be manually enhanced with extra information, and can be passed as a parameter to the other command.

1. The Atom header generation command shall be run the following way :
2. java jar pillar4helper-app.jar atomheader [options]

Mandatory parameters :

* --output : path to the Atom output file

Other optional parameters :

* --title : the title to insert in the Atom header. If not provided, a placeholder value will be used
* --id : the ID of the feed to insert in the Atom header. If not provided, a placeholder value will be used
* --link : the link URL of the feed to insert in the Atom header. If not provided, a placeholder value will be used
* --author : the name of the author of the feed to insert in the Atom header. If not provded, a placeholder value will be used.

### Output

The command will output a basic, but conformant, Atom header, like the following, with placeholder values :

<?xml version="1.0" encoding="UTF-8"?>

<feed xmlns="http://www.w3.org/2005/Atom">

<title>\*\*Put title here\*\*</title>

<link rel="self" type="application/atom+xml" href="\*\*Put link here\*\*" />

<author>

<name>\*\*Put author name here\*\*</name>

</author>

<id>\*\*Put Atom ID here\*\*</id>

</feed>

## **Annex : query to generate the CSV of ELIs from Cellar**

http://publications.europa.eu/webapi/rdf/sparql

prefix cdm: <http://publications.europa.eu/ontology/cdm#>

select (STR(?eli) AS ?ELI) ?date where {

?x cdm:resource\_legal\_eli ?eli .

?x <http://publications.europa.eu/ontology/cdm/cmr#lastModificationDate> ?date

}

## **Annex : query to generate the CSV of ELIs from Legilux**

https://data.legilux.public.lu/sparql

SELECT ?eli (STR(?publicationDate) AS ?publicationDateString) (STR(?titleLang) AS ?title)

WHERE {

?eli a jolux:NationalLegalResource .

?eli jolux:publicationDate ?publicationDate .

?eli jolux:isRealizedBy/jolux:title ?titleLang .

}

ORDER BY DESC(?publicationDate)

Note :

* Not all ELIs in this list correspond to webpages with metadata inside (Memorial do not include metadata). How to restrict to only entries that correspond to entries that have metadata in their webpages ?
* Same thing for Memorial B entries : do we need to filter them out ?
* A few entries don’t have a publicationDate