ELI Pillar4 **Helper documentation**

Version : 1

Date : 18/07/2022

**Table des matières**

[Introduction 2](#_Toc1)

[Prerequisites 2](#_Toc2)

[Commands documentation 2](#_Toc3)

[Atom generation from sitemap 2](#_Toc4)

[Command 2](#_Toc5)

[Output 3](#_Toc6)

[Sitemap + Atom generation from CSV 3](#_Toc7)

[Command 3](#_Toc8)

[Output 4](#_Toc9)

[CSV format 5](#_Toc10)

[Sitemap + Atom generation from SPARQL 5](#_Toc11)

[Command 5](#_Toc12)

[SPARQL Query 6](#_Toc13)

[Querying through a proxy 6](#_Toc14)

[Atom header generation 6](#_Toc15)

[Command 6](#_Toc16)

[Output 7](#_Toc17)

[Annex : query to generate Pillar4 from Cellar 7](#_Toc18)

[Annex : query to generate Pillar4 from Legilux 8](#_Toc19)

# Introduction

ELI Pillar 4 Helper is a tool that helps 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 ; this will read the most recent sitemap entries to insert them in the Atom feed.
* or generate both a sitemap and an Atom feed from a tabular CSV file ; the CSV file contains the ELI URI, its update date, and optionaly a title.
* or generate both a sitemap and an Atom feed from a SPARQL query executed against a SPARQL endpoint ; the query must return the ELI URI, its update date, and optionaly a title.

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.

# **Commands documentation**

## Atom generation from sitemap

### Command

The Atom generation process is run the following way :

java jar pillar4helper-app.jar sitemap2atom [options]

With the following possible options :

Mandatory parameters :

* --sitemapInput or -i :
  + Path to the input Sitemap file, from which the most recent entries will be read to be inserted in the Atom feed.
* --sitemapBaseUrl or -su :
  + Base URL of every ELI listed in the sitemap file. All ELIs in the sitemap MUST begin by this base URL.
* --atomOutput or -ao :
  + Path to the file where the output Atom feed will be written
* --atomHeader or -ah :
  + Path to the input Atom 'skeleton' file containing the Atom header information; this skeleton can be generated by another command.

Other optional parameters :

--atomDays or -ad

* + The number of days to consider to include sitemap entries in the Atom feed. Entries updated since this number of days will be included. 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 in the feed.

The Atom feed output file can then be copied to the web server to make it accessible at its final URL. The Pillar IV specification recommands to use ../eli/eli-update-feed.atom

## Sitemap + Atom generation from CSV

### Command

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

java jar pillar4helper-app.jar csv2pillar4 [options]

With the following possible options :

Mandatory parameters :

--input or -i :

* + Path to the input CSV file. The CSV file MUST contain 2 columns : the ELI URI, and the update date of this ELI. The CSV file MUST have a first line containing the headers, that will be ignored at parsing. The date MUST use the format yyyy-MM-dd or yyyy-MM-dd'T'hh:mm:ss. An optional third column can contain a title to be included in the Atom feed.

--sitemapOutput or -so :

* + Path to the output directory where sitemap files will be generated

--atomOutput or -ao :

* + Path to the output file where the Atom feed will be generated

--sitemapBaseUrl or -su :

* + Base URL of ELIs that will be listed in the sitemap file. All ELIs in the sitemap MUST begin by this base URL.

--atomHeader or -ah

* + Path to the input Atom 'skeleton' file containing the Atom header information; this skeleton can be generated by another command.

Other optional parameters :

--atomUrl or -au :

* + Target Atom Feed URL. If set, a dct:relation attribute will be inserted in the sitemap file to this URL

--atomDays or -ad :

* + The number of days to consider to include sitemap entries in the Atom feed. Entries updated since this number of days will be included. 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.
   1. If there is less than 50000 entries, a single output file will be generated in the output directory, with name sitemap.xml.
   2. If there is more than 50000 entries, multiple output files will be generated with name sitemapX.xml (sitemap1.xml, sitemap2.xml, etc.), and a single sitemap index will be generated with name sitemap.xml.
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.

The sitemap and Atom feed can be copied to the web server at their final target location. The Pillar IV specification recommands to use ../eli/sitemap.xml.

### 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.

## Sitemap + Atom generation from **SPARQL**

### Command

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

java jar pillar4helper-app.jar sparql2pillar4 [options]

Parameters :

The parameters are the same as for the csv2pillar4 command (--sitemapOutput, --atomOutput, --sitemapBaseUrl, --atomHeader, etc.), with the exception of –input that is replaced by the following 2 parameters

--query or -q :

* + Path to the file containing a SPARQL query to execute against the specified SPARQL endpoint. The file must be a valid SPARQL query, withan extension typically ending in '.rq'. The query MUST return 2 columns with the following name : ?eli must contain the IRI of a LegalResource, and ?updateDate contains the update date of the resource in xsd:date or xsd:dateTime datatype. An optional column '?title' can contain a title to be included in the Atom feed.

--endpoint or -e :

* + URL of the SPARQL endpoint to execute the query.

### **SPARQL Query**

The SPARQL query MUST return variables named ?eli and ?updateDate, and optionaly ?title. See the corresponding annexes for examples of SPARQL queries.

### **Querying through a proxy**

If the command is to be executed behind a proxy, you can provide the proxy parameters through the standard Java options http.proxyHost , http.proxyPort, http.proxyUser and http.proxyPassword. These options are passed as system properties of the Java Virtual Machine, with the « -D » prefix. This is an example of how it looks like :

java -Dhttps.proxyHost=456.78.999.99 -Dhttps.proxyPort=8012 -Dhttps.proxyUser=aaaaaaaa -Dhttps.proxyPassword=xxxxxxxxxx -jar pillar4.jar sparql2pillar4 --query cellar\_query\legilux\_query.rq --endpoint https://data.legilux.public.lu/sparqlendpoint --sitemapOutput ...

## **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 or -o : path to the Atom output file

Other optional parameters :

* --title or -t : the title to insert in the Atom header. If not provided, a placeholder value will be used
* --id or -i : the ID of the feed to insert in the Atom header. If not provided, a placeholder value will be used
* --link or -l : the link URL of the feed to insert in the Atom header. If not provided, a placeholder value will be used
* --author or -a : 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 Pillar4 from Cellar**

# To be executed against http://publications.europa.eu/webapi/rdf/sparql

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

select (IRI(STR(?eliString)) AS ?eli) ?updateDate where {

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

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

}

## **Annex : query to generate Pillar4 from Legilux**

# To be executed against https://data.legilux.public.lu/sparqlendpoint

PREFIX jolux: <http://data.legilux.public.lu/resource/ontology/jolux#>

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

WHERE {

?eli a jolux:Act .

?eli jolux:publicationDate ?publicationDate .

OPTIONAL { ?eli jolux:isRealizedBy/jolux:title ?titleLang . }

}

ORDER BY DESC(?publicationDate)