DELIVERABLE

Project Acronym: **EUCases**

Grant Agreement number: **611760**

Project Title: **European and National Legislation and Case Law Linked in Open Data Stack**

D2.1 Report on Crawler Tools

Authors:

Tenyo Tyankov	APIS
Hristo Konstantinov	APIS
Florian Schmedding	AVERBIS
Dr Livio Robaldo	UNITO

Project co-funded by the European Commission within **FP7-ICT-2013-SME-DCA**		
Dissemination Level		
PU	Public	■
PP	Restricted to other programme participants (including the Commission Services	
RE	Restricted to a group specified by the consortium (including the Commission Services	
CO	Confidential, only for members of the consortium (including the Commission Services)	

List of Abbreviations

API	Application programming interface
ASP	Active server pages
DB	Database
DBMS	Database management system
EC	European Community
EU	European Union
EUPL	European Union Public License
FTP	File Transfer Protocol
GUID	Globally Unique Identifier
HTML	Hypertext Markup Language
ID	Identifier
ISO	International Organization for Standardization
MD5	Message-Digest Algorithm 5, a cryptographic hash function
MIME	Multipurpose Internet Mail Extensions
PDF	Portable Document Format
RIS	Rechtsinformationssystem (Legal Information System of the Republic of Austria)
SQL	Search and Query Language
UI	User Interface
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UTC	Coordinated Universal Time
UTF	Unicode Transformation Format
XHTML	Extensible Hypertext Markup Language
XML	Extensible Markup Language
XSD	Extensible Markup Language Schema Definition

Revision History, Status, Abstract, Keywords, Statement of Originality

Revision History

Rev.	Date	Author	Organisation	Description
0.1	27.03.2014	Hristo Konstantinov	APIS	First draft
0.2	10.04.2014	Tenyo Tyankov	APIS	Description of the architecture of the Crawler framework and tools added (section 2)
0.3	11.04.2014	Hristo Konstantinov	APIS	Section 1 about the acquisition of legal open data added
0.4	15.04.2014	Tenyo Tyankov	APIS	Description of the Crawler web service and the crawler tools of APIS (sections 3.1, 3.2 and 3.3.1)
0.5	17.04.2014	Hristo Konstantinov, Tenyo Tyankov	APIS	Section 4 added, source code and software documentation for APIS' crawlers prepared for publication
0.6	29.04.2014	Dr Livio Robaldo	UNITO	Description of the crawler tools of UNITO
0.7	30.04.2014	Florian Schmedding	AVERBIS	Description of the Web crawler developed by Averbis
1.0	05.05.2014	Klaus Piesche	EMPIRICA	Final version
1.1	15.01.2015	Hristo Konstantinov	APIS	Amendments according to the requests of the reviewers

Date of delivery	Contractual:	30.04.2014	Actual:	05.05.2014
Status	final ✓ /draft ☐			

Abstract (for dissemination)	This document presents the results from the work on the development of the crawler framework and tools. Their task is to ensure the initial acquisition and subsequent regular updates of the legal open data relevant to the project objectives. The established common framework connects via web services the various crawlers with the data storage database. More than 40 legal portals and web sites of open access journals have been identified as sources of raw data for the EUCases Linking Platform. The present report provides statistics of the downloaded documents and metadata. The source code of the developed crawler framework and tools is published as open source software on the project web site together with the supporting software documentation.
Keywords	crawler, legal open data, legal portals, download, legislation, case law, open access journal

Statement of originality

This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.

Table of Content

Executive Summary

The acquisition of legal open data is the first stage of the EUCases open data life-cycle. As a rule, it is performed predominantly by specialised software tools for download of the relevant legal documents and metadata, the so called crawlers. Therefore, the first task after the initial study of the legal open data portals in the EU and the 6 project Member States (Austria, Bulgaria, France, Germany, Italy and United Kingdom) was to develop a common crawler framework and tools. The chosen decentralised software architecture allows for simultaneous operation of various software tools developed by different partners using different programming environments. However, crawling web sites is not the only approach for data acquisition. In accordance with Commission policy on open data some legal portals already provide free of charge data delivery services via FTP download, API or web services (EUR-Lex, legislation.gov.uk, Legifrance). In such cases the simple copy-paste operation or the development of a trivial tool for interaction with the API or web services of the portals are fully sufficient for data acquisition.

To ensure compatibility and centralised data storage, a common crawler framework was established. It integrates the following programing modules:

- ***Crawlers*** – each one of them takes care of initial data download, checking for new data or changed data and subsequent download of data updates;
- ***Crawler web service*** – receives downloaded data from each crawler and takes care of validation and storage in a relational database.

More than 40 public web portals and sites of open access journals in the field of law have become sources of raw legal data for the purposes of the EUCases project. After the development of the crawler framework and tools an initial download of the relevant legal contents has been performed. Along with this, a standard procedure for downloading data updates for each of the portals has been approved. In section 1.3.1 are quoted the volumes of the initially downloaded data and the number of downloaded documents for each of the crawled web site.

Section 3 of the present report includes a detailed technical description of the crawler framework and tools. The source code of the developed crawling software with the supporting documentation is published under the European Union Public License (EUPL), v.1.1 on the web site of the project and on github:

http://eucases.eu/fileadmin/EUCases/documents/EUCases_CrawlerFramework&Tools_SourceCode.zip

https://github.com/EUCASES-UNITO/Crawlers.

1 Acquisition of legal open data

1.1 Overview

The acquisition of legal open data is the first stage of the EUCases open data life-cycle. This task is performed in four phases:

- ***Study of the public legal open data portals*** in the EU and the 6 project Member States (Austria, Bulgaria, France, Germany, Italy and United Kingdom) – it was conducted as integral part of the state of the art analysis provided in Deliverable 1.1 "Report on state of the art and user needs". The results from this study were crucial for choosing the right approach for data acquisition (see next section) and for drafting a final list of the legal data portals to be crawled (see section 1.3.1 "Crawled portals")
- ***Design and functional specification of the crawler framework and tools*** – this second phase was completed with the preparation of Deliverable 1.2 "Software project documentation for the linking platform, user requirements, applications, tools and language resources"
- ***Development of the crawler framework and tools and initial data download*** – the present report summarises the results from the performance of work task 2.1 "Development of crawler tools" and describes the volumes and formats of the initially downloaded data
- ***Regular updates with new or changed data*** – in principle, this task will be performed fully automated in the remaining months of the project life-time. However, in the event of changes in the site or data structure of a crawled web portal, some modifications in the respective crawler tool will also be needed.

This chapter presents briefly the approaches for data acquisition, the data sources and the downloaded data sets.

1.2 Approaches

One of the main tasks of the study of the public legal portals in the EU and the 6 project Member States has been to examine the legal terms and conditions for re-use of their data sets. On the basis of the findings a final positive list of the data sources to be used within the EUCases project has been drawn up. Portals which still prohibit the public re-use of their legal databases, such as the Italian Corte Suprema di Cassazione, were excluded from the list.

Depending on the open data and licensing policy of the data publishers the following approaches for acquiring the identified legal open data sets have been applied:

- Data crawling
- Data licensing and delivery services of the portal owner, or
- Combination of the above two approaches.

Below we present briefly these three approaches.

1.2.1 Data crawling

Crawler tools have been developed to download open data from those public portals which explicitly allow the free re-use of their legal contents without prior formal procedure for the conclusion of a license agreement or where the free re-use can be inferred from the provisions of the national legislation in the area of copyright, freedom of information or re-use of public sector information (e.g. the Bulgarian pubic legal portals).

Depending on whether the respective portal offers a convenient means for automated downloading of data or the acquisition is left entirely in the care of the data re-user two types of crawlers have been produced by consortium partners:

- ***Simple crawling tools*** ensuring relatively uncomplicated software classes for interaction with the API or web services provided by the legal portal. Such tools have been developed, for instance, for communication with the *legislation.gov.uk* API[1], the web services of the Hudoc database of the European Court of Human Rights[2] and the German portal *gesetze-im-internet* ensuring free access to the consolidated versions of the German federal legislative acts in XML-format via a list with URL links to their texts published on the Government open data portal.[3]
- ***Complex crawling tools*** for consecutively visiting and downloading published legal documents and metadata, often by repeatedly querying a search form (deep web resources crawling), e.g. the case law portal of the Bulgarian Supreme Judicial Council.

1.2.2 Data licensing and delivery services

This approach is applicable for public legal portals which provide specialised data delivery services via FTP. Data re-users are granted a license and can rely on receiving a snapshot copy of the full data set (for downloading in bulk) and regular updates on a daily, weekly or monthly basis. Currently, there are three public portals in EU and the six project Member States which offer such kind of services: EUR-Lex (EU), Legifrance (France) and Rechtsinformationssystem (Austria).

As of 2014 the EUR-Lex portal offers its data licensing and delivery service for free. APIS as the partner who is responsible for acquisition of the EU legal resources has applied for EUR-Lex license and regularly downloads data updates by simple copy-paste operation from the FTP server of the Publications Office. Due to provisional technical problems of the new EUR-Lex portal, only daily and monthly updates are currently available to FTP users. However, the archive with the historical data will be soon available to all FTP users for download. In the event of a long delay APIS is ready to download the data via a web service provided for free by the EUR-Lex portal.

A major change in the licensing policy is expected also for the Legifrance portal. In its correspondence with representatives of Legifrance the consortium partner APIS was assured that later in 2014 a new free of charge license for delivery of the portal's data sets in XML format will be introduced. In order to fulfil in due time its obligations under the project APIS has developed a crawler for downloading French case law from the Legifrance portal. However, the French data will be downloaded only after the new licensing policy will be introduced since the current rules do not allow the download of a large amount of documents.

Because of the still relatively high fees for annual subscription for data delivery services of the RIS portal, the consortium decided to apply the combined approach for acquisition of Austrian legal data (see next section).

[1] http://www.legislation.gov.uk/developer.

[2] http://hudoc.echr.coe.int/sites/eng/Pages/search.aspx#.

[3] https://www.govdata.de/suchen/-/details/gesetze-im-internet.

1.2.3 Combined approach

In the case of Austrian legal data APIS (as the responsible partner in EUCases consortium) has applied for one-time purchase of a single copy of the relevant data sets of the portal Rechtsinformationssystem (RIS) and in the same time has developed a crawler for downloading the documents updated after the purchase date. The reason to use this combined approach for acquisition of Austrian legislation and case law was the fact that 1) the one-time purchase of one RIS data set (the so called RIS application) costs only EUR 110.00 whereas the fee for daily or weekly data updates amounts EUR 1,000.00 per RIS application and year, and 2) the free download of RIS data is explicitly allowed without any restrictions in respect of the number of downloaded documents. Altogether four RIS applications have been purchased by APIS on the amount of EUR 440.00:

- Bundesrecht konsolidiert (Consolidated federal legislation)
- Verfassungsgerichtshof (Constitutional Court)
- Verwaltungsgerichtshof (Supreme Administrative Court)
- Justiz – Oberster Gerichtshof, Oberlandesgerichte, Landesgerichte, Bezirksgerichte, Oberster Patent- und Markensenats (Justice – Supreme Court of Justice, State Courts of Appeal, State Courts, Regional Courts, Supreme Patent and Trade Mark Senate).

For all these application a crawler to download updated and new documents published after the purchase date has been developed. The same crawler is also used for the new RIS applications providing as of 01.01.2014 case law of the newly established Bundesverwaltungsgericht (Federal Administrative Court) and Landesverwaltungsgerichte (State Administrative Courts).

1.3 Sourced legal open data

1.3.1 Crawled portals. Description of the acquired legal open data

More than 40 public web portals and sites of open access journals in the field of law have been identified as sources of raw legal data for the purposes of the EUCases project. Crawlers for downloading the relevant data have been developed for most of them. Direct download by a simple copy-paste operation from an FTP server of the data provider is foreseen for the portals EUR-Lex and Legifrance only. Nevertheless, a crawler for downloading French case law from Legifrance was developed to prevent a possible delay in the introduction of the new free license for FTP data delivery service. An FTP download has been also used to source the relevant legal data of the portal Rechtsinformationssystem (RIS) of the Austrian Federal Chancellery. However, this download included only a snapshot copy of the respective data sets. Therefore, also in this case a crawler to download the following data updates from RIS was developed.

After the development of the crawler tools an initial download of the relevant legal contents has been performed. Along with this, a standard procedure for downloading data updates for each of the portals has been approved (see next section 1.3.2 “Data update procedures”). The table below presents the results from the initial download: the portals and sites being sources of legal data for EUCases project, the responsible project partner and the type of tool developed/used for data download, the type of the downloaded legal content, the data format and language, the data volumes (in GB), the number of downloaded documents and the download speed.

Table 1 – Crawled legal portals

Nr.	Portal	Partner & tool	Content type	Format	Language	Volume, nr. of downloaded docs, download speed
1.	**EUR-Lex** - http://eur-lex.europa.eu - http://new.eur-lex.europa.eu	APIS FTP service	- Treaties (Sectors 1) - International agreements (Sectors 2) - Secondary Legislation (Sectors 3) - Complementary Legislation (Sectors 3) - Preparatory acts (Sectors 5) - Case law (Sector 6) - National Execution Measures (Sector 7) - National case law (Sector 8)	XML (Formex) XHTML	EN, DE, FR, IT, BG	***Not relevant (*)***
2.	**Gesetze im Internet** (Laws on the Internet) - www.gesetze-im-internet.de	APIS Crawler	- Legislation	XML	DE	158 MB; 6323; 00:21 h
3.	**Verwaltungsvorschriften im Internet** (Regulations on the Internet) - www.verwaltungsvorschriften-im-internet.de	APIS Crawler	- Administrative regulations	XML	DE	6 MB; 715; 00:02 h
4.	**Bundesverfassungsgericht** (Federal Constitutional Court) - www.bundesverfassungsgericht.de	APIS Crawler	- Case law	HTML	DE	44 MB; 5394; 00:36 h
5.	**Bundesgerichtshof** (Federal Court of Justice) - www.bundesgerichtshof.de	APIS Crawler	- Case law	PDF	DE	2.91 GB; 44730; 03:28 h
6.	**Bundesverwaltungsgericht** (Federal Administrative Court) - www.bverwg.de	APIS Crawler	- Case law	HTML, PDF	DE	99 MB; 20683; 01:59 h
7.	**Bundesfinanzhof** (Federal Fiscal Court) - www.bundesfinanzhof.de	APIS Crawler	- Case law	HTML	DE	28 MB; 4615; 00:17 h
8.	**Bundesarbeitsgericht** (Federal Labour Court) - www.bundesarbeitsgericht.de	APIS Crawler	- Case law	HTML	DE	44 MB; 2537; 00:18 h
9.	**Bundessozialgericht** (Federal Social Court) - www.bsg.bund.de	APIS Crawler	- Case law	HTML	DE	30 MB; 1131; 00:10 h
10.	**Bundespatentgericht** (Federal Patent Court) - www.bundespatentgericht.de	APIS Crawler	- Case law	PDF	DE	1.43 GB; 23841; 01:12 h
11.	**Legifrance** - www.legifrance.gouv.fr	APIS FTP service + Crawler	- Case law	HTML	FR	***Not relevant (**)***
12.	**Legislation.gov.uk** - www.legislation.gov.uk	APIS Crawler	- Legislation	XML	EN	477 MB; 75100; 12:43 h
13.	**Judiciary.gov.uk** - www.judiciary.gov.uk	APIS Crawler	- Case law	PDF, HTML	EN	111 MB; 523; 01:34 h
14.	**Supreme Court (UK)** - www.supremecourt.uk	APIS Crawler	- Case law	PDF, HTML	EN	99 MB; 374; 00:04 h
15.	**House of Lords Judgments: archive**	APIS	- Case law	HTML	EN	11 MB; 786; 00:03 h

Nr.	Portal	Partner & tool	Content type	Format	Language	Volume, nr. of downloaded docs, download speed
	- http://www.publications.parliament.uk/pa/ld/ldjudgmt.htm	Crawler				
16.	**Rechtsinformationssystem (RIS)** (Legal information system) - www.ris.bka.gv.at	APIS FTP service + Crawler	- Legislation - Case law	XML, HTML	DE	1.3 GB; 436115; 11:29 h
17.	**Darzhaven vestnik** (State Gazette) - http://dv.parliament.bg	APIS Crawler	- Legislation	HTML	BG	151 MB; 59993; 23:34 h
18.	**Varhoven Kasatsionen Sad** (Supreme Court of Justice) - www.vks.bg	APIS Crawler	- Case law	HTML	BG	1.22 GB; 95166; 28:45 h
19.	**Varhoven Administrativen Sad** (Supreme Administrative Court) - www.sac.government.bg	APIS Crawler	- Case law	HTML	BG	729 MB; 190404; 46:29 h
20.	**Portal "Sadebni aktove" – Vish Sadeben Savet** (Case Law Portal of the Supreme Judicial Council) - http://legalacts.justice.bg	APIS Crawler	- Case law	HTML, PDF	BG	39 GB; 1074969; 152:16 h
21.	**Normattiva** - www.normattiva.it	UNITO Crawler	- Legislation	XML, HTML	IT	2.33GB; 79636; 35:20 h
22.	**Corte Constituzionale** - www.corteconstituzionale.it	UNITO Crawler	- Case law	XML, HTML	IT	3.125 GB; 55636; 140 h
23.	**Consiglio di Stato** - www.giustizia-amministrativa.it	UNITO Crawler	- Case law	HTML	IT	51 GB; 1748915; 875 h
24.	**HUDOC database** (Case Law Portal of the European Courts of Human Rights) - http://hudoc.echr.coe.int	APIS Crawler	- Case law	HTML	EN	358 MB; 49170; 10:16 h
25.	**JIPITEC – Journal of Intellectual Property, Information Technology and E-Commerce Law** - www.jipitec.eu	AVERBIS Crawler	- Articles	PDF	EN	44,4 MB; 95; n/a
26.	**Utrecht Journal of International and European Law** - www.utrechtjournal.org	AVERBIS Crawler	- Articles	PDF	EN	18,6 MB; 52; n/a
27.	**European Journal of Law and Technology** - ejlt.org	AVERBIS Crawler	- Articles	PDF	EN	15,6 MB; 76; n/a
28.	**Web Journal of Current Legal Issues** - webjcli.org	AVERBIS Crawler	- Articles	HTML	EN	2,4 MB; 30
29.	**Home - GoJIL - Goettingen Journal of International Law** - www.gojil.eu	AVERBIS Crawler	- Articles	PDF	EN	27,6 MB; 137
30.	**Juridica International** - www.juridicainternational.eu	AVERBIS Crawler	- Articles	PDF	EN	73,7 MB; 332, 00:15 h
31.	**Comparative Law Review** - www.comparativelawreview.com/ojs/index.php/CoLR	AVERBIS Crawler	- Articles	PDF	EN	15,5 MB; 52; 00:08 h
32.	**International Free and Open Source Software Law Review** - www.ifosslr.org/ifosslr/index	AVERBIS Crawler	- Articles	PDF	EN	17,8 MB; 64; 00:05 h

Nr.	Portal	Partner & tool	Content type	Format	Language	Volume, nr. of downloaded docs, download speed
33.	**Law Review (International Journal of Law and Jurisprudence Online Semi-annually Publication)** - www.internationallawreview.eu	AVERBIS Crawler	- Articles	PDF	EN	9,2 MB; 64; 00:08 h
34.	**Juridical Tribune** - www.tribunajuridica.eu	AVERBIS Crawler	- Articles	PDF, HTML	EN	(***)
35.	**Agora International Journal of Juridical Sciences** - univagora.ro/jour/index.php/aijjs/index	AVERBIS Crawler	- Articles	PDF	EN	9,7 MB; 102; 00:10 h
36.	**International Journal for Court Administration** - www.iacajournal.org/index.php/ijca	AVERBIS Crawler	- Articles	PDF	EN	18,5 MB; 89; 00:13 h
37.	**Bocconi Legal Papers - A Student-Edited Journal** - bocconilegalpapers.org	AVERBIS Crawler	- Articles	PDF, HTML	EN	8,35 MB; 28; 00:18 h
38.	**ELSA - The European Law Students' Association - Malta Law Review** - elsamaltalawreview.com	AVERBIS Crawler	- Articles	PDF, HTML	EN	(***)
39.	**Beijing Law Review - Social Sciences & Humanities - Journals - SCIRP** - www.scirp.org/Journal/blr	AVERBIS Crawler	- Articles	PDF, HTML	EN	(***)
40.	**Bulletin of the Transilvania University of Brasov - Series VII - Social Sciences & Law** - webbut.unitbv.ro/bu2011/Series VII/Series VII.html	AVERBIS Crawler	- Articles	PDF, HTML	EN	(***)
41.	**Electronic Journal of Comparative Law** - www.ejcl.org	AVERBIS Crawler	- Articles	PDF, HTML	EN	(***)

(*) Data will be downloaded via FTP server after the provisional technical problem of the new EUR-Lex portal with the publication of the archive data sets will be solved.

(**) Data will be downloaded via FTP server after the new free of charge data delivery service of the Legifrance portal will be introduced.

(***) Data has not been downloaded yet.

1.3.2 Data update procedures

The distributed architecture of the crawler framework and tools allows that each crawler can be run on one or various PCs or servers simultaneously or at different time. This flexible approach was chosen, because data will be crawled by various partners and tools in compliance with the frequency of information update and the download speed of the respective web sites.

A. Update procedures for APIS crawlers

All crawlers are in a single command line application and depending on the configuration file a different site is crawled. Crawlers read configuration from the ‘app.config’ file. For a detailed description of the configuration file see “Crawler configuration” in Section 3.3.1 “APIS’ crawlers”.

As long as all different crawlers logic is in the same executable, if more than one crawler is needed to start simultaneously on the same PC/server the following must be done: a copy of the application must be made in as many different folders as needed. For each copy corresponding amendments in the 'app.config' file must be made depending on the site to be crawled (see Main configuration in Section 3.3.1 "APIS' crawlers": "Crawler configuration").

When a crawler is started a check for existence of SQLite DBMS with the according structure is performed (SQLiteData\\SQLiteCrawlerData.sqlite). If the database does not exist an empty one is copied from SQLiteData\\SQLiteCrawlerData.sqlite.

After initial download all the documents will be processed and sent to the Crawler web service. For each document MD5 sum and other information is stored in SQLite DB. Depending on the frequency of document updates on the crawled web site crawlers must be run at appropriate time intervals. In each subsequent launch after the initial download of data the crawler performs downloading of all the documents and verification against the information stored in the database. MD5 sum comparison is performed and according to it the crawler decides if a document is in *Add*, *Update*, *Delete* or *None* mode (ADD, UPD, DEL, NONE) and this information is stored in the database.

After download and verification are finished, all the documents are sent to the Crawler web service for further processing. It is to be noticed here that if there is no connection with the web service all relevant document information will be sent when the crawler is started next time.

It is recommended that each crawler tool is put as a task in „Task scheduler" of the Windows OS. Appropriate frequency of execution should be selected according to the web site data change frequency, for instance: once a day or once a week.

B. Update procedures for UNITO crawlers

Every crawler is available from command line or user interface. Each interface has a configuration that can be changed depending on the needs and structure in the client computer. Crawlers can save and read configuration file. For a detailed description of the configuration file see "Crawler configuration" in Section 3.3.2 "UNITO's crawlers".

As long as all different crawlers logic is in different executable, every crawler can start simultaneously on the same PC/server (see Section 3.3.2 "UNITO's crawlers": "Crawler configuration").

When a crawler is started, it creates different work folders. After initial download, all the documents will be processed and sent to the Crawler web service. For each document MD5 sum and other information is stored in local repository. Depending on the frequency of document updates on the crawled web site crawlers must be run at appropriate time intervals. In each subsequent launch after the initial download the crawler performs downloading of all the documents and verification against the information stored in the database. MD5 sum comparison is performed and according to it the crawler decides if a document is in *Add*, *Update*, *Delete* or *None* mode (ADD, UPD, DEL, NONE) and this information is stored in the database.

After download and verification are finished, all the documents are sent to the Crawler web service for further processing. If there is no connection with the web service, the rollback is done and the procedure has to be repeated.

C. Update procedures for AVERBIS crawlers

The crawler from Averbis is intended to download articles from Open Access Journals. It is available from the command line. After starting it begins to download the initial seed documents. Then it follows each permitted link. The permission is checked using regular expressions on the URLs. When the crawler encounters an article overview page it extracts metadata such as title and authors and annotates the outgoing link to the PDF version of the article. After finishing the retrieval of the PDF file a document containing the metadata and the binary content is send to an indexing process. The indexing process is capable of detecting additions and deletions. The current indexer writes each document to the file system.

1.4 Scalability of the crawler tools

From the very beginning the vision for the future EUCases service has been and is based on its gradual extension with additional data sources from other EU countries and/or new legal portals. Therefore the scalability of modules and tools the developed within the project lifetime is an important issue respected by all project participants.

However, crawling data from public legal portals is quite specific activity which determines the relatively low scalability level of the crawler tools developed by APIS and UNITO. *The first reason* for this is that legal information is presented in a structured, but not uniform way. Sometimes crawlers should download in a bundle two or more documents being different parts of a single legal document. In other cases documents are accompanied by metadata of important legal value which have to be downloaded in one package with document's text. *The second reason* for the specificity of the crawlers of legal data is the fact that legal documents often are stored in a database which contents are inaccessible for conventional tools for automatic download. In such cases crawlers should perform searches and download the texts and metadata appearing in the search results. All this makes the development of crawlers of legal data fully dependent on the concrete structure of the portal to be crawled. They should be developed on a case-by-case basis. Depending on the complexity of the legal portal and of the legal data structure, the development of a new crawler may require between 3 and 15 days of work.

Notwithstanding the foregoing, the built crawling framework provides a common infrastructure for the development of new crawlers, thus reducing the efforts for programming of a new crawler approximately by 20%.

A little bit different is the situation with the journal crawler developed by AVERBIS. The journal crawler is not specific to legal journals. Open access journals are crawled by using a very well known, open source framework for web crawling, i.e. Apache Nutch. Nevertheless, to additionally include further sites containing PFDs of scientific articles, one has to provide Nutch-plugins in order to properly collect meta-information available on the corresponding HTML pages (for example title, journal, issue information above or beneath the HTML-link to the PDF article). Basically, the crawler looks for pages that contain meta information about an article and the corresponding links. The crawler extracts the meta data from these pages and attaches it to the link pointing to the PDF document. When the crawler processes these links – i.e. downloads the PDF documents – the article meta data is available and can be sent together with each article to the receiving database. Depending on the page structure, the meta data extraction may be performed with generic tools or may require a very specific processing. The pages that should be inspected by the crawler and the relevant meta data fields can be configured for each journal. Depending on the structure and composition of the HTML-resource this will result in a less or more complex task, approximately ranging between 5 and 15 days of work.

2 Architecture of the crawler framework and tools

2.1 Overview

Since large volumes of data are envisaged for crawling from various legal portals a distributed architecture has been chosen for the design of the crawler framework and tools. This approach allows for independent operation of each of the crawlers and guarantees a reliable and centralised storage of downloaded data for further processing. Another good reason gave the fact that the crawlers had to be developed by several project partners (APIS, UNITO and AVERBIS) and with different programming technologies (.Net Framework, Java and C++). To ensure compatibility and centralised data storage, a common crawler framework was established. It integrates the following programing modules:

- ***Crawlers*** – each one of them takes care of initial data download, checking for new data or changed data and subsequent download of data updates;
- ***Crawler web service*** – receives downloaded data from each crawler and takes care of validation and storage in a relational database. This web service was implemented with Windows Communication Foundation (WCF) and Microsoft Visual Studio 2013. Since WCF supports SOAP (Simple Object Access Protocol), this has allowed to solve the problem with the usage of different programming technologies (.Net Framework, Java and C++).

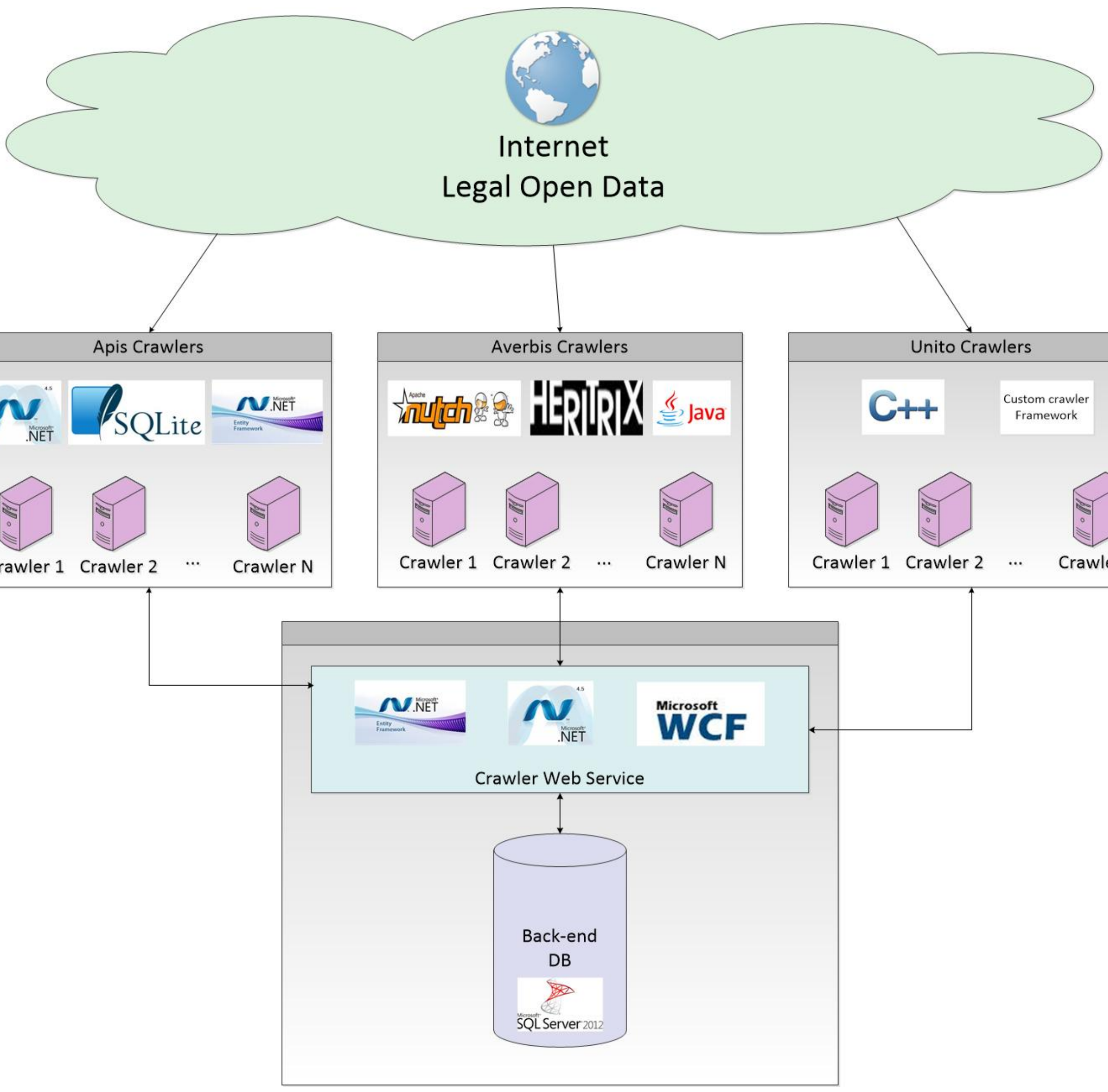
The main advantages of the chosen distributed architecture are:

- Data from web sites can be downloaded on different computers independently of one another
- Even when the Crawler web service is not available, it will not interfere with the operation of any of the crawlers, because downloaded data could be submitted afterwards
- The use of common standards for the implementation of Crawler web service allows crawlers to be implemented in different programming technologies
- Centralised and secure data storage in a relational database MSSql server
- Accuracy of data as they are validated by the Crawler web service before being stored in the database
- Traceability of the stored data – detailed information is recorded in the database in the process of submitting and storing data
- Easy maintenance of the data – back-up and further processing.

2.2 Architecture diagram

The following figure presents the distributed architecture of the Crawler framework and tools:

Figure 1 – Architecture of the Crawler framework and tools



3 Crawler framework and tools description

3.1 Technologies

The Crawler framework and tools have been developed by three partners of the EUCases consortium: APIS, UNITO and AVERBIS. Each of them has extensive experience in working with different integrated development environments. Therefore, the used technologies are very diverse.

Whereas AVERBIS has chosen Apache Nutch crawling framework for the development of a crawler for downloading open access articles (quite suitable for this particular task), APIS and UNITO had preferred to use technologies already well known to them from previous projects and internal work. The decision to distribute the development of crawlers for legal portals to APIS and UNITO, i.e. to more than one partner, was conditioned not only by the desire for allocation of tasks, but also by the fact that UNITO had some existing tools that could be reused with small modifications. Another good reason is the relatively low scalability level of crawling solution for legal data which was already discussed in section 1.4 above. Approximately 80% of the cost of crawlers rests in the code specific for the portal. Furthermore, continuous maintenance is needed when portals are modified, so it is needed to have a technology which is well known by the managers of the crawlers. This adds to the fact that, as specified in the projects' "Description of work", for UNITO there was already a crawler for Normattiva legal portal which has been the basis for the new one. Normattiva has some peculiarities in how sessions are dealt with, which need special coding. This was the main reason to distribute the development of crawlers for Italian legal portals to UNITO instead of assigning such task solely to APIS.

Moreover, also the Giustizia amministrativa portal has some complexities due to longer terms of replies which are not accepted by standard networking libraries.

Crawler web service

The Crawler web service has been developed by APIS using Microsoft Visual Studio 2013, .NET Framework 4.5, Widows Communication Foundation and Microsoft SQL Server 2012.

Crawlers

The following software and technologies have been used for the development of the various crawler tools:

- ***APIS*** – Microsoft Visual Studio 2013, Entity Framework, SQLite v3 database
- ***UNITO*** – internal framework developed in C++ that is able to extract information from structured text by means of a set of pre-compiled regular expressions and a matching engine and used in previous work for crawling data from Italian legal portals
- ***AVERBIS*** – Apache Nutch, Nutch plugins, and Java for developing connector to Crawler web service and extending frameworks.

3.2 Crawler web service

The Crawler web service is developed on WCF (Windows Communication Foundation), .NET Framework 4.5 and Microsoft SQL Server 2012. Its main purpose is to provide a common interface to all crawlers to submit legal documents to the service and to store them in a relational database.

The Crawler web service provides one method with a single argument:

```
string UploadFile(UploadDocumentGroup document)
```

The argument UploadDocumentGroup is complex type and has two public properties:

- *MetaInfo* – Provides all needed meta information about the submitted document, such as language, date of download, format of the document, etc. It is an XML string encoded in UTF8 which is validated by XSD schema (Appendix I). All elements and attributes are required and are validated against the schema
- *Data* – This is an array of bytes in ZIP format. It contains all files describing the submitted document (e.g. a legal document with images).

Description of "MetaInfo" XML string

Since there are documents that are composed of group of files, they need to be sent in a package as a single file. The package is a ZIP archive. The accompanying "**MetaInfo**" XML string describes the package and lists all files related to the document.

The element 'documentgroup' describes the package itself and the element 'document' describes each file in the package.

```
<documentgroup …>
        <document …/>
        <document …/>
        …..
</documentgroup>
```

Attributes of the 'documentgroup' element:

- **date** – string representation of the date and time when the document is downloaded. It must be written in conformance with ISO 8601 yyyy-MM-ddTHH:mm:ss (date="2013-11-21T09:50:39") and UTC compatible
- **crawler** – string identifier of the crawler uploading the document (e.g. crawler="UKCaseLawDownloadSupremecourt")
- **lang –** language of uploaded document in ISO 639 standard (e.g. lang="EN")
- **format** – defines the file type of the document in Internet Media Type (MIME type) standard[4] (e.g. format="application\zip")
- **file** – string filename (e.g. file="UKSC_2010_0237.zip")
- **identifier** – unique identifier of the document across all crawlers. To ensure this, the identifier must be Globally Unique Identifier (GUID)
- **operation** – one of the following: Add – new document, Upd – updated document, Del – deleted document, None – no change
- **type** – numeric representation of the type of the document: 1 – Legislation, 2 – Case Law.

Attributes of the 'document' element:

- **format** – defines the file type of the document in Internet Media Type (MIME) standard (format="text/html")
- **file** – string filename (e.g. file="300113B9B51.12.0.html")
- **identifier** – unique identifier of the document across all crawlers. To ensure this, the identifier must be GUID

[4] "RFC2046: Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", N. Freed and N. Borenstein, November 1996. Available at http://www.ietf.org/rfc/rfc2046.txt.

- **operation** – one of the following: Add – new document, Upd – updated document, Del – deleted document, None – no change None – no change
- **url** – URL of the source from which the document was downloaded
- **md5** – MD5 hash sum of the document.

Here is an example of the XML description:

```
<?xml version="1.0" encoding="utf-8"?>
        <documentgroup xmlns:xsd=http://www.w3.org/2001/XMLSchema
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" date="2014-01-16T06:55:18"
        crawler="DCrawlerBverwg" lang="EN" format="application/zip"
        filename="300113B9B51.12.0.zip" identifier="442261d3-6737-40bc-aa77-6fd118a56882"
        operation="None" type="1">
                <document format="text/html" file="300113B9B51.12.0.html" identifier="00fb4dae-
                4a34-49b8-8dd2-4a587e8ed2e0" operation="None"
                url="http://www.bverwg.de/entscheidungen/entscheidung.php?ent=300113B9B51.12.0
                " md5="e6444e96d13c4cffd516c39ccde76677" />
        </documentgroup>
```

Service workflow

The workflow comprises the following sequential steps:

- *Document validation* – To ensure trouble-free further processing of documents, each one of them submitted for storage, is validated against an XSD schema (Appendix I)
- *Deserialisation* of METAInfo XML into objects – This operation ensures easy processing of the metadata
- *Zip validation* – If a document group from the type ADD or UPD is submitted, validation is done (whether the zip is not empty, whether it contains documents with the same names as those submitted with the METAInfo XML)
- *Unzip of the submitted data in the memory* – This is done, because documents are stored in the database in uncompressed format
- *Storing data* – Documents are stored in the database by using the relevant operation:
  - *ADD* – when new documents are stored
  - *UPD* – when documents existing in the database are updated
  - *DEL* – when documents existing in the database are deleted.
- *Writing information in the CrawlerLog table* – For each submitted document logging information is written in the database. This is very helpful for future statics and for tracing of whole process of submitting and storing documents. If there is no connection with the database, the logging information is written locally in a file. When the connection is restored, the locally saved logging information is read and stored in the database.

Errors returned by the Crawler web service

- **XML Validation** – errors are returned in the following format:
  ******XML Validation error***
  Severity:
  Message:
- **ZipData validation** – it is not allowed to submit an empty archive for operation other than DEL

Validation Message: ZipData length is zero, document identifier: {identifier} and Operation: {Operation}

- **Operation validation** – occurs in the following cases:
  - If a document group is submitted by ADD operation and there is such a group with the same identifier
    Validation Message: There is a document identifier: {identifier}. You can't use Operation: {Operation}
  - If a document group is submitted/deleted by UPD/DEL operation and there is no such group with the same identifier
    Validation Message: No document identifier: {identifier}. You can't use Operation: {Operation}
- **Exception message** – it is displayed when the service throws an exception (e.g. there is no connection with the database). The message contains exception description and stack trace.

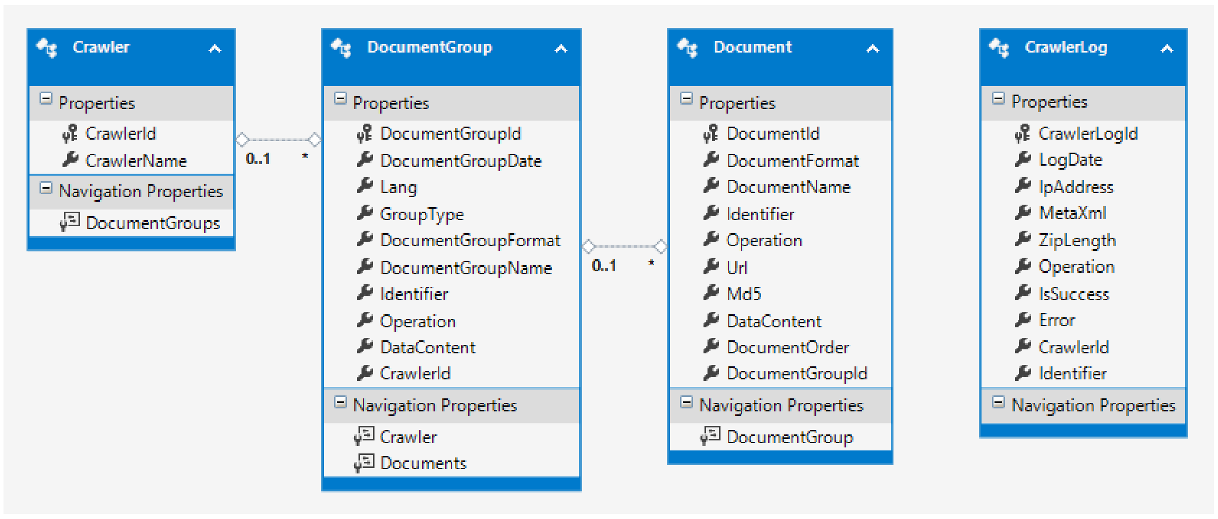
Description of the database tables

The Crawler web service is using Entity Framework to store submitted documents into the relational database (Microsoft SQL server 2012 standard). In order to facilitate subsequent processing of the data, upon receipt and verification file packages are uncompressed. Storing documents in the database is carried out in the following tables:

- *Crawlers* – Contains information about the name of the crawler
- *DocumentGroup* – Contains information for each group of documents
- *Document* – Contains information for each document in a document group
- *CrawlerLog* – contains information for every submitted document and errors in working process.

The figure below shows the attributes of the tables and the relationships between them:

Figure 2 – Storing documents in the database: tables attributes



Error handling

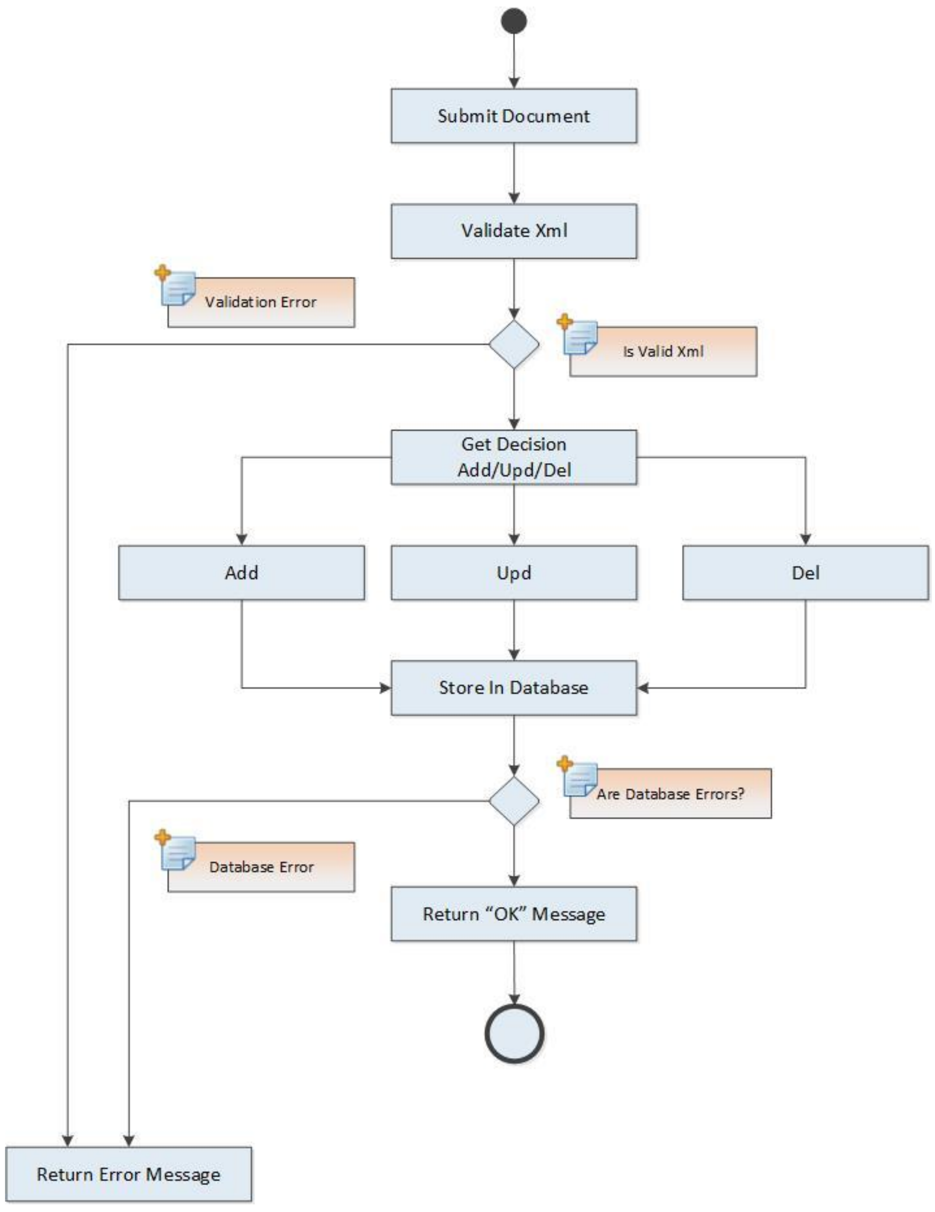
The method *'UploadFile'* returns string as a result of processing input arguments.

- If there are no errors, the return string is "OK"
- If there is an error, the return string will be in following format:

Error: <error description>

The figure below shows visually the process of receiving and storing data in database:

Figure 3 – Receiving and storing documents in the database



Crawler web service configuration

The Crawler web service reads configuration from web.config file.

- **Database configuration**

```
<add name="CrawlerFrameworkEntities" connectionString="metadata=res://*/CrawlerFra
meworkModel.csdl|res://*/CrawlerFrameworkModel.ssdl|res://*/CrawlerFrameworkModel.
msl;provider=System.Data.SqlClient;provider connection string=&quot;data source=YourS
qlServer;initial catalog=CrawlerFramework;persist security info=True;user id=YourUserId;
password=YourPassword;MultipleActiveResultSets=True;App=EntityFramework&quot;" pr
oviderName="System.Data.EntityClient" />
```

- **Windows Communication Foundation** – in order to process large files (up to 20 MB) custom configuration needs to be done.

```
      <system.serviceModel>
    <services>
          <service name="ServiceEUCases.ServiceEUCases">
           <endpoint address="" binding="basicHttpBinding" bindingConfiguration="CustomConfi
      g" contract="ServiceEUCases.IServiceEUCases" />
      </service>
     </services>
     <bindings>
      <basicHttpBinding>
        <binding name="CustomConfig" closeTimeout="00:10:00" openTimeout="00:10:00" send
Timeout="00:10:00" maxBufferPoolSize="2147483647" maxBufferSize="2147483647" maxRe
ceivedMessageSize="2147483647" messageEncoding="Mtom">
          <readerQuotas maxDepth="2147483647" maxStringContentLength="2147483647" max
ArrayLength="2147483647" maxBytesPerRead="2147483647" maxNameTableCharCount="2
147483647" />
        </binding>
      </basicHttpBinding>
     </bindings>
     <behaviors>
      <serviceBehaviors>
        <behavior>
          <!-
 To avoid disclosing metadata information, set the values below to false before deployment -->
            <serviceMetadata httpGetEnabled="true" httpsGetEnabled="true" />
           <!-
       To receive exception details in faults for debugging purposes, set the value below to true.
       Set to false before deployment to avoid disclosing exception information -->
            <serviceDebug includeExceptionDetailInFaults="false" />
        </behavior>
      </serviceBehaviors>
     </behaviors>
     <protocolMapping>
      <add binding="basicHttpsBinding" scheme="https" />
     </protocolMapping>
     <serviceHostingEnvironment aspNetCompatibilityEnabled="true" multipleSiteBindingsEnabl
ed="true" />
   </system.serviceModel>
```

- **Database creation**

ServiceEUCases.Data project contains the file CrawlerFramework.sql in the subdirectory “Scripts”. This file contains all sql scripts needed for initial creation of the database. When using this script, special attention is needed, because the script will drop and recreate database with the name “CrawlerFramework”.

Description of the common types (interfaces, classes, enumerations)

IServiceEUCases Interface

This interface provides functionality for submitting a single DocumentGroup to the Crawler Web service.

Public Member Functions

- *UploadFile* (*UploadDocumentGroup* document).

ServiceEUCases Class

Implements IServiceEUCases interface, verifies the validity of the DocumentGroup and stores it into the database.

Public Member Functions

- *UploadFile* (*UploadDocumentGroup* uploadDocumentGroup) – This method is used to submit DocumentGroup to the Crawler web service.

UploadDocumentGroup Class

This class stores the needed information about the submitted DocumentGroup.

Properties

- *MetaInfo* – XML description of the DocumentGroup
- *Data* – Byte array containing the data for the DocumentGroup in ZIP format.

DocumentGroup Class

This class provides information for a single document group.

Public Member Functions

- *DocumentGroup ()* – Constructor of the class.

Properties

- *<DocumentMetaInfo> Document* – List of DocumentMetaInfo objects
- *Date* – String representation of the date and time when the document group has been downloaded
- *Crawler* – String identifier of the crawler that is downloading the documents
- *Lang* – Language of the document group
- *Format* – File type of the document
- *FileName* – File name of the document group
- *Identifier* – Unique identifier (GUID) of the document group
- *Operation* – Type of the operation for this document group
- *GroupType* – Type of the document (1 – Legislation, 2 – Case Law).

DocumentMetaInfo Class

This class provides information for a single document.

Properties

- *DataContent* – Binary representation of the crawled document
- *Format* – File type of the document
- *File* – File name of the document
- *Identifier* – Unique identifier (GUID) of the document
- *Operation* – Type of the operation for this document
- *Url* – The URL of the crawled document
- *Md5* – The MD5 hash sum of the document.

3.3 Crawler tools: technical specification

3.3.1 APIS' crawlers

APIS is developing the crawlers for downloading legislation and case law from the legal portals in EU and in the following Member States: Austria, Bulgaria, France, Germany and United Kingdom as well as from the web portal of the European Court of Human Rights.

The study of the legal portals has shown that often legal materials are published as compound documents consisting of two or more files, e.g. main document with annexes or images. In order to preserve the structure of such compound documents, each crawler must take care to combine the files composing the document in a zip group (package), which contains an XML description in the following format:

```
<documentgroup …>
        <document …>
        ……………..
        </document>
</documentgroup>
```

The 'documentgroup' element describes the information about the entire group of files constituting the compound document, and the 'document' element contains information about each file.

The attributes of the 'documentgroup' and of the 'document' element have been listed above (see also the example of the XML description).

Information for each document group is preserved in the following way:

- All files are archived DocumentGroup in Zip format and saved to the file system

- The metadata for the DocumentGroup is stored in a relational database (SQLite).

It is important to note that downloaded data may fluctuate (e.g. newly published court decisions have been downloaded, consolidated version of a legal act has been modified, etc.) and each of the crawlers should take care to their frequent updates. This is performed by periodically downloading the data and checking for changes in any of the downloaded files.

Crawler download phases

- **Initial download** – During the initial download of data the crawlers generate MD5 hash sum for each downloaded file and this information is saved in the SQLite database. All files are marked as new (attribute 'operation' of the 'document' element) and are submitted to the Crawler web service for storing and processing.
- **Subsequent download** – In each subsequent download of data for each downloaded file crawlers generate MD5 sum again and check if the file is new, has been modified or deleted. If the MD5 sum is different, this information will be reflected in the XML description (attribute 'operation' of the 'document' element) and the file is sent to the Crawler web service for storing and processing.

Crawler configuration

All crawlers developed by APIS are in a single command line application. For their development Microsoft Visual Studio 2013 and the relational database SQLite have been used. Depending on the configuration file a different site is crawled. Crawlers read configuration from the app.conig file.

- **Main configuration –** the section "<appSettings>" contains the following keys:
    - DestinationFolder – working directory for each crawler
    - Language – language of the crawled documents
    - MaxDegreeOfParallelism – count of parallel documents to be downloaded
    - CrawlerName – name of the crawler to be executed. This is a very important key, because it specifies the name of the crawler class who is responsible for downloading the corresponding web site.

For instance:

```
<appSettings>
  <add key="DestinationFolder" value="D:\Crawler\Bundespatentgericht" />
   <add key="Language" value="DE" />
   <add key="MaxDegreeOfParallelism" value="2" />
   <add key="CrawlerName" value="DE.Bundespatentgericht" />
 </appSettings>
```

- **Service connection configuration –** with this settings the communication with the Crawler web service (timeout session, maximum document size, etc.) is being configured. Currently, the maximum volume of the documents that can submitted to the WCF web service is set to 20 MB.

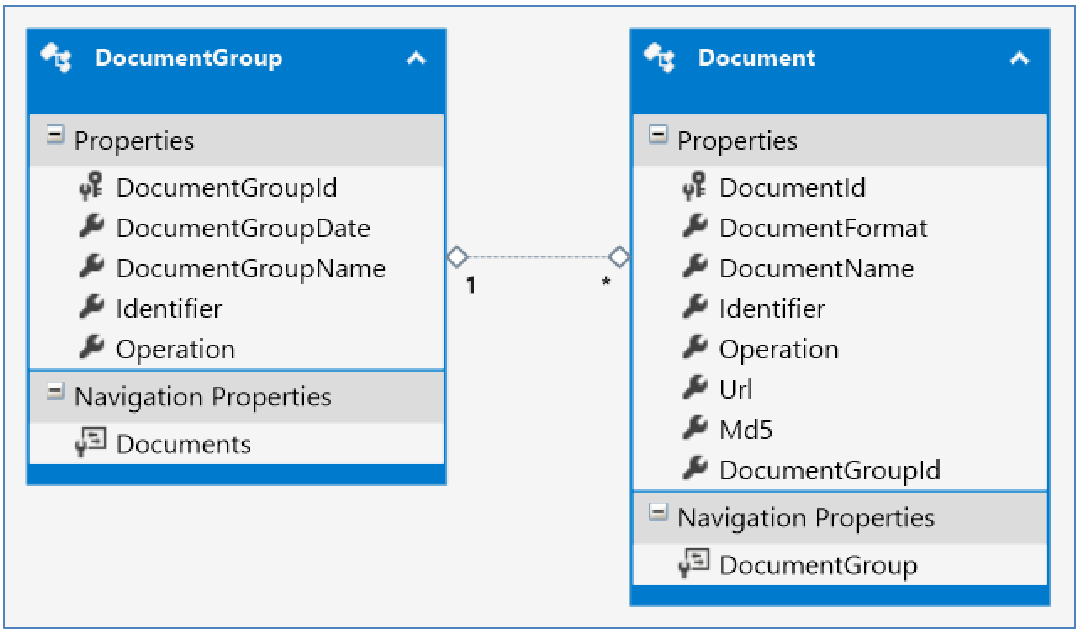
```
<system.serviceModel>
    <bindings>
      <basicHttpBinding>
        <binding name="BasicHttpBinding_IServiceEUCases" closeTimeout="00:10:00"
          openTimeout="00:10:00" receiveTimeout="00:10:00" sendTimeout="00:10:00"
          allowCookies="false" bypassProxyOnLocal="false" maxBufferPoolSize="2147483647"
          maxReceivedMessageSize="2147483647" useDefaultWebProxy="true"
          messageEncoding="Mtom" />
      </basicHttpBinding>
    </bindings>
    <client>
      <endpoint address="http://techno.eucases.eu/CrawlerService/ServiceEUCases.svc"
        binding="basicHttpBinding" bindingConfiguration="BasicHttpBinding_IServiceEUCases"
        contract="ServiceEUCasesReference.IServiceEUCases" name="BasicHttpBinding_IServi
ceEUCases" />
    </client>
  </system.serviceModel>
```

SQLite Database structure

Meta information for all crawled documents is stored into two tables:

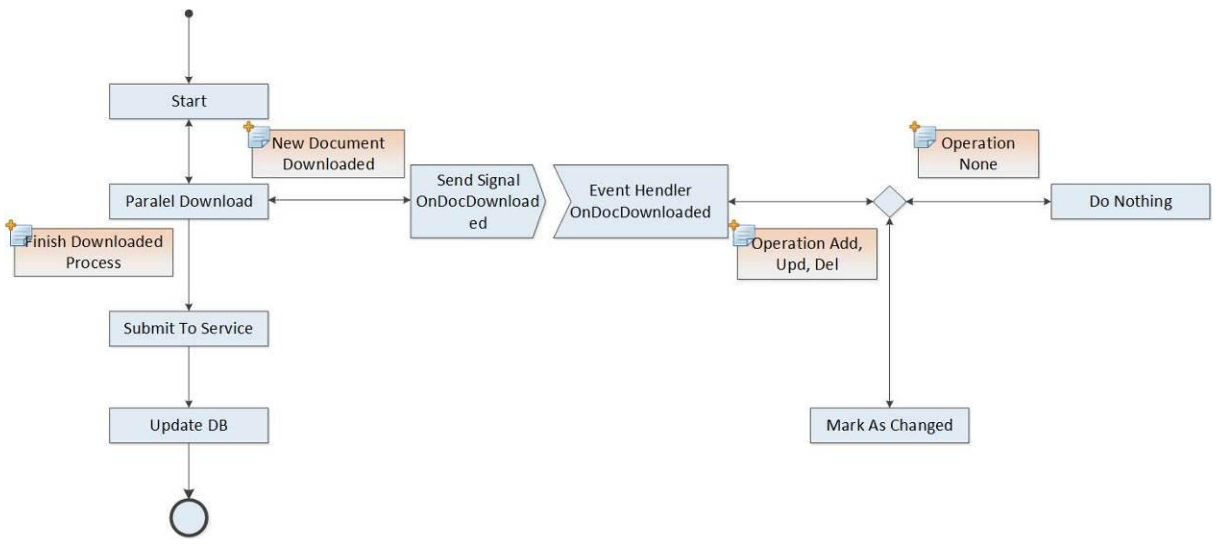
- ***DocumentGroup*** – contain information for all groups of documents
- ***Document*** – contain information for all documents.

Figure 4 – DocumentGroup and Document tables



The figure below shows visually the process of downloading and submitting data to the Crawler web service.

Figure 5 – Process of downloading and submitting legal data



Description of the common types (interfaces, classes, enumerations)

ICrawler Interface

This interface defines methods and events which all crawlers must support.

Public Member Functions

- *Start ()* – Starts the crawling process.

Events

- *OnDocDownloaded* – Each crawler must fire this event when the document is downloaded completely.

BaseCrawler Class

This is the base class inherited by all crawlers. It implements the event firing functionality.

Public Member Functions

- *Start ()* – Starts the crawling process
- *GetDataFormat (string* link*)* – Gets well formatted Uri/Url and returns the format of the document
- *GetFileName (string* path*)* – Gets the file path and returns the file name without the extension.

Protected Member Functions

- *DoOnDocDownloaded (list <DocumentMetaInfo> document, string folder)* – This method is called for every downloaded document from extenders.

Properties

- *OnDocDownloaded* – Implements the event handler *OnDocDownloaded* of the *ICrawler Interface.*

CrawlerManager Class

This class is responsible for the whole crawling process and for the submission of downloaded documents to the Crawler web service.

Public Member Functions

- *CrawlerManager ()* – Initialises a new instance of the CrawlerManager class
- *Start (BaseCrawler* crawlerClass*)* – Starts the process of checking for differences for each downloaded document. Saves and sends the document to the Crawler web service if there is a difference.

CrawlerLog Class

This class provides functionality for logging exceptions and information into a log file.

Static Public Member Functions

- *LogException (exception* exc*, string* url*)* – writes url with exception along with the innerexception layer into the log file
- *LogInfo (string* msg*)* – writes a message into the log file
- *LogException (exception* exc*)* – writes an exception along with the innerexception layer into the log file.

Properties

- *LogDir* – The logging directory
- *InfoLog* – The name of the log file.

Operation Enum

This enumeration describes the type of operation for the crawled document.

Enumerator

- *Add* – The document is new
- *Upd* – The document is updated
- *Del* – The document is deleted
- *None* – There is no change in the document.

DocumentGroup Class

This class provides information for a single document group. Since there are documents that are composed of group of files, they need to be combined in a document group in order to keep their consistency. To ensure this, each document must be put in a DocumentGroup object (even the document is composed by single document).

Public Member Functions

- *DocumentGroup ()* – Constructor of the class.

Properties

- *<DocumentMetaInfo> Document* – List of DocumentMetaInfo objects
- *Date* – String representation of the date and time when the document group has been downloaded
- *Crawler* – String identifier of the crawler that is downloading the documents
- *Lang* – Language of the document group
- *Format* – File type of the document

- *FileName* – File name of the document group
- *Identifier* – Unique identifier (GUID) of the document group
- *Operation* – Type of the operation for this document group
- *GroupType* – Type of the document (1 – Legislation, 2 – Case Law).

DocumentMetaInfo Class

This class provides information for a single document.

Properties

- *DataContent* – Binary representation of the crawled document
- *Format* – File type of the document
- *File* – File name of the document
- *Identifier* – Unique identifier (GUID) of the document
- *Operation* – Type of the operation for this document
- *Url* – The url of the crawled document
- *Md5* – The MD5 hash sum of the document.

ServiceContractor Class

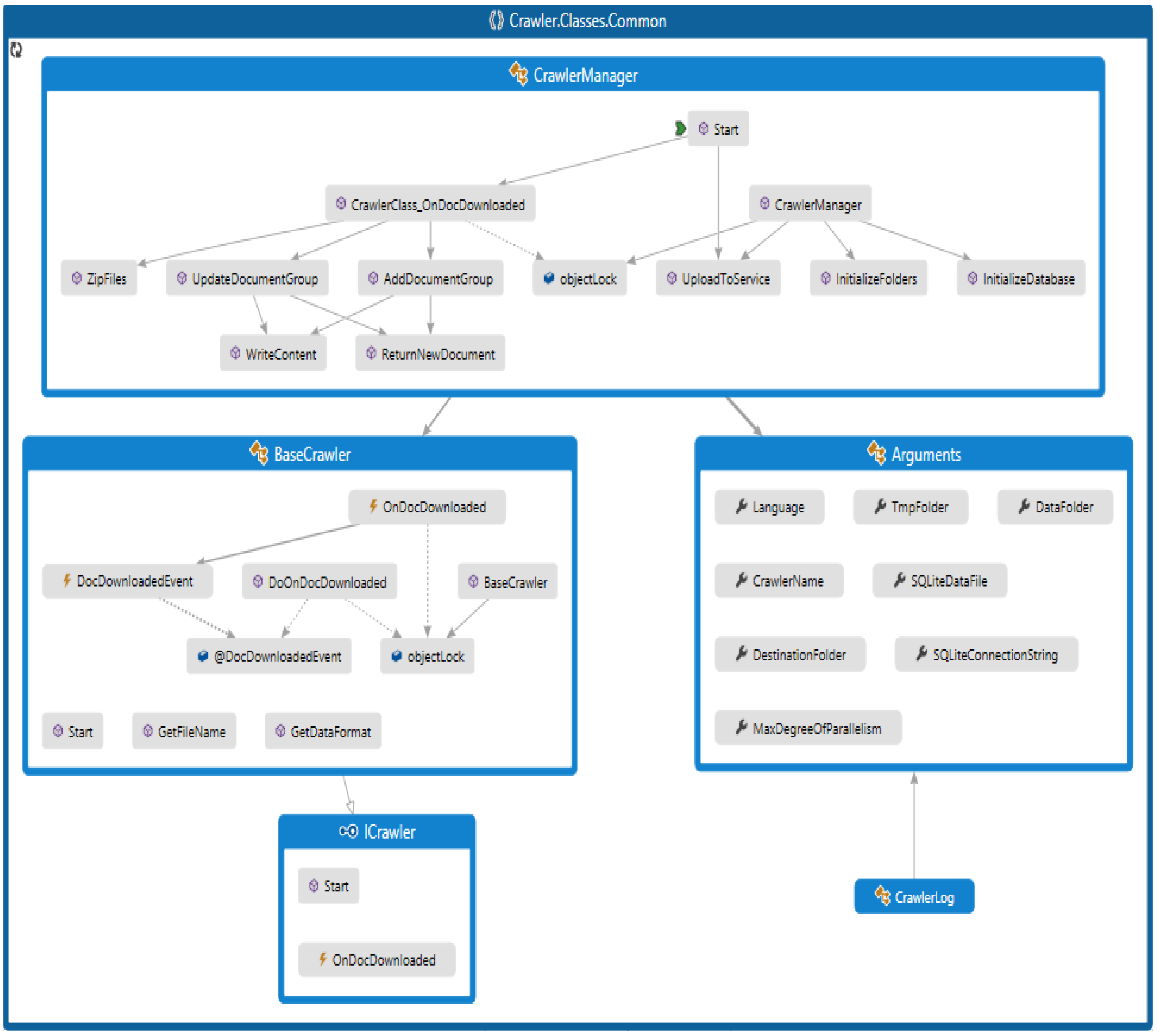
This class provides functionality for submission of the crawled document to the Crawler web service.

Public Member Functions

- *ServiceContractor* – Initialises the class
- *UploadToService (DocumentGroup* file*)* – Serialises information for the crawled document to XML and submits the document and the XML to the Crawler web service.

The following figure shows the hierarchy of the common classes and the connection between them.

Figure 6 – Crawler Common Classes



3.3.2 UNITO's crawlers

UNITO is developing the crawlers for downloading legislation and case law from the legal portals in Italy getting data from three specific sites: www.cortecostituzionale.it, www.normattiva.it and www.giustizia-amministrativa.it.

The study of the legal portals has shown that often legal materials are published as compound documents consisting of two or more files, e.g. main document with annexes or images. In order to preserve the structure of such compound documents, each crawler must take care to combine the files composing the document in a zip group (package), which contains an XML description in the following format:

```
<documentgroup …>
        <document …>
        ……………..
        </document>
</documentgroup>
```

The ‘documentgroup’ element describes the information about the entire group of files constituting the compound document, and the ‘document’ element contains information about each file.

The attributes of the ‘documentgroup’ and of the ‘document’ element have been listed above (see also the example of the XML description).

Information for each document group is preserved in the following way:

- All files are archived DocumentGroup in Zip format and saved to the file system

- The metadata for the DocumentGroup is stored in a local repository.

It is important to note that downloaded data may fluctuate (e.g. newly published court decisions have been downloaded, consolidated version of a legal act has been modified, etc.) and each of the crawlers should take care to their frequent updates. This is performed by periodically downloading the data and checking for changes in any of the downloaded files.

Crawler download phases

- **Initial download** – During the initial download of data the crawlers generate MD5 hash sum for each downloaded file and this information is saved in the database. All files are marked as new (attribute ‘operation’ of the ‘document’ element) and are submitted to the Crawler web service for storing and processing.
- **Subsequent download** – In each subsequent download of data for each downloaded file crawlers generate MD5 sum again and check if the file is new or has been modified. If the MD5 sum is different, this information will be reflected in the XML description (attribute ‘operation’ of the ‘document’ element) and the file is sent to the Crawler web service for storing and processing.

Crawler configuration

There are three crawlers developed by UNITO. They share the same architecture. For their development Microsoft Visual Studio 2013 has been used. Depending on the configuration file different period-related documents are downloaded. Crawlers read configuration from a file configuration.

Unito.EUCases.Crawlers.CorteCostituzionale

- **Main configuration**
  - StartYear: year from which the search is starting
  - EndYear: year to which the search is over
  - URL: the service where the app is pointing at

Unito.EUCases.Crawlers.Normattiva

- **Main configuration**
  - StartYear: year from which the search is starting
  - StartMonth: month from which the search is starting
  - EndYear: year to which the search is over
  - EndMonth: month to which the search is over
  - URL: the service where the app is pointing at

Unito.EUCases.Crawlers.GiustiziaAmministrativa

- **Main configuration**

  - StartYear: year from which the search is starting
  - EndYear: year to which the search is over
  - URL: the service where the app is pointing at

Uploader configuration

There is only one component responsible for uploading files.

Unito.EUCases.CrawlersUploader

- **Main configuration**
  - EucasesServiceURL: url of webservice Eucases
  - PathAddFiles: local work folder where to put the files in add operation
  - PathUpdateFiles: local work folder where to put the files in update operation
  - PathLocalRepository: the path of local repository
  - OutputFolder: folder of zipped files
  - SentFolder: folder of sent zipped files and metadata

Worker configuration

This is a sample for the configuration file for Normattiva's worker where one can set the crawler parameters:

- Start year
- End Year
- Web site URL

and the uploader parameters:

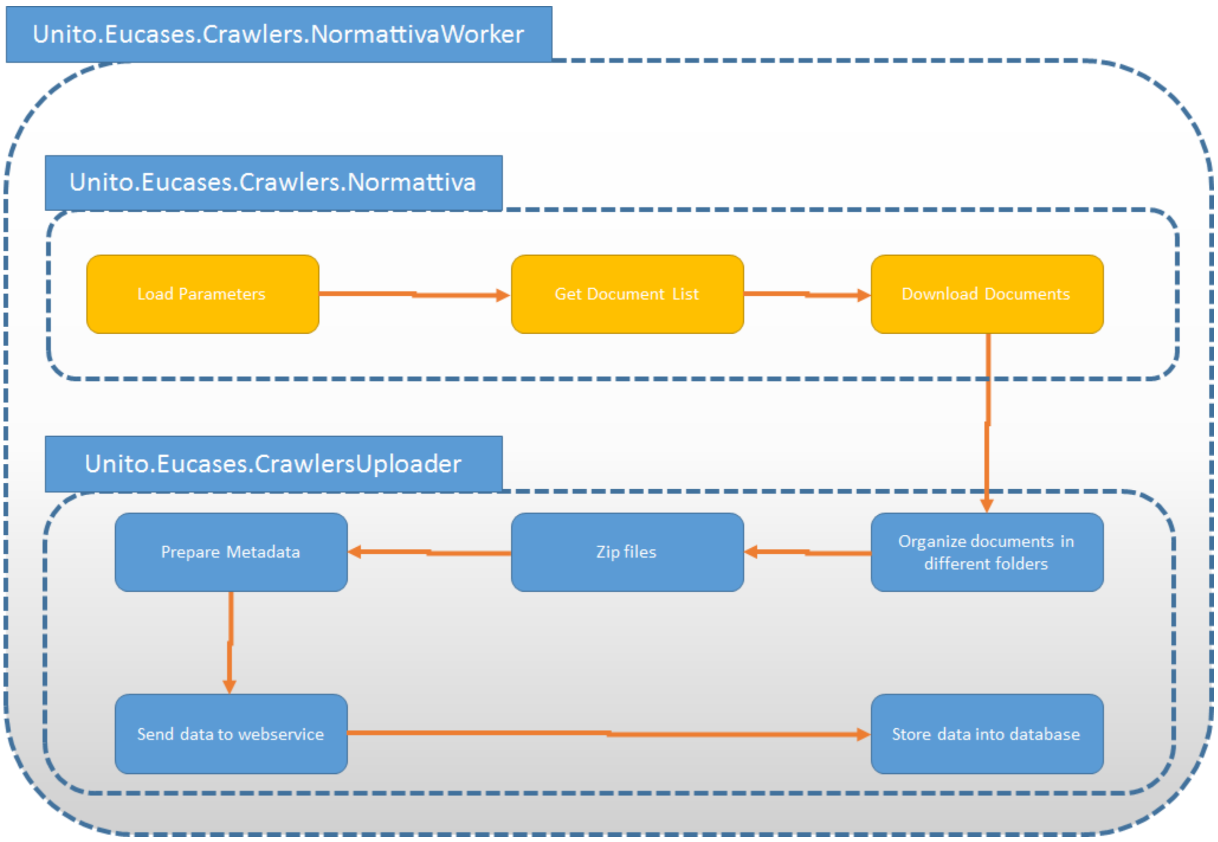
- EUCasesServiceURL:Eucases service url
- PathAddFiles:Working folder for the files in "add"
- PathUpdateFiles:Working folder for the files in "update"
- PathLocalRepository: working folder for local repository
- Outputfolder: working folder for zipped files
- SentFolder: archiving folder for sent items

```
<?xml version="1.0" encoding="utf-8"?>
<CorteCostituzionaleParameters xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <CrawlerParameters>
    <EndYear>2001</EndYear>
    <StartYear>2000</StartYear>
    <URL>http://www.cortecostituzionale.it/actionPronuncia.do</URL>
  </CrawlerParameters>
  <UploaderParameters>
    <EUCasesServiceURL>C:\downloads\CorteCostituzionale</EUCasesServiceURL>
    <PathAddFiles>C:\downloads\CorteCostituzionale\AddFiles</PathAddFiles>
<PathLocalRepository>C:\downloads\CorteCostituzionale\LocalRepository</PathLocalRepository
>
    <PathUpdateFiles>C:\downloads\CorteCostituzionale\UpdateFiles</PathUpdateFiles>
    <OutputFolder>C:\downloads\CorteCostituzionale\OutputFiles</OutputFolder>
    <SentFolder>C:\downloads\CorteCostituzionale\SentFiles</SentFolder>
  </UploaderParameters>
  <DestinationFolder>C:\downloads\CorteCostituzionale</DestinationFolder>
</CorteCostituzionaleParameters>
```

Architecture of UNITO's crawlers

The schema is representing the architecture of the Normattiva Crawler. The same concepts are valid also for the other crawlers of UNITO.

Figure 7 – Architecture of the Normattiva Crawler


Unito.Eucases.Crawlers.NormattivaWorker
Unito.Eucases.Crawlers.Normattiva
Load Parameters
Get Document List
Download Documents
Unito.Eucases.CrawlersUploader
Prepare Metadata
Zip files
Organize documents in different folders
Send data to webservice
Store data into database

Unito.Eucases.Crawlers.NormattivaWorker

This component is responsible for storing file configuration, managing and tracing the activities of:

- Unito.Eucases.Crawlers.Normattiva
- Unito.Eucases.CrawlersUploader

Unito.Eucases.Crawlers.Normattiva

This component is responsible for the download of an array of documents depending on range requested using configuration parameters.

Unito.Eucases.Crawlers.CorteCostituzionaleWorker

This component is responsible for storing file configuration, managing and tracing the activities of:

- Unito.Eucases.Crawlers.CorteCostituzionale
- Unito.Eucases.CrawlersUploader

Unito.Eucases.Crawlers.CorteCostituzionale

This component is responsible for the download of an array of documents depending on range requested using configuration parameters.

Unito.Eucases.Crawlers.GiustiziaAmministrativaWorker

This component is responsible for storing file configuration, managing and tracing the activities of:

- Unito.Eucases.Crawlers.GiustiziaAmministrativa
- Unito.Eucases.CrawlersUploader

Unito.Eucases.Crawlers.GiustiziaAmministrava

This component is responsible for the download of an array of documents depending on range requested using configuration parameters.

Unito.Eucases.CrawlersUploader

This component is responsible for storing in a local database of the metadata of the files and for uploading the documents to the web service Eucases.

Its primary mission is to create the metadata and the zip files, storing MD5 hash file so it can be reused for successive upload.

The component is common for all three crawlers (Normattiva, CorteCostituzionale and Giustizia-Amministrativa).

Crawler Interface

ICrawler<P,R> Interface

Public Member Functions

- IEnumerable<IDownloadItem> GetDownloadList();

This interface ask to programmer to define a list of document to download

- IDownloadResult<R> Download(IDownloadItem item)

This function ask to programmer to define a function Download getting as input parameter the above calculated array of object inheriting the IDownloadItem interface

IDownloadItem Interface

It is a very simple interface where the developer is asked to create a class to store:

- the URL
- an array of navigation path
- an ID
- an array of post values used to get the desired item.

IDownloadResult<R> Interface

This interface has no method. It simply defines a record for associating two objects: the one specifying the parameters of the request sent to the website and the one specifying the content sent from the website to the crawler.

- IDownloadItem Request
- R Content

R is generic type that is specifically defined for each crawled website.

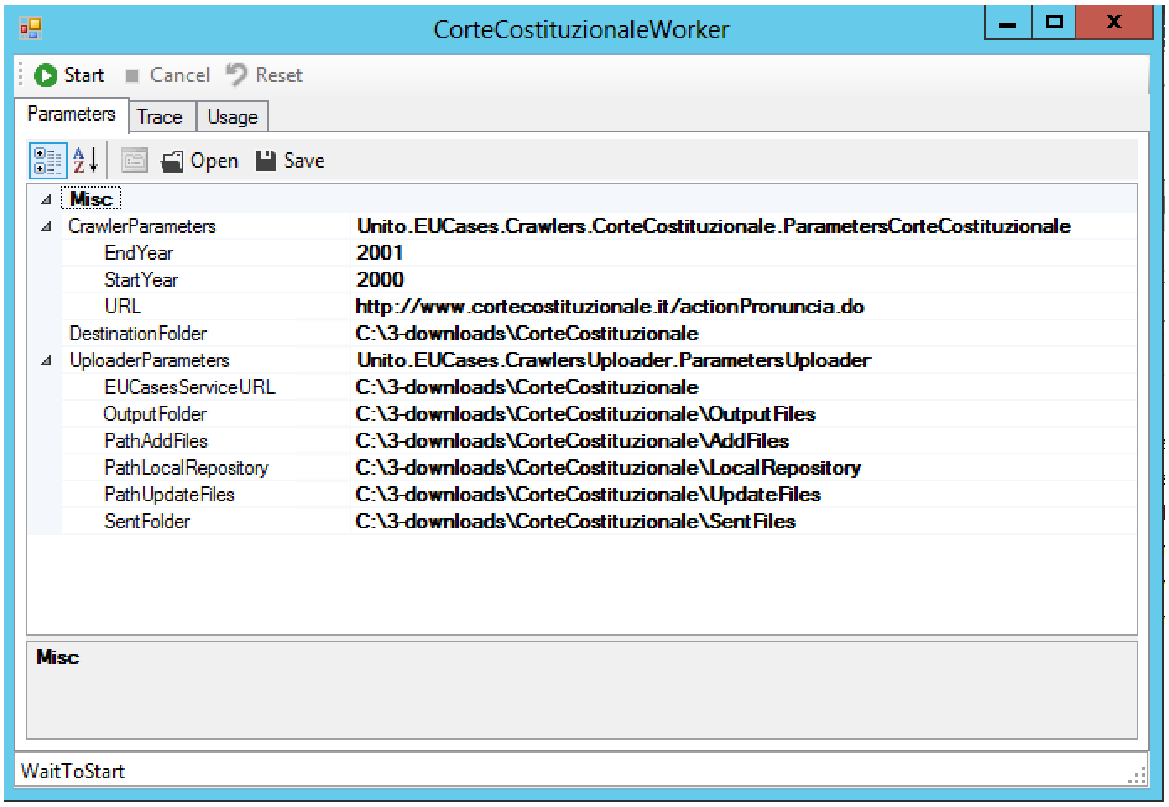
Worker Interface

WorkerBase<P,R>

The WorkerBase class permits to define parameters to enter and provides results back from the functions implemented in the Worker.

It permits an automatic management of parameters by the User Interface or by the shell by only adding attributes to the class implemented as parameters.

Figure 8 – User interface of the CorteCostituzionaleWorker

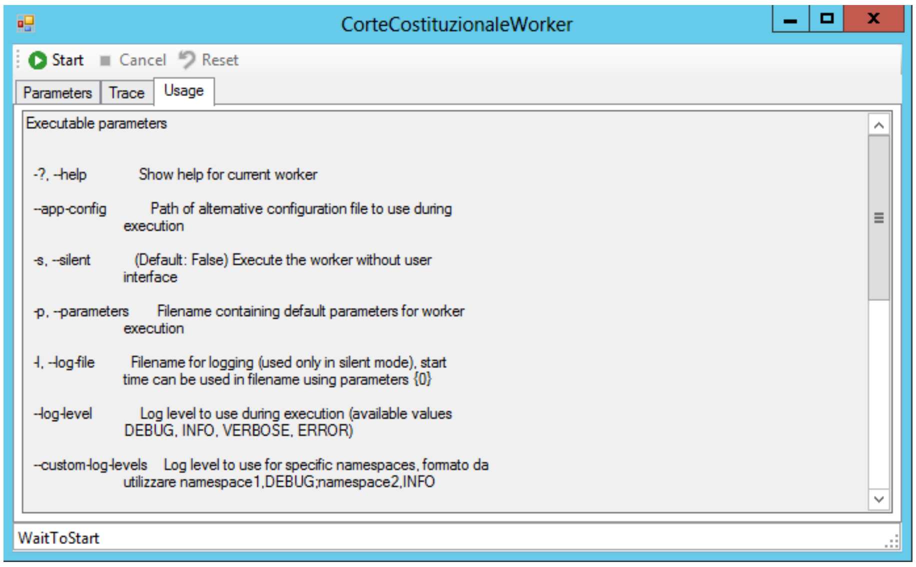


Misc	
CrawlerParameters	Unito.EUCases.Crawlers.CorteCostituzionale.ParametersCorteCostituzionale
EndYear	2001
StartYear	2000
URL	http://www.cortecostituzionale.it/actionPronuncia.do
DestinationFolder	C:\3-downloads\CorteCostituzionale
UploaderParameters	Unito.EUCases.CrawlersUploader.ParametersUploader
EUCasesServiceURL	C:\3-downloads\CorteCostituzionale
OutputFolder	C:\3-downloads\CorteCostituzionale\OutputFiles
PathAddFiles	C:\3-downloads\CorteCostituzionale\AddFiles
PathLocalRepository	C:\3-downloads\CorteCostituzionale\LocalRepository
PathUpdateFiles	C:\3-downloads\CorteCostituzionale\UpdateFiles
SentFolder	C:\3-downloads\CorteCostituzionale\SentFiles

Misc

WaitToStart

It is possible to execute the program from the command shell. The available parameters are listed in the user interface as shown below or the user can ask directly for them from the command shell by typing <name of crawler> and “-?” Or “-Help”.



```
Executable parameters


-?, --help          Show help for current worker

--app-config          Path of alternative configuration file to use during
                    execution

-s, --silent          (Default: False) Execute the worker without user
                    interface

-p, --parameters      Filename containing default parameters for worker
                    execution

-l, --log-file          Filename for logging (used only in silent mode), start
                    time can be used in filename using parameters {0}

--log-level          Log level to use during execution (available values
                    DEBUG, INFO, VERBOSE, ERROR)

--custom-log-levels    Log level to use for specific namespaces, formato da
                    utilizzare namespace1,DEBUG;namespace2,INFO
```

WaitToStart

3.3.3 AVERBIS’ crawlers

The crawlers developed by Averbis are based on Apache Nutch. Nutch is an extensible crawler framework built to run reliable crawls on web-scale by distributed execution on Apache Hadoop. For smaller jobs like intranet crawls there exists a local execution mode without the requirement of setting up Hadoop.

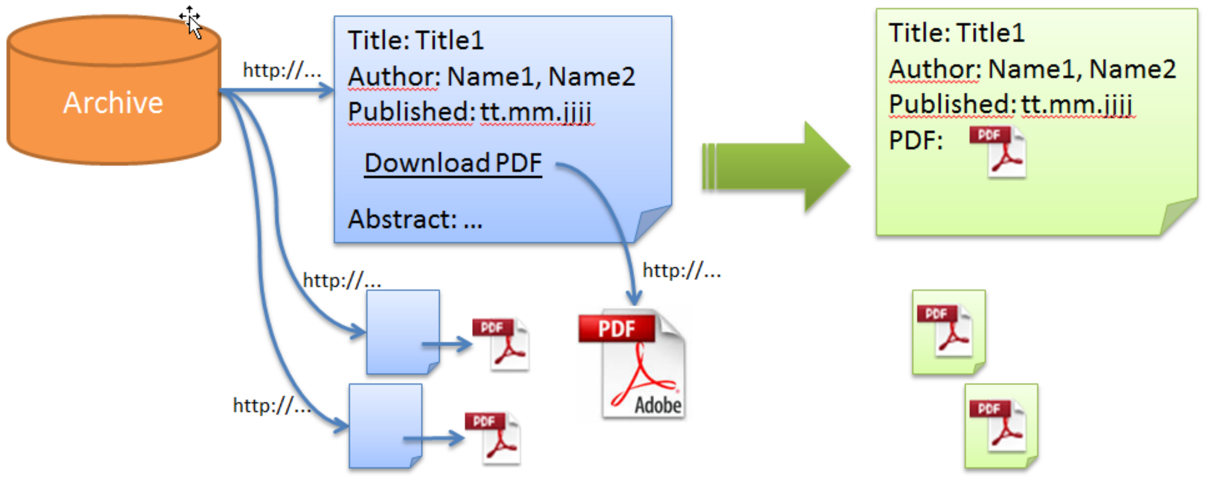
As important as the scalability is the extensibility of Nutch because is not possible for any framework developer to foresee every use case the framework will be applied to. Consequently, the core of Nutch consists mostly of abstract classes and interfaces that provide extension points for additional modules and very generic crawler functionality but even basic components such as HTTP network access and HTML parsing are yet contained in plugins.

After evaluating the requirements for crawling Open Access Journals it has become clear that Nutch offers sufficient flexibility to implement the necessary custom procedures while other components that are independent from the specific requirements can be reused. Finally, Averbis has developed a plugin for the collection of metadata about the articles such as title and authors and a custom indexer plugin, and made minor changes to a third-party plugin for extracting content from a web site.

1. Crawling Open Access Journals

The goal of crawling an Open Access Journal is obtaining each of its articles (full text, mostly in binary from as PDF) together with metadata about it. The entry point to start from is the journal’s archive where a list of all issues (rarely of all articles) can be found. Similarly, each issue provides its contents with links to abstracts of each article (see Fig. 9).

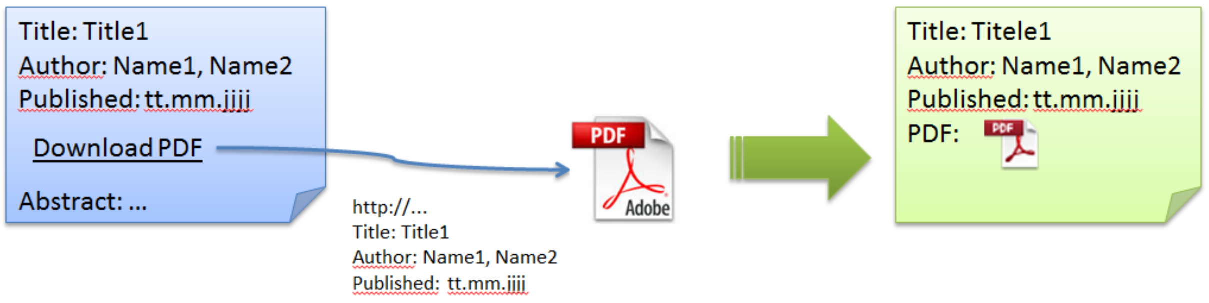
Figure 9 – Archive content should be indexed as bundled documents



When the crawler comes to an abstract page it should collect relevant information about the article from it. This information should be stored together with the full text of the article. In order to avoid post-processing the downloaded data after finishing the crawl for assembling the metadata from the abstract and the full text into a single document that can be sent to the Crawler web service, the Averbis crawler annotates the link to the article’s full text with the metadata from the abstract.

Hence, when the full text is retrieved the metadata are already available can be sent directly together with the text to the web service. The proceeding is sketched in Fig. 10.

Figure 10 – The documents can be bundled and indexed on the fly by annotating the outlinks



2. Plugin outlinkmeta

The functionality for annotating the outlinks is contained in a plugin named outlinkmeta. Essentially, it is a parser filter and is invoked after other parsing of a fetched page is done. The plugin is generic and the list of metadata keys whose values should be added to an outlink can be configured. The metadata values are retrieved from the parsing metadata. During the page's preparation for indexing the plugin attaches the metadata to the document that is send to the indexer.

3. Plugin eucases-indexer

Another custom plugin is needed for sending the bundled documents to the crawler web service. The eucases-indexer plugin implements the IndexWriter interface. Thus it receives all documents from the previous crawler steps. It checks whether a document contains all required information and should be sent to the web service. Unlike the outlinkmeta plugin, this plugin is very specific to the current task. The only configuration option is the address of the web service.

4. Plugin extractor

The extractor plugin is available from https://github.com/BayanGroup/nutch-custom-search. The crawler uses it to extract the information about the articles from the abstract pages after other parsing has completed. The data elements can be localised by CSS-rules or XPATH-expressions. Additionally, some simple functions (e.g., replace) can modify the extracted data. The metadata entries are added to the parsing metadata.

For the crawler, the plugin's processing of multi-valued metadata entries was slightly modified.

4 Publication of source code and software documentation

4.1 Authorship and OSS license

The crawler framework and tools have been developed as open source software by members of the EUCases consortium as part of their work on the EUCases project supported under the 7th Framework Program of the European Commission. Author of the code and documentation of the crawler tools for Italian legal portals is Prof. Guido Boella, of the University of Turin, the code and documentation of the crawler tools for web sites of selected open access journals in the field of law have been developed by Averbis GmbH, and those for all other legal portals relevant to the goals of the EUCases project – by Apis Hristovich EOOD. The source code and the software of the crawler framework and tools developed under the EUCases project are made publicly available under the European Union Public Licence (EUPL, v.1.1).[5]

Unless required by applicable law, software distributed under the license is distributed on an "AS IS" basis, without warranties or conditions of any kind, either express or implied. More specifically, the licensor does not provide any warranty that crawler tools will download open legal data properly in the case of subsequent changes of the structure of the respective web sites. See the EUPL for the specific language governing permissions and limitations under the license.

4.2 Disclaimer of liability for infringement of copyright or database right on the crawled data

Licensees can exercise all rights granted under the EUPL only with respect to the use of the software (source code, documentation and executable code) of the crawler tools developed by members of the EUCases consortium. Neither this license nor any document or deliverable made publicly available by the EUCases consortium can be understood as entitlement for re-use of any data and/or documents which the crawler tools are capable to download from public legal portals and/or web sites of open access journals. The EUCases consortium or any of its members will in no event be liable for infringement of copyright or database right by a licensee occurred in a consequence of the use of the software of the crawler tools. The licensee is solely responsible for checking on his own the conditions for re-use of data for each public legal portal or web site of an open access journal he intends to crawl and will be solely liable for any breach of the legal terms for data re-use.

4.3 Source code: build instructions and software documentation

The source code of the crawler tools with build instructions and documentation is published on the EUCases project web site:

http://eucases.eu/fileadmin/EUCases/documents/EUCases_CrawlerFramework&Tools_SourceCode.zip,

and on GitHub:

https://github.com/EUCASES-UNITO/Crawlers.

[5] https://joinup.ec.europa.eu/software/page/eupl/licence-eupl.

Appendix I: XSD schema for document metadata validation

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema          attributeFormDefault="unqualified"          elementFormDefault="qualified"
xmlns="xsdDocumentgroup"                                  targetNamespace="xsdDocumentgroup"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="documentgroup">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="1" maxOccurs="unbounded" name="document">
          <xs:complexType>
            <xs:attribute name="format" use="required">
              <xs:simpleType>
                <xs:restriction base="xs:string">
                  <xs:enumeration value="application/zip" />
                  <xs:enumeration value="application/gzip" />
                  <xs:enumeration value="application/pdf" />
                  <xs:enumeration value="image/gif" />
                  <xs:enumeration value="image/jpeg" />
                  <xs:enumeration value="image/png" />
                  <xs:enumeration value="image/tiff" />
                  <xs:enumeration value="image/bmp" />
                  <xs:enumeration value="text/css" />
                  <xs:enumeration value="text/html" />
                  <xs:enumeration value="text/htm" />
                  <xs:enumeration value="text/javascript" />
                  <xs:enumeration value="text/xml" />
                </xs:restriction>
              </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="file" type="xs:string" use="required" />
            <xs:attribute name="identifier" use="required">
              <xs:simpleType >
                <xs:restriction base="xs:string">
                  <xs:pattern     value="[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-
fA-F0-9]{12}"/>
                </xs:restriction>
              </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="operation" use="required">
              <xs:simpleType >
                <xs:restriction base="xs:string">
                  <xs:enumeration value="Add" />
                  <xs:enumeration value="Upd" />
                  <xs:enumeration value="Del" />
                  <xs:enumeration value="None" />
                </xs:restriction>
              </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="url" type="xs:string" use="required"/>
            <xs:attribute name="md5" use="required" >
              <xs:simpleType >
```

```
                <xs:restriction base="xs:string">
                  <xs:pattern value="[a-fA-F0-9]{32}"/>
                </xs:restriction>
              </xs:simpleType>
            </xs:attribute>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="date" type="xs:dateTime" use="required" />
      <xs:attribute name="crawler" type="xs:string" use="required" />
      <xs:attribute name="lang" type="xs:string" use="required" />
      <xs:attribute name="format" use="required">
        <xs:simpleType>
          <xs:restriction base="xs:string">
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
      <xs:attribute name="filename" type="xs:string" use="required" />
      <xs:attribute name="identifier" use="required" >
        <xs:simpleType >
          <xs:restriction base="xs:string">
            <xs:pattern value="[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-
9]{12}"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
      <xs:attribute name="operation" use="required">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="Add" />
            <xs:enumeration value="Upd" />
            <xs:enumeration value="Del" />
            <xs:enumeration value="None" />
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:complexType>
  </xs:element>
</xs:schema>
```