OJS Software Workshop Report

Alexander Bigga[1], Martin Brändle[2], Ronny Bölter[3], Constanze Decker[4], Maria España[5], Gerhard Eilbacher[3], Matthias Flasko[7], Frank Krabbes[8], August H. Leugers-Scherzberg[9], Marianna Mühlhölzer[10], Dennis Müller[11], Philip Münch[12], Tom Niers[13], Daniel Nüst[13], Adrian Pachzelt,[15], Maximilian Plich[16], Ronald Steffen[17], Klaus Thoden[18], Hanna Varachkina[19], Paul Warner[20], Nils Weiher[22], Daniela Wolf[22], and Dulip Withanage[23]

[1]Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden (SLUB), alexander.bigga@slub-dresden.de
[2]Zentrale Informatik, Universität Zürich, martin.braendle@uzh.ch
[3]Leibniz Institute for Psychology Information, Trier, {RB,GeE}@Leibniz-Psychology.org
[11]Universitätsbibliothek Mannheim, dennis.mueller@bib.uni-mannheim.de
[12]Universitätsbibliothek Osnabrück, philip.muench@ub.uni-osnabrueck.de
[13]Institute for Geoinformatics, University of Münster, {t_nier01,daniel.nuest}@uni-muenster.de
[15]Universitätsbibliothek J. C. Senckenberg, a.pachzelt@ub.uni-frankfurt.de
[18]Edition Open Access/Max Planck Institute for the History of Science
[19]varachkina@sub.uni-goettingen.de
[22]Heidelberg
[23]withanage@ub.uni-heidelberg.de

April 22, 2020

Abstract

This report summarizes the achievements of the OJS community members from Germany and Switzerland in the OJS Workshop in Heidelberg University Library, Germany from February 20 and 21, 2020. Main goal of the workshop was to share knowledge and challenges, conceptualize and document problem solving suggestions and collectively develop software in and around OJS. Participants worked on a variety of subjects including data import/export plugins, search functionality, containerization, long-time archiving and XML workflows in and around OJS and OMP.

The workshop is a continuation of fruitful meetings within the German OJS user and developer community under auspices of (OJS-de.net network).

Contents

1 Import/Export-Plugin **4**
- 1.1 Current Situation . . . 4
- 1.2 Open Issues . . . 4
- 1.3 Use cases . . . 4
- 1.4 Goal . . . 5
- 1.5 What needs to be done . . . 5
- 1.6 Problems we see . . . 5
- 1.7 Results . . . 5

2 Search (Elastic-Search / Solr / Lucene) **6**
- 2.1 Current Situation . . . 6
- 2.2 Goal . . . 6
- 2.3 Results . . . 6

3 Plugin for connecting external Long-term archiving systems (Rosetta, Archivematica, Dataverse, DSpace/Sword, ...) **7**
- 3.1 Goal . . . 7
- 3.2 Questions to be answered . . . 7
- 3.3 Requirements . . . 8
- 3.4 Workflow . . . 9

4 Dockerbased development environments and automisation **11**
- 4.1 Goal . . . 11
- 4.2 Tasks . . . 11
- 4.3 Results . . . 12

5 XML-Workflow **13**
- 5.1 Goal . . . 13
- 5.2 Tool evaluation . . . 14
- 5.3 Example conversions . . . 15
- 5.4 Final evaluation . . . 16
- 5.5 Next steps . . . 16

1 Import/Export-Plugin

Philip Münch, Alexander Bigga, Mariana España

1.1 Current Situation

- Native XML plugin can only export/import issues/articles, and only metadata and galley files, not workflow uploads, messages or user accounts.
- There is a user account import/export plugin.
- There is no way to import/export journal settings.
- Import/Export plugins are callable from the command line, so that automatic backups can be done this way.
- The new REST API provides lists of journals, issues and articles

1.2 Open Issues

- 19.02.2020 (!) "option to reference files in export xml" (`https://github.com/pkp/pkp-lib/pull/5524`)
  - On export the files can be excluded from embedding in XML. The files are referenced by href instead.
- 11.01.2018: Extend native import/export plugin to include additional entities (`https://github.com/pkp/pkp-lib/issues/3261`)
  - Open PRs: OJS (`https://github.com/pkp/ojs/pull/1978`), PKP-Lib (`https://github.com/pkp/pkp-lib/pull/3722`)
  - For OJS 2.x there has been an `https://github.com/lepidus/fullJournalTransfer`) plugin.
- `https://github.com/pkp/pkp-lib/issues/5067` (OJS2 → OJS3 via XML exports)
- `https://github.com/pkp/pkp-lib/issues/4880` (native XML plugin needs to be changed to support versioning)

1.3 Use cases

- Migrating between instances
  - Preparatory and productive instance
  - Migrating a journal to a different service provider
- Backing up journals independently on multi-journal instances

1.4 Goal

- We need a way to export a full journal.
- The task could be divided into multiple steps:
  - Exporting journal settings (There is no way to do this at present)
  - Exporting users (This can already be done)
  - Exporting all issues (The Native XML Plugin already does part of this)
- Since the size of a single export file in XML format with Base64-encoded files would be impractical for a full journal, this seems like the best solution.

1.5 What needs to be done

- The Native XML plugin needs to be extended to also export all interry workflow files and all messages (optionally).
- A new plugin has to be created to export journal settings. This should be callable from the command line as well for automatic backup purposes.

1.6 Problems we see

If all workflow results are to be imported again, there needs to be a way to link them back to the users that produced them. What happens if a user is not present?

We propose to separate between "people" records that are created per journal, and account records that are global. These can be linked and unlinked. We suspect this might also be a good idea for GDPR compliance.

1.7 Results

We decided to just ask for a current status on the main issue above, since some work has already been done that would fulfill our needs, but we are not able to discern how far this has come.

Regarding implementing a new plugin for importing/exporting journal settings, we will need more information on how such a plugin would be implemented and what provisions might already be in the code.

2 Search (Elastic-Search / Solr / Lucene)

Constanze Decker, Tom Niers, Nils Weiher

2.1 Current Situation

- Currently one plug-in is mainly used for searching - Lucene/Solr Plug-In
  - `https://github.com/ojsde/lucene/`
  - Developped by CEDIS, FU Berlin
  - Should be in the OJS Plug-In Gallery soon (further information by Ronald Steffen)
- Here is a more detailed article about the comparison between Solr and ElasticSearch: `https://logz.io/blog/solr-vs-elasticsearch/`

2.2 Goal

- Comparison of ElasticSearch and Solr
- Analysis of the current state of the Lucene plug-in
- Clarify which advantages and disadvantages the way using a search plug-in has compared to the API

2.3 Results

- For overall search and search in OMP and OJS together, there seems to be no existing solution. The University Library of Heidelberg is working on its own solution in this context.
- Rewriting the existing Lucene/Solr Plug-In to ElasticSearch would be possible, but would require a lot of effort, since the configuration and schema is designed specifically for Solr.
  For ElasticSearch this would have to be rewritten to a JSON structure and the ElasticSearch configuration would also have to be started from scratch.
- After exchanging information about the current situation on the topic of search, the working group also dealt with a general introduction to plug-in development, as members still had specific questions.

3 Plugin for connecting external Long-term archiving systems (Rosetta, Archivematica, Dataverse, DSpace/Sword, ...)

Gerhard Eilbacher, Ronny Bölter, Martin Brändle, Paul Warner

3.1 Goal

- Generic Plugin (Able to connect with different systems)
- Plugin enables User to choose which files to send from OJS to long-term Archives
- Plugin features functionality to validate file formats, integrity of content and if all required Metadata has been provided

3.2 Questions to be answered

- What must be archived on article level? Possibilities:
  - Only metadata and published galleys/supplements
  - Metadata, published galleys/supplements including all versions (OJS 3.2!)
  - All above, including reviewer reports
  - All above, including article history

These options should be configurable by journal manager
Think of long-term archiving as >10 years; with a digital archive it should be possible to reconstruct the history or development of an article/issue/journal

- Who can archive?
  - Journal manager
  - Special role within OJS?
  - Should not be given to roles such as author, reviewer, editor
- Packaging : What is in an archival package?
  - Individual article
  - Selected articles
  - Only published articles / all articles?
  - A complete issue
  - A complete journal
- What must be delivered?

  - Some archival software require either complete archival packages (e.g. Submission Information Package (SIP) including METS archival format description), others just individual metadata and files to be archived. In the latter case, the data stewart of the digital archive will create the package.

- Push or pull mechanism? (we prefer push)
  - Push: OJS delivers a package to archive
  - Pull or harvesting (oai): OJS prepares a download section with a archival "package", digital archive fetches it (e.g. manually or automatically). Disadvantages: Who is allowed to pull?

3.3 Requirements

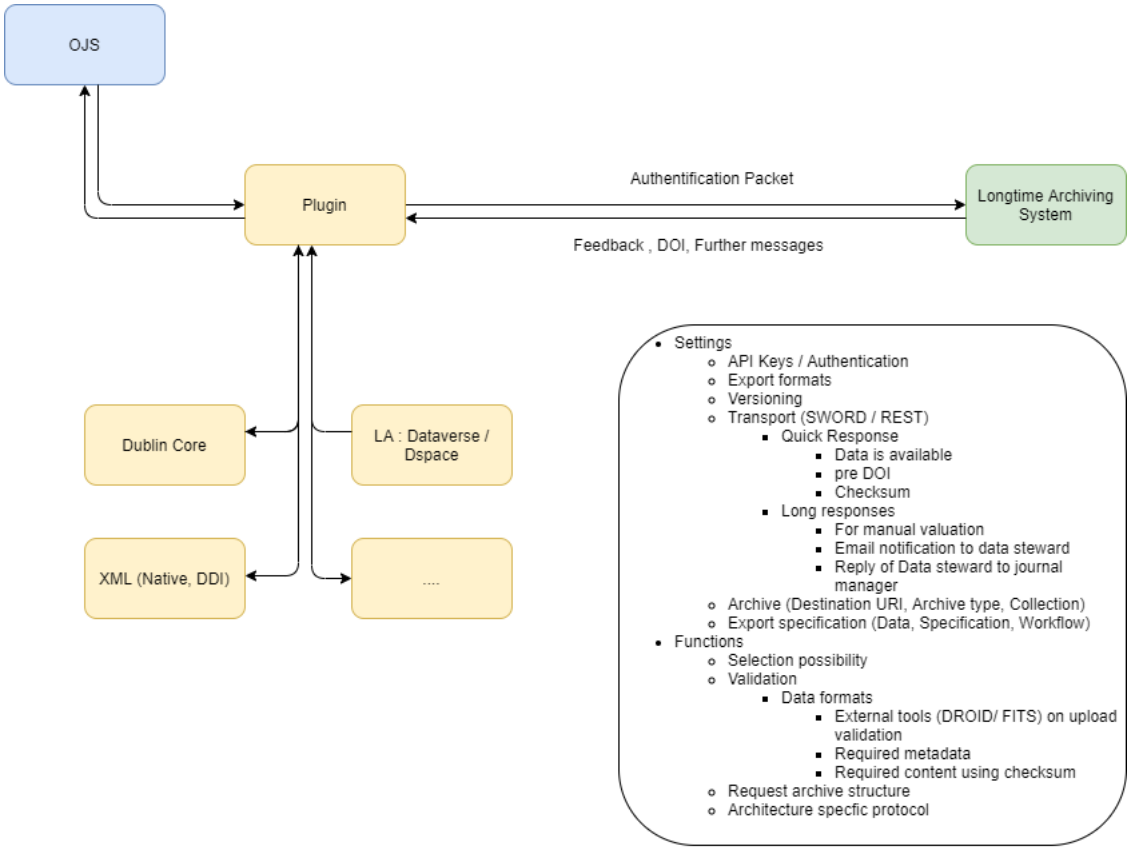

Figure 1: Architecture

3.4 Workflow

- The archival plugin must hook in at various points of the OJS workflow
  - Upload of files: Fixity (e.g. MD5 checksum) check, file format identification
  - Delivery of archival "package" (again fixity check should be made before)
- Delivery of package: Depending on digital archive and its configuration, there may be a non-interactive or a interactive workflows:
  - Non-interactive: Archive accepts package, validates it, and if valid puts it into archival storage without human interaction. A delivery receipt including acceptance status is sent back. Interaction is continuous, but may be async. Example for such archive softwares may be repositories such as DSpace or EPrints, or DLCM
  - Interactive: Package goes to a pre-ingest or ingest buffer, where it is validated by a data steward, and if valid, packaged (e.g. according to OAIS standards). Data steward sends back delivery receipt including acceptance status (e.g. archived, rejected). Communication is completely async, and there may be days or weeks between messages. Example for such archive softwares may be Archivematica or DLCM
- Upon delivery of a "package", plugin sends a notification message (e.g.) to the data steward of the archive

Dataflow
Modular setup

- Plugin communicates with OJS plugin for archival software (via pipe / internal API)
- OJS archival software plugins (DSpace, …) communicate with export format plugins for metadata format creation (e.g. DC, DataCite XML Schema, METS, …)
- Plugin has specified communication protocol with archival software (e.g. Sword/REST, …)
- Plugin may send metadata and files in a "loose" package or may send everything in a compressed SIP (e.g. using METS/PREMIS for metadata and format description and files)

GUI

- Settings

- File list: Provides both selection option and status (formats, metadata inputfield, error messages)
- Button for “package” delivery

Maintenance of the Plugin

- Sub-Plugins (for export) have to be kept up-to-date for new versions of archive software and backwards compatible
- General maintenance (e.g. in case of api changes in sub-plugins)

Datastructure

- Metadata of the article should be enhanced with
  - Status in workflow: Not archived, in review, archived, …
  - Checksums (for metadata, for each file) for fixity checks
  - File format identifier (e.g. PRONOM fmt)
- Plugin should have an internal list of valid file formats (e.g. a list of PRONOM format identifiers) for long term archiving (e.g. various PDF/A formats, text formats such as JSON, CSV, etc.)
- Plugin should have a knowledge about mandatory metadata for each archive software configured

4 Dockerbased development environments and automisation

Daniel Nüst, Dennis Müller, Dulip Withanage

4.1 Goal

Create a step-by-step tutorial to set-up a local development environment using Docker containers for plugins developers. The tutorial should include instructions for running proper debugging sessions with common IDEs.

Such a setup would be especially helpful for part-time developers who want to contribute minor enhancements to the complex OJS code base. This approach is an alternative to the common pattern of using git submodules (which not many developers are familiar with) or cloning the plug-in project into the plug-in folder, which is a common practise.

The runtime environment should be encapsulated in a way that the OJS code on the host computer can be mounted into an OJS environment. Version Change of Docker based OJS environments should be easily possible. Developer should be able to debug in preferred IDE (VSCode, Atom, WebStorm, etc.) without complex configurations.

Preferred user workflow may look like below.

1. ```
   docker run --rm -it --volume $(pwd)/my-ojs-plugin:/var/
       ojs/.../plugins:ro -p 8081:8081 -p 9000:9000 ojs-dev
       :3.1.2
   ```

2. Start debugging session in IDE

3. (if working on templates from within a plugin)

   ```
   docker exec ojs-dev ./clean_frontent_cache.sh
   ```

4.2 Tasks

1. Try out the current "official" Docker images
2. Write a `Dockerfile` enabling the development method sketched above
   (a) Xdebug
   (b) ports
   (c) Bind mounts for (plugin-)directories (as volumes in docker-compose.yml?)
3. Test workflow on group members computers
4. Write instructions
5. Open Pull Requests in `https://www.github.com/pkp/docker-ojs`
   ```

4.3 Results

We could not reach all the goals, unfortunately, but group members were able to greatly extend their knowledge and understanding of the current state of development of the official PKP OJS Docker images. After a detailed inspection of the repository `https://www.github.com/pkp/docker-ojs`, including the `docker-compose` configurations, the group split into maller units: DM installed Docker with the help of DN and worked on running OJS in containers locally; DN continued with the Docker files and configurations.

The group also contacted the main developer Marc Bria of the UAB, Barcelona. Marc Bria demonstrated the current status and helped in fixing issues of OJS Docker files and images issue #12) His input helped the group memebers better understood the currently working image variants (i.e., PHP7 and Apache). DN eventually contributed some changes back to the repository, including an update of the documentation, see `https://github.com/pkp/docker-ojs/pull/14`. MD successfully ran different OJS versions locally based on the container images.

To conclude, we think the official OJS Docker images are a great step towards easy testing and deployment of OJS. The potential for using them as a development environment is there, and has some benefits. The idea and the first coding attempts are preserved in issue #17 on the PKP Docker images project repository.

5 XML-Workflow

Adrian Pachzelt, Klaus Thoden, Marianna Mühlhölzer, Hanna Varachkina, Ronald Steffen, Matthias Flasko, August H. Leugers-Scherzberg, Maximilian Plich

5.1 Goal

- Evaluate possible workflows from office documents (docx) via XML (TEI-P5) to various output formats (HTML, PDF, JATS, EPUB, …)
- Workflows should be individually configurable
- Workflow should be available within an OJS installation
- Ideally, there would be an XML editor available as OJS plugin to allow editing of documents, as well.

5.1.1 Ressources

In the PKP Sprint in 2018 we created on overview of XML import tools dedicated to PKP uses.

5.1.2 Road towards an OJS XML editor

The goal for an all-in-one solution is the inclusion of an XML editor in the OJS installation, which will be primarily used to make final changes in a document. The workflow would look like the one depicted in figure 2

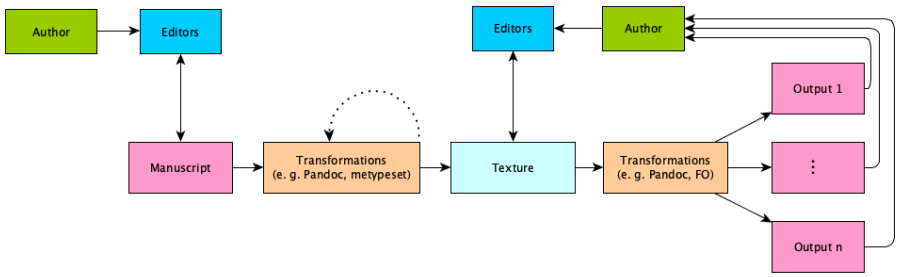

Figure 2: Multi-format workflow (abstraction)

The workflow requires

- Back-end for editors to perform editing and document conversion tasks
- Front-end for authors (used for writing and/or revision) to work directly in OJS XML editor[1]

[1] Our experience from similar editors is however the following: authors work with the usual word processors and copy formatted text over to the target platform.

5.1.3 Further considerations for front-end

1. Select input format (through dropdown menu): docx, odt, etc.
2. Insertion of metadata
3. Citation management
4. Image management
5. Select output format: HTML, JATS, etc.

5.2 Tool evaluation

The JATS standard (`https://jats.nlm.nih.gov/`) represents a condensed file format, which includes all metadata and text annotations of a single article in a machine-readable form. To make this format more accessible and easier editable for humans, the GUI-editor Texture (`https://github.com/substance/texture`) was created and is by now available as an OJS-plugin. In the workshop, we looked at a number of conversion tools to evaluate their coverage of JATS and how well, if at all, their output can be managed by Texture .

Beforehand, we set up set of requirements to be met.

1. Authors need to respect pre-configured document templates and editorial guidelines
2. Fixed parameters should be configurable for conversion (On what level? Hosting, Journal Manager/Editor)
3. Interactively query for variable parameters
4. Bibliographic references need to be standardized, their parsing has to be implemented separately
5. Automatic post-processing should be an option
6. Manual post-processing will be necessary in most cases
7. Other, individually defined, processing steps should be integrated, e.g. for
   (a) Language identification (e.g. with Apache Tika (`https://tika.apache.org/`) for localisation and hyphenation, possibly including dictionaries (that need to be maintained)
   (b) Adjustment of licenses
8. Store and make available the log of editing steps

We evaluated the following tools

1. meTypeset (footnotes need to be set separately)

2. docxConverter (no footnotes, only numeric citations possible)
3. Pandoc native
4. oxGarage

5.3 Example conversions

5.3.1 meTypeset

- Conversions are only possible if LibreOffice or Word are installed on the system.
- Together with the developers, the ReferenceLinker would need to be improved. In particular, the parsing of the list of references is error-prone.

5.3.2 Pandoc

```
pandoc -f docx -t jats --standalone -o <out_file>.xml
<in_file>.docx
```

- Result with complex docx files (Tables and formulæ, only formulæ): the generated JATS cannot be displayed in Texture:
  - `<table> is not valid`
  - `<email> not valid` (e-mail-link in Word)
  - simple enumeration i), ii), iii) does not work
  - simple graphics (without caption) do not work
  - `<alternatives>`-tag in `<inline-formula>` does not work
  - … and many errors more
- After removing all the tables graphics, formulæ and enumerations:
  - No commented errors from Texture anymore
  - Browser alert: `ERROR:Node already exists`
- Complete Citavi-bibliography is put into one `<p>`-tag

5.3.3 oxGarage

Can be used online (`https://oxgarage2.tei-c.org/`) or installed locally using Docker (see `https://github.com/TEIC/oxgarage/`) or on the command line (scripts available in `https://github.com/TEIC/Stylesheets` repository)

- Local command line commands:

```
/home/user/dev/Stylesheets/bin/docxtotei --localsource=/
    home/user/dev/dependencies/TEI  EVA_Paper_b.docx ep\
    textunderscore b-tei.xml
```

```
/home/user/dev/Stylesheets/bin/teitonlm --localsource=/home
    /user/dev/dependencies/TEI ep_b-tei.xml ep_b-jats.xml
```

- Result: JATS is not readable in Texture:

  ```
  <caption> is not allowed at the current position
  ```

5.4 Final evaluation

- Pandoc, meTypeset, XSLT, and Texture do not always all the tags that are allowed in JATS. There needs to be a list of tags that must not be used.
- A check of these dis-allowed elements has to be run before starting the conversion pipeline.
- Information on the formats should be collected, kept up-to-date, and made available to the community (Forum?, ojs-de.net?)

5.5 Next steps

- Development of the workflow requires lots of time, money and personnel
  1. joint project of several institutions feasible?
  2. is Texture an option at all? → state of the plugin?
- The repository `https://github.com/GrazingScientist/OJS-XML-Pipeline-Plugin/tree/master` contains examplary files used above.
- Information exchange and further development communicated through GitHub issue in that repository.