

Enriching The Metadata On CERN Document Server



Nalin Chhibber

Supervisor: Jens Vigen

Mentor(s): Tullio Basaglia, Javier Martin Montull

**[June – August]
2014**

**Scientific Information Service
General Infrastructure Services Department**

European Organization for Nuclear Research

CH-1211 Geneva 23, Switzerland

Contents:

1. Introduction.....	3
1.1 Background	
1.2 The Set-up	
2. Phase-I.....	4
2.1 MarcXML Standard	
2.2 Data acquisition from Springer	
2.3 Data acquisition from Forschungszentrum Jülich	
3. Phase-II.....	8
3.2 Configurable file in Crossref API	
3.1 Template for Crossref	
4. Results	9
4.1 Further Work	
4.2 What I have learned	
5. References.....	9

Abstract:

The project report revolves around the open source software package called Invenio. It provides the tools for management of digital assets in a repository and drives CERN Document Server. Primary objective is to enhance the existing metadata in CDS with data from other libraries. An implicit part of this task is to manage disambiguation (within incoming data), removal of multiple entries and handle replications between new and existing records. All such elements and their corresponding changes are integrated within Invenio to make the upgraded metadata available on the CDS. Latter part of the report discuss some changes related to the Invenio code-base itself.

1. Introduction:

1.1 BACKGROUND

As previously mentioned, Invenio is an open source software, typically used for open access repositories for scholarly and/or published digital content and as a digital library[1]. Invenio has been originally developed at CERN to run the CERN document server, managing over 1,000,000 bibliographic records in high-energy physics since 2002, covering articles, books, journals, photos, videos, and more.

Invenio comes as a suite of several more or less independent modules. My entire work was involved with 'admin-area' of Invenio and its specific modules, especially those related to data acquisition and data provision. Brief descriptions of these modules, specific tasks performed by them and corresponding interfaces are presented below[2].

DATA ACQUISITION RELATED MODULES:

Admin Module	Module Description
OAI Harvest Admin	Enables you to configure OAI metadata harvester for eventual periodical batch upload of data.
BibConvert Admin	Explains how to use bibliographic data convertor. Useful for batch upload of data.
BibMatch Admin	Tools for matching XML MARC files against the repository content. Useful when importing third-party metadata files.
BibUpload Admin	Enables you to configure eventual local special operations to be done on the data being uploaded.
WebSubmit Admin	Enables you to configure the submit interface and logic for various document types
BibEdit Admin	Enables you to directly manipulate bibliographic data, edit a single record, do global replacements, and other cataloguing tasks.
BibCheck Admin	Enables you to manage BibCheck configuration files. BibCheck is used to verify and correct records.
Publiline Admin	Enables you to approve documents by using a complex approval workflow.
BibAuthority Admin	Enables you to configure Authority Control in Invenio

Table-1

DATA PROVISION RELATED MODULES:

Admin Module	Module Description
BibIndex Admin	Enables you to configure "word files", i.e. to define which bibliographic fields are indexed into which word indexes.
BibRank Admin	Enables you to configure various ranking methods to be used by the search engine.
BibSort Admin	Enables you to configure the sorting methods displayed to the user.
BibClassify Admin	Enables you to automatically classify documents according to keyword taxonomies and thesauri.
BibFormat Admin	Enables you to specify how the bibliographic data is presented to the end user in the search interface.
BibSword Client Admin	Enables you to consult and refresh the status of the forwarded record to any SWORD Remote Server.
OAI Repository Admin	Enables you to define which records should be tagged to be exposed via the OAI Repository gateway, so that other other repositories can harvest your records.
WebSearch Admin	Enables you to configure the search interface for various metadata collections.
WebStat Admin	Enables you to configure the usage statistics reporting system.

Table-2

Other than the tools mentioned above, one of the tool under **SYSTEM GLUE MODULE**- BibSched was also very helpful. BibSched is a command line program that enables you to inspect bibliographic task queue, to postpone or reschedule jobs, to make priorities, to run periodical tasks, etc. I had no specific work related to **PERSONALIZATION RELATED MODULES**.

1.2 The Set-up

Invenio source code is open-sourced on github[3]. One can follow the installation guidelines and configure essential environment variables and run the local instance of Invenio demo-site: Atlantis. I used the Invenio dev-scripts from tibor simko - the creator of Invenio[4]. Dev-scripts are really handy and saves you from reinventing the wheel.

a. Clone devscript repository

Put the scripts somewhere under your PATH. For example, if you clone this repository under \$HOME/private/src:

```
$ cd $HOME/private/src
$ git clone https://github.com/tiborsimko/invenio-devscripts.git
```

then in your \$HOME/.bashrc you can say:

```
$ export PATH=$HOME/private/src/invenio-devscripts:$PATH
```

b. Configure devscript variables

One can use invenio-kickstart helper script that will populate \$HOME/.bashrc for you with some sensible defaults. This is useful especially for VM boxes.

c. Install Invenio

```
$ invenio-kickstart --yes-i-know --yes-i-really-know
```

d. Set up sudo rights

```
$ cat /etc/sudoers.d/johndoe
johndoe ALL=(www-data) NOPASSWD: ALL, \
    (root) NOPASSWD: /bin/rm -rf /opt/invenio/var/tmp/ooffice-tmp-files, \
    (root) NOPASSWD: /bin/mkdir -p /opt/invenio/var/tmp/ooffice-tmp-files, \
    (root) NOPASSWD: /bin/chown -R nobody /opt/invenio/var/tmp/ooffice-tmp-files, \
    (root) NOPASSWD: /bin/chmod -R 755 /opt/invenio/var/tmp/ooffice-tmp-files, \
    (root) NOPASSWD: /etc/init.d/apache2, \
    (root) NOPASSWD: /etc/init.d/mysql
```

2. Phase - I :

The primary objective of project was to deal with 'data acquisition' related modules and upgrade the metadata.

The metadata input into a running Invenio system can be done in two ways:

- **admin-oriented batch mode**, i.e. OAI Harvest to get data from OAI repositories, BibConvert to convert any input data into XML MARC, and BibUpload to upload XML MARC files into Invenio; and
- **author-oriented interactive mode**, i.e. WebSubmit to submit documents via Web. Once the data are uploaded in Invenio, you may want to modify them via BibEdit to edit the metadata.

2.1 MARC STANDARDS

MARC (MACHine-Readable Cataloging) standards are a set of digital formats for the description of items cataloged by libraries, such as books. It was developed by Henriette Avram[5] at the US Library of Congress during the 1960s to create records that can be used by computers, and to share those records among libraries. MARC 21 was designed to redefine the original MARC record format for the 21st century and to make it more accessible to the international community. MARC 21 has formats for the following five types of data: Bibliographic Format, Authority Format, Holdings Format, Community Format, and Classification Data Format.

All the bibliographic data in the Invenio system are internally represented in the MARC 21 format. There are several good reasons for this:

- MARC format is the standard in the library world. It is well established and has been used since 1960s.
- MARC is flexible enough to represent any metadata structure you may need now or in the future. Therefore, Invenio can adapt to your needs without altering its internal data structure.
- MARC technology, albeit developed in the punch card times (1960s!), can be well combined with recent technologies like XML. In fact, whenever bibliographic metadata are to be worked with externally in a file format, Invenio uses recently standardized MARC XML format provided by the Library of Congress.

MARXML is an XML schema based on the common MARC21 standards. It is a widely used standard for the representation and exchange of bibliographic, authority, holdings, classification, and community information data in machine-readable form.

CHOOSING MARC REPRESENTATION OF YOUR METADATA

METADATA CONCEPT	PROPOSED MARC 21 REPRESENTATION
Abstract	520 \$a
Author, first	100 \$a
Author(s), additional	700 \$a
Collection identifier	980 \$a
Email	8560 \$f
Imprint	260 \$a,b,c; 300 \$a
Keywords	6531 \$a
Language	041 \$a
OAI identifier	909CO \$o
Publication info	909C4 \$* [many subfields]
References	999C5 \$* [many subfields]
Primary report number	037 \$a [unique throughout the system!]
Additional report number(s)	088 \$a
Series	490 \$a,v
Subject	65017 \$a
Title	245 \$a
URL (e.g. to fulltext)	8564 \$u, \$z

The advantage of using these Invenio defaults is that one can use pre-defined configurations of BibConvert, BibFormat, and BibIndex.

2.2 DATA ACQUISITION FROM SPRINGER

Springer Publishing is an American publishing company of academic journals and books. They shared some Table of contents and Book back matter with CERN and provided us the ftp access. So the work flow to include this metadata into CDS was:

- Get the metadata
- Get the Record IDs corresponding to individual metadata
- Upload metadata to CDS on corresponding record

So I was first supposed to download the to download the zips locally, extract the metadata and organize the contents. Fortunately, their data was already in good format. The only issue was: I only had the ISBNs of those books and nothing else. To upload the files on CDS, one need to know the Record ID for corresponding material(book/journal/article). So I wrote a python script to get record IDs corresponding to the ISBN number.

```
from invenio.invenio_connector import InvenioConnector

def getRecid():
    server_url = "http://cds.cern.ch"
    try:
        server = InvenioConnector(server_url)
    except Exception, e:
        sys.stderr.write("Error...: %s\n" % (str(e),))
    global isbn
    isbn = [9780387095349, 9789400770942, 9789400755482]
    for f in xrange(len(isbn)):
        isbn[f] = isbn[f]
        try:
            rec = server.search(p="020_a:%s" % (isbn[f],), of="id")
            print("{0}\t\t{1}".format(isbn[f], rec))
        except Exception, e:
            sys.stderr.write("Error_: %s\n" % (str(e),))
```

Total records updated with Table of Contents: 2636

Total records updated with Book back matter: 2326

2.2 DATA ACQUISITION FROM FORSCHUNGSZENTRUM JÜLICH

Forschungszentrum Jülich GmbH (Jülich Research Centre) is a member of the Helmholtz Association of German Research Centres and is one of the largest interdisciplinary research centers in Europe. They allowed our group to access their metadata and use them in CERN Document Server. However, they only provided the access to certain materials with links in a file. However, this data was not sanitized and full of ambiguity. This certainly changed our work-flow to 4-step process:

- Get the metadata
- Sanitize records with disambiguation, handle multiple records and manage replications.
- Get the Record IDs corresponding to individual metadata
- Upload metadata to CDS on corresponding record

This time I wrote a script to download the metadata from the links they provided.

```
import urllib2

for line in open("urls.txt"):
    columns = line.split(" ")
    if len(columns) >= 2:
        try:
            f=urllib2.urlopen(columns[1])
            output = open(str(columns[1].rsplit('/',1)[1].replace('-', '')), 'w')
            output.write(f.read())
            output.close()
        except Exception, e:
            print(str(e)+"\n"+str(columns[1]))
```

SANITIZING THE DATA:

Following issues were supposed to be addressed:

1. Repeated ISBN with same link
2. Repeated ISBN with a different link.
3. Alternate ISBN(found in CDS) with a different link.
4. Not repeated, however alternate ISBN of the same record with different links.
5. ISBN not found on cds

It was relatively easy to deal with case 1 and 5. Once found, such records were supposed to be ignored/discarded. However, case 3 needed a smarter python script and case 2, 4 needed manual intervention. Once the data was sanitized, I used script like before to get the record IDs for each ISBN.

CREATING THE MARCXML:

This part was relatively easy and a simple python script solved the purpose.

```
import os
dictionary={}

def create_MARC(recid, path):
    out = []
    out.append('\t<record>')
    out.append('\t\t<controlfield tag="001">' + str(recid) + '</controlfield>')
    out.append('\t\t<datafield tag="FFT" ind1=" " ind2=" ">\n' +
        '\t\t\t<subfield code="a">' + path + '</subfield>\n' +
        '\t\t\t<subfield code="d">' + "Table of contents" + '</subfield>\n' +
        '\t\t\t<subfield code="t">Additional</subfield>\n' +
        '\t\t</datafield>')
    out.append('\t</record>\n')
    return "\n".join(out)

def createFile(f,out):
    f.write(out)

def loadDictionary():
    global dictionary
    with open("../MARCXML/Recs") as f:
        for line in f:
            (key, val) = line.split()
            dictionary[key] = val

def main():
    loadDictionary()
    count=0
    f=open("file.xml", "w")
    f.write("<?xml version='1.0' encoding='UTF-8'?'>\n")
    f.write("<collection>\n")
    pname="/home/nalinc/Desktop/Work/newTOCs/FILES/"
    fname=os.listdir(pname)
    for i in xrange(len(fname)):
        isbn = fname[i]
        isbn=fname[i][0: 8]
        if isbn in dictionary.keys():
            count+=1
            recid=dictionary[isbn]
            path="/afs/cern.ch/user/n/nchhibbe/public/TOC/"+str(fname[i])
            out=create_MARC(recid,path)
            createFile(f,out)
    f.write("</collection>\n")
    f.close()
    print("Done! {} files processed, {} accepted".format(len(fname),count))

if __name__ == '__main__':
    main()
```

3. Phase - II :

3.1 CONFIGURABLE FILE IN CROSSREF API

CrossRef is an official Digital Object Identifier (DOI) Registration Agency of the International DOI Foundation. Invenio makes use of it to import the data for articles and journals automatically with the help of DOIs. The Crossref API in Invenio is in `crossrefutils.py` it was supposed to be made configurable so that we can use a different file instead of 'crossref2marcxml.xml'. Previously, this file was hard-coded and it was relatively difficult to make changes into the code. The idea was to move this information from python code to xml configuration file to avoid unnecessary builds in future.

Following is the diff of changes made

```
diff --git a/config/invenio.conf b/config/invenio.conf
index e20cfbd..243b27a 100644
--- a/config/invenio.conf
+++ b/config/invenio.conf
@@ -2462,6 +2462,9 @@ CFG_CROSSREF_PASSWORD =
## CFG_CROSSREF_EMAIL -- crossref query services email
CFG_CROSSREF_EMAIL =
+## CFG_CROSSREF_2MARC -- filename in CFG_ETCDIR for crossref
+CFG_CROSSREF_2MARC = crossref2marcxml.xml
+

diff --git a/modules/miscutil/lib/crossrefutils.py b/modules/miscutil/lib/crossrefutils.py
index 571ae37..06df829 100644
--- a/modules/miscutil/lib/crossrefutils.py
+++ b/modules/miscutil/lib/crossrefutils.py
@@ -27,7 +27,7 @@ from xml.dom.minidom import parse
from time import sleep
from invenio.config import CFG_ETCDIR, CFG_CROSSREF_USERNAME, \
- CFG_CROSSREF_PASSWORD, CFG_CROSSREF_EMAIL
+ CFG_CROSSREF_PASSWORD, CFG_CROSSREF_EMAIL, CFG_CROSSREF_2MARC
from invenio.bibconvert_xslt_engine import convert
from invenio.bibrecord import record_get_field_value
from invenio.urlutils import make_invenio_opener
@@ -82,7 +82,7 @@ def get_marcxml_for_doi(doi):
# from bibconvert_xslt_engine file
# Setting the path to xsl template
xsl_crossref2marc_config = "%s/bibconvert/config/%s" % \
- (CFG_ETCDIR, "crossref2marcxml.xml")
+ (CFG_ETCDIR, CFG_CROSSREF_2MARC)
output = convert(xmltext=content, \
                 template_filename=xsl_crossref2marc_config)
```

3.2 TEMPLATE FOR CROSSREF

This task required changes in the `crossref2marcxml.xml` with the requirements from CDS. As discussed earlier, crossref module is used to import the metadata with the help of DOI. Requirement was to perform a mapping for collection name (in 980) to a template (which is stored in `/opt/cds-invenio/etc/bibedit/record_templates/`) to merge the imported information with existing templates. The diff for this task can be found on next page.


```

diff --git a/modules/bibconvert/etc/crossref2marcxml.xml b/modules/bibconvert/etc/crossref2marcxml.xml
index 3544f3d..76c2541 100644
--- a/modules/bibconvert/etc/crossref2marcxml.xml
+++ b/modules/bibconvert/etc/crossref2marcxml.xml
@@ -20,10 +20,7 @@
<!-- checking different types of documents -->
<xsl:when test="$doi[@type='journal_article']">
  <datafield tag="980" ind1=" " ind2=" ">
-    <subfield code="a"><xsl:text>Published</xsl:text></subfield>
-  </datafield>
-  <datafield tag="980" ind1=" " ind2=" ">
-    <subfield code="a"><xsl:text>citeable</xsl:text></subfield>
+    <subfield code="a"><xsl:text>ARTICLE</xsl:text></subfield>
  </datafield>
</xsl:when>

```

These changes along with some minor tweaks in the configuration file resulted in the desired changes.

4. RESULTS:

4.1 FURTHER WORK

There are still possibilities to improve Invenio. One of them being the ISBN import functionality. Currently, the metadata can only be imported with the help of crossref API through Digital Object Identifiers (DOI). It would be really handy to import essential metadata with the help of ISBNs. However, there are various cases that need to be considered for the same:

1. Support of ISBN-10 and ISBN-13.
2. Handling multiple results while querying with a single ISBN number.
3. Filtering the special characters (underscores/hyphens) before querying

4.2 WHAT I HAVE LEARNED

First and foremost, I have experienced the joy of working in a team. Invenio has been the first massively collaborative software engineering project I've worked on. Perhaps the area I've benefited the most from my stay at CERN is my ability to program in Python. Despite having used Python before, spending eight weeks on a project has given me some valuable experience and learning. MARCXML and MARC standard was pretty new to me. I have learned the basics of cataloging and data acquisition. The metadata import, though it seems easy, is complicated and deserves right attention. I acknowledged the fact that metadata we get is not guaranteed to be organized and can contain ambiguous, repeated, unrelated, alternate and non-interesting records. It often requires considerable amount of thought process to write the scripts keeping all these things in mind. I have learned to work on Invenio's command line admin-modules and how stuff works behind the scenes. Above all I have experienced the work flow followed by the developers at one of the most advanced research organizations on planet. I have acquired some new debugging skills, sharpened my knowledge on Git which will make me a better software engineer.

5. REFERENCES:

- [1] <http://invenio-software.org/>
- [2] <https://cds.cern.ch/help/admin/>
- [3] <https://github.com/inveniosoftware/invenio>
- [4] <https://github.com/tiborsimko/invenio-devscripts>
- [5] <http://www.loc.gov/loc/lcib/0605/avram.html>
- [6] <http://www.github.com/nalinc/Invenio>