Integrating Symplectic Elements with DSpace

Andrew Bennet, Software Developer
andrewb@symplectic.co.uk
Welcome, Dr Nina Snaith
School of Natural Sciences
nina.snaith@lilliput.ac.uk

My Actions (9)

There are 39 publications for you to claim or reject

Claim publications

My Summary

21 publications, plus 39 pending

0 records of impact
You have no records of impact yet.

0 professional activities
You have no professional activities yet.

0 teaching activities
You have no teaching activities yet.

0 grants

How do we calculate this? 😊

Claimed publications:
21 journal articles

In indexes:
6 Web of Science
9 Scopus

Accepted for Publication?
Deposit your work
Publication Metadata in Elements

- Reads publication metadata from a variety of sources
- Deduplicates everything
- Store full copies of all metadata
Publication Metadata in Elements

Crossref

Web of Science

Elements Publications
Repository Tools 1

Why?

● Reuse high quality metadata
● Streamlined process for academics
● Monitoring - Open Access policy compliance
Repository Tools 1

How?

- Built a repository “plugin” / custom API
- Elements pushes files and metadata to custom API
- Elements reads workflow status from custom API, stores this against a publication object.
Limitations

- Doesn’t pull item metadata into Elements
- Elements only knows about items deposited via it
- Only one repository connection possible
- Custom API is a maintenance burden on Symplectic
Repository Tools 2
Repository Tools 2

Philosophy:

- Treat repository like any other data source
  - Read and store full item metadata
  - Multiple repository connections in Elements
- Use native APIs for writing and reading
- Configured within Elements
Functionality:

- Harvest all DSpace items; match to existing publications
- Deposit files and metadata from Elements
- Monitor for changes in DSpace
- Attach additional files to existing DSpace items
Demo
How it works: Harvest

REST API

Read Items
How it works: Differential Harvest

- OAI-PMH endpoint can request modified item IDs
- Only works for Live items
How it works: Deposit

- Sword V2 endpoint inserts items into the workflow
- Sword V2 endpoint allows user impersonation
Interlude: REST API Experiences

- Pretty good documentation
- Expanding feature coverage (but not fully comprehensive!)
- Occasionally suffers from bugs
- Excited for DSpace 7.x API
Interlude: Integration Testing

Automated testing against full range of supported DSpace versions using Docker images
How it works: Crosswalks

- Two sets of crosswalks required
  - Inbound: map DSpace to Elements
  - Outbound: map Elements to DSpace
- Developed a format which allows comprehensive specification of metadata mappings

```xml
<xwalk:field-mapping to="title">
  <xwalk:field-source from="dc.title"/>
</xwalk:field-mapping>
```
Inbound Crosswalks

<api:person>
  <api:last-name>Bender</api:last-name>
  <api:initials>CM</api:initials>
</api:person>

<dc.contributor.author> Bender, C.M. </dc.contributor.author>
Outbound Crosswalks

<api:person>
  <api:last-name>Bender</api:last-name>
  <api:initials>CM</api:initials>
  <api:first-names>Carl M</api:first-names>
  <api:identifiers>
    <api:identifier scheme="orcid">0000-0002-6571-2872</api:identifier>
  </api:identifiers>
</api:person>

<dc.contributor.author> Bender, C.M. (0000-0002-6571-2872) </dc.contributor.author>
Crosswalk Capabilities

- String manipulation (split, concatenate, format, etc)
- Boolean algebra
- Map multivalued compound data types
- Dictionary lookups
- Regex!
Crosswalk Capabilities

- Regex allows powerful metadata manipulation

Future Plans

- “Subsequent Deposit” - coming next week
- DSpace 7 support - when available
- Automatic Metadata Updates - later 2018
  - Automated push of new metadata in Elements to DSpace
  - Framework to support “relevant” change detection: in development