

Creating DOIs with rich metadata using DSpace

Open Repositories, June 2024

Who we are

- Sheila Rabun, Lyrasis
- Kelly Stathis, DataCite
- Pascal Becker, The Library Code
- Claudio Cortese, 4Science



Agenda

- DataCite DOI and Metadata Basics
 - What is a DOI?
 - DataCite account structure
 - DOI registration process
 - The DataCite Metadata Schema
- Configuring DSpace to register DataCite DOIs
 - DSpace 7
 - DSpace-CRIS 7
- Getting involved

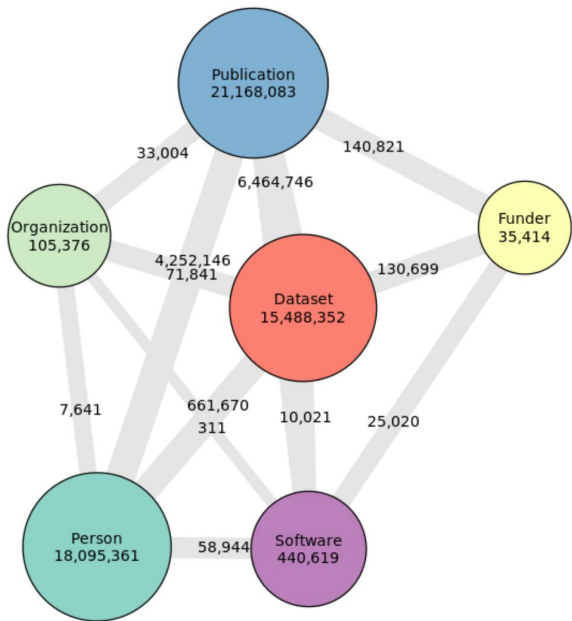


CONNECTING RESEARCH,
ADVANCING KNOWLEDGE

DataCite

DOI and Metadata Basics

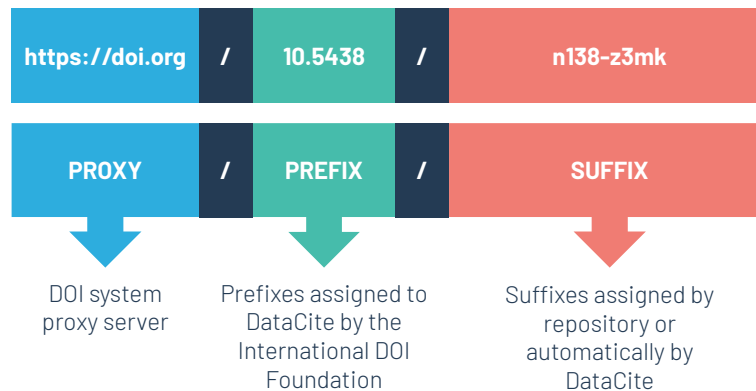
What is a DOI?



PID Graph as of October 2023

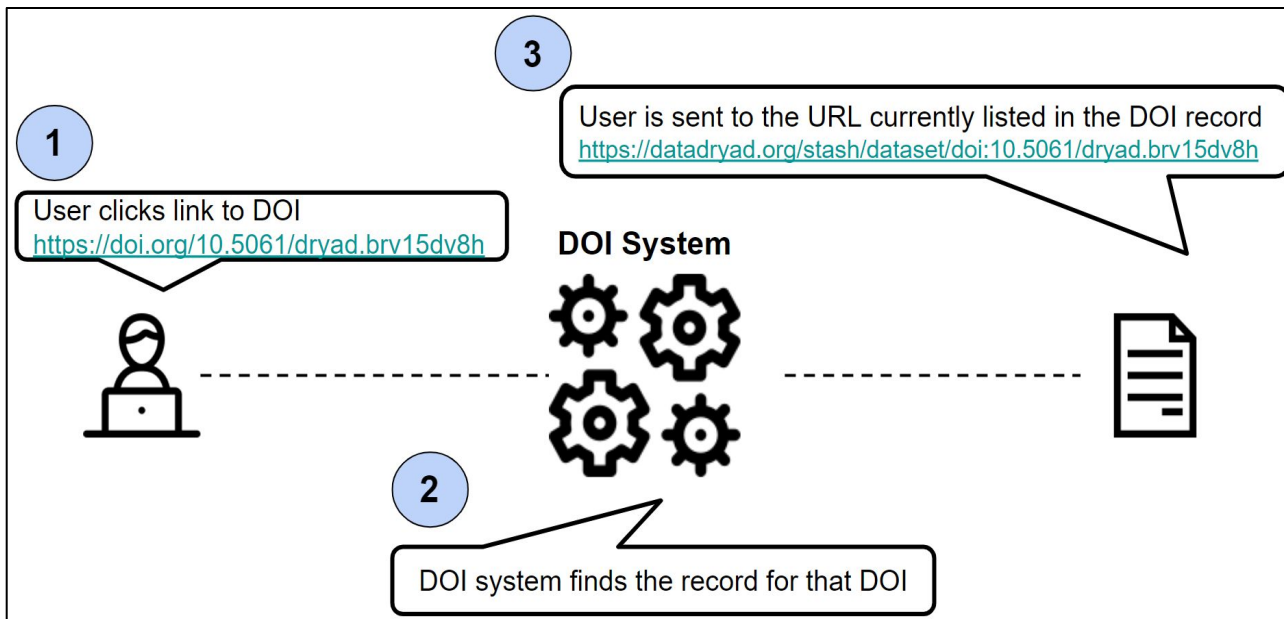
- **D**igital **O**bject Identifier - persistent identifier (PID) for outputs & resources
- Registered through a DOI registration agency such as DataCite or Crossref
- DOIs:
 - contain **open metadata** describing the object
 - redirect to a landing page containing information about the object
 - can be used with other PIDs to **connect related entities**
 - support the **FAIR Principles**, making outputs and metadata more **F**indable, **A**ccessible, **I**nteroperable, and **R**eusable

DOI Structure



Why DOIs?

DOIs help to ensure that outputs & resources can continue to be found, accessed, and used over time despite changes in the location (URL) of the object - avoiding the "404 Page Not Found" error.



DataCite DOI Registration Agency



- Global non-profit membership organization
- Started in 2009 as part of an effort to make data more citable
- As of 2024, DataCite DOIs can be assigned to 30 different resource types
- 3000+ repositories in the world provide DataCite DOIs for data and other research outputs.
- One of 12 official DOI registration agencies worldwide



3000+

Repositories



303

Members



52

Countries



60m+

DOIs



1392

Organizations

DataCite account structure



- All DataCite Direct Members or Consortium Organizations have two types of live/production and sandbox/testing DataCite accounts:
 - **Direct Member or Consortium Organization Account** - only one per organization, used for managing information about the organization and creating repository account(s)
 - **Repository Account(s)** - can have multiple per organization, used for registering DOIs and updating DOI metadata and URLs
- DOIs are created using the Repository Account credentials:
 - Account ID (e.g., ABCD.EFGH)
 - Password

DOI registration process



- DataCite offers two APIs that enable DOI registration:
 - REST API
 - MDS API
- Users authenticate with DataCite Repository Account credentials
- Options for DOI registration:
 - Build your own integration using a DataCite API
 - Use the DataCite Fabrica web interface
 - Work through a repository platform, including DSpace, that integrates with a DataCite API

DataCite Metadata Schema



- The DataCite Metadata Schema is used for DataCite DOIs.
- DataCite members create metadata during the registration of a DOI which provides information about the relevant resource.
- Includes 20 metadata properties.
- Intended for accurate and consistent identification of a resource for citation and retrieval purposes.

DataCite Metadata Schema Properties



- The schema consists of 20 metadata properties (sometimes called “fields” or “elements”).
- Hierarchical structure: some properties have sub-properties.
- Six properties are mandatory; six are recommended; eight are optional.
- Some can be repeated.
- Some have controlled list values, some allow free text.

Mandatory Properties

6 Mandatory

ID	Property
1	Identifier
2	Creator
3	Title
4	Publisher
5	PublicationYear
10	ResourceType

```
<identifier identifierType="DOI">10.21384/example</identifier>
<creators>
  <creator>
    <creatorName nameType="Personal">Garcia, Sofia</creatorName>
  </creator>
</creators>
<titles>
  <title xml:lang="en-US">Minimal DataCite XML Example</title>
</titles>
<publisher xml:lang="en">DataCite</publisher>
<publicationYear>2023</publicationYear>
<resourceType
resourceTypeGeneral="Other">Example</resourceType>
```

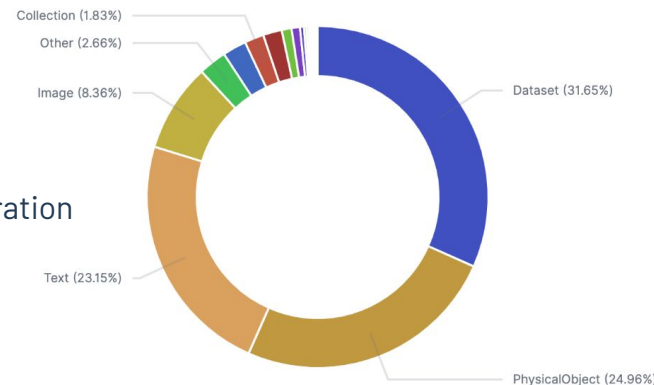
resourceTypeGeneral

6 Mandatory

ID	Property
1	Identifier
2	Creator
3	Title
4	Publisher
5	PublicationYear
10	ResourceType

Audiovisual
Book
BookChapter
Collection
ComputationalNotebook
ConferencePaper
ConferenceProceeding
DataPaper
Dataset
Dissertation
Event
Image
Instrument
InteractiveResource
Journal

JournalArticle
Model
OutputManagementPlan
PeerReview
PhysicalObject
Preprint
Report
Service
Software
Sound
Standard
StudyRegistration
Text
Workflow
Other



All Properties

6 Mandatory

ID	Property
1	Identifier
2	Creator
3	Title
4	Publisher
5	PublicationYear
10	ResourceType

6 Recommended and 8 Optional

ID	Property	Obligation
6	Subject	R
7	Contributor	R
8	Date	R
9	Language	O
11	AlternateIdentifier	O
12	RelatedIdentifier	R
13	Size	O
14	Format	O
15	Version	O
16	Rights	O
17	Description	R
18	GeoLocation	R
19	FundingReference	O
20	RelatedItem	O

Why include robust metadata?

Understanding the scholarly record

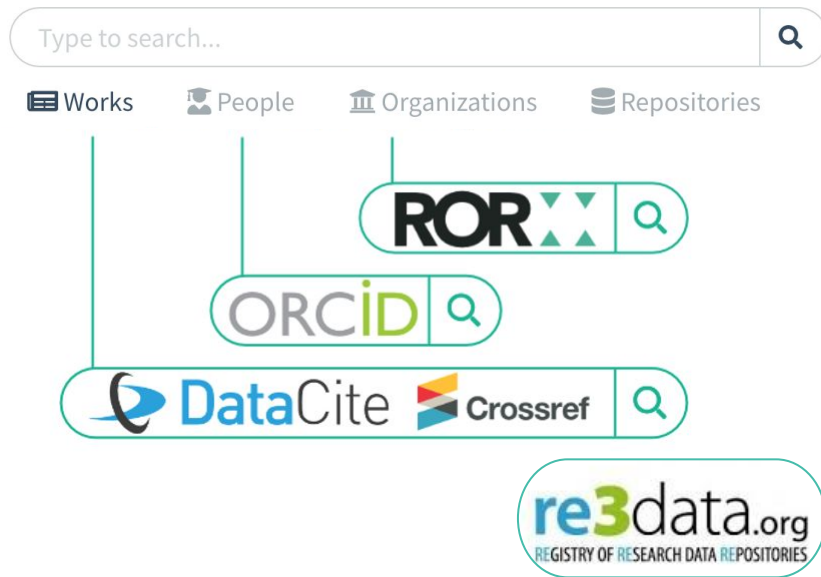
We can answer questions about works, people, and organizations, such as:

- How many papers cite this dataset?
- What software was used to create this dataset?
- What research outputs were produced by this researcher (person)?
- Which datasets are associated with a particular research institution?
- What works were funded by a particular organization?

DataCite Commons

DataCite Commons (<https://commons.datacite.org>) is a portal where anyone can go to search the entire DataCite metadata catalog as well as several other PID resources. You can find:

- **Works:** Search the metadata catalog of all DataCite DOIs, as well as a large number of Crossref DOIs.
- **People:** Search for researchers with ORCID iDs.
- **Organizations:** Search for organizations with ROR IDs.
- **Repositories:** Search for DataCite repositories and repositories with re3data records.



- **DataCite support site:** <https://support.datacite.org/>
 - Account structure: <https://support.datacite.org/docs/datacite-account-types>
 - Testing guide: <https://support.datacite.org/docs/testing-guide>
- **DataCite Metadata Schema website:** <https://schema.datacite.org/>
- **DataCite Metadata Schema docs:**
<https://datacite-metadata-schema.readthedocs.io/en/4.5/>

Getting started

DOI registration through DSpace integration

If you are using DSpace but not registering DOIs yet, you can enable DataCite DOI registration using the integration - we'll demonstrate how today.

- To register DOIs, you'll need a DataCite Repository account, whether as a Direct Member or as part of a Consortium.
- More information on how to join DataCite:
<https://datacite.org/become-a-member/>.

If you are already registering DOIs through DSpace, this workshop will show you how to make the most of the integration.

Getting involved

DataCite community

- Become a DataCite member
 - If you are not yet a member or part of a consortium, get in touch with us:
<https://datacite.org/become-a-member/>
- Help shape the DataCite Metadata Schema
 - Suggest changes here: <https://schema.datacite.org/contribute.html>
- Contribute to the Global Access Fund
 - Support underrepresented communities in accessing PID infrastructure:
<https://datacite.org/global-access-fund-call-for-support/>
- Reach out to us at support@datacite.org with any questions

DSpace community

- Join DSpace communication channels
 - DSpace community mailing lists: <https://wiki.lyrasis.org/display/DSPACE/Mailing+Lists>
 - DSpace Wiki: <https://wiki.lyrasis.org/display/DSPACE/>
- Register your DSpace instance
 - Submit this form: <https://registry.lyrasis.org/registry/register-your-site/> to show up on the global DSpace registry list/map:
https://registry.lyrasis.org/?gv_search&filter_10=DSpace&filter_4_6&filter_3&filter_20&filter_28&mode=all
- Contribute to the community
 - Become a DSpace member organization:
<https://www.lyrasis.org/programs/Pages/DSpace.aspx>
 - Participate in community governance: <https://dspace.lyrasis.org/governance/>
 - Contribute code and other expertise: <https://dspace.lyrasis.org/community-contributors/>

Pascal-Nicolas Becker

DOIs with DataCite and DSpace

Workshop at Open Repositories 2024

Göteborg, June 3rd, 2024



<https://lib-co.de/or24>

Overview

Agenda

- DOI support in DSpace
- How to mint DOIs with DataCite and DSpace
- Sending metadata to DataCite, enriching your metadata
- Filtering which items get a DOI
- Listing persistent identifiers in submission



DOI support in DSpace

- Minting of DOIs with DataCite introduced in 4.0 (Dec. 2013)
 - DSpace can create DOIs
 - DSpace send metadata to DataCite
 - DSpace can register URLs for DOIs at DataCite
- Filtering which items get DOIs assigned introduced in 7.1 (Nov. 2021)
 - Only mint DOIs for items that contain a file, have or do not have certain metadata, are included in some specific collections, ...
 - Allow administrators to manually and retrospectively mint DOIs for item that don't have one
- Show identifiers in submission introduced in 7.5 (Feb. 2023)
 - Show the identifiers an item will get once it is accepted in the repository
 - Allows people to include the DOI into a file they want to publish



Basic concepts

- DataCite assigns a DOI prefix to each of its members, example: 10.5072
- You can mint any DOI that starts with this prefix and a slash:
 - ✓ 10.5072/dspace-is-great-123
 - ✓ 10.5072/xa.9823-6lk_j2390
 - X 10.5072.aklj-klj
 - X 10.123/dspace-is-great-123
- DSpace normally assigns sequential numbers for DOIs: 10.5072/1, 10.5072/3, ...
- DSpace introduced the concept of “namespaces”, so that you can use one prefix with multiple DSpace installations
 - A namespace is just any static string, between the prefix and suffix of a DOI, e.g. 10.5072/edu-repository-123, 10.5072/edu-repository/123
- DataCite uses different credentials for their test and production environment



States of a DOI

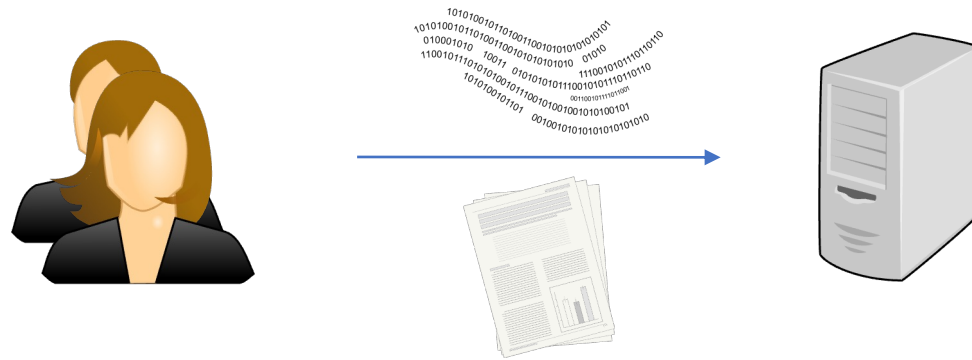
- DataCite requires DOI registrants to send metadata and URLs for a DOI to get fully registered
- Sending metadata is a prerequisite for sending URLs to which a DOI should be redirected
- In DSpace, we refer to sending metadata as "reserving" a DOI, because it semantically associates a DOI with an item (or its metadata)
- In DSpace, we call sending URL(s) for a DOI "registration", because this is what finally makes the DOI usable and active
- If DSpace should "register" a DOI for which no metadata has been sent yet, it will detect this and automatically send metadata first
- A DOI can have different states: it can exist only in DSpace, metadata but not URLs could have been sent to DataCite, or it can exist in DSpace, metadata and URLs have been sent to DataCite
- A DOI can be marked at DataCite as being no longer "active"



Storing DOIs within DSpace

- DOIs are metadata of an item and stored as such
 - This is true for DOIs minted externally (publishers) and internally (directly by your DSpace instance)
 - DSpace stores metadata in the database table metadatavalue
- For the DOIs minted by DSpace, DSpace must track their status and send updates to DataCite
 - DSpace stores DOIs it minted and their status in the database table 'doi'
 - See <https://github.com/DSpace/DSpace/blob/main/dspace-api/src/main/java/org/dspace/identifier/DOIIdentifierProvider.java> for a list of states and their internal IDs (states are stored as integers)
 - If an item with a DOI gets deleted, DSpace still stores the DOI and its status “deleted” in the database table 'doi'
- DOIs minted by DSpace will be added as metadata to an item, after all information has been sent successfully to DataCite

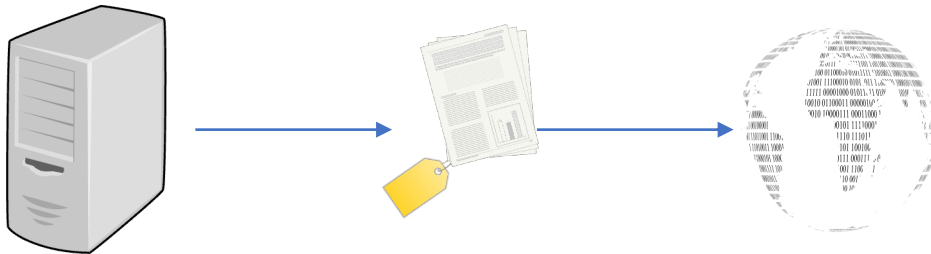
When does DSpace mint DOIs?



A user submits an item



Reviewers accept and archive the item, DSpace mints a DOI and stores it in the doi table only



Later a cronjob sends information to DataCite and stores the DOI also as metadata, so it shows up on the item view



DataCite's DOI APIs

- DataCite offers different APIs to register DOIs
- DSpace currently (as of version 8.0) uses the MDS (metadata store)
 - <https://support.datacite.org/docs/mds-api-guide>



Minting DOIs

Preconditions

- You need a DOI prefix and credentials from DataCite
- Normally you need a membership directly at DataCite or via a DataCite consortium
- If you do not have a test and production account, you can request those

Basic Configuration I

- In DSpace you can override any setting of any *.cfg file in local.cfg. It's up to you if you make the following changes in dspace.cfg or in local.cfg
- Configure the username, password, doi prefix, namespace, and DOI publisher:

```
identifier.doi.user = DATACITE.USERNAME  
identifier.doi.password = TOP-SECRET!1!!  
identifier.doi.prefix = 10.5072  
identifier.doi.namespaceseparator = dspace-  
crosswalk.dissemination.DataCite.publisher = My University
```
- Add “doi” to the default DOI Event Consumers:

```
event.dispatcher.default.consumers = versioning, discovery, eperson,  
harvester, doi
```
- Restart tomcat after changing this configuration



DOIIdentifierProviders vs. VersionedDOIIdentifierProvider

- The only differences between the DOIIdentifierProvider and the VersionedDOIIdentifierProvider is the way they generate DOIs for versions of items
- The DOIIdentifierProvider always concatenates the DOI prefix, a slash, the namespace (if set) and a number: 10.5072/dspace-1, 10.5072/dspace-2, ...
- The VersionedDOIIdentifierProvider does the same for first versions of items and adds a dot and the version number to following versions of the same item: 10.5072/dspace-1, 10.5072/dspace-1.2, 10.5072/dspace-1.3, ...
- The ISO norm 26324:2012 states: “The DOI name is an opaque string for the purposes of the DOI system. No definitive information may be inferred from the specific character string of a DOI name.”



Basic Configuration II

- DSpace uses Spring to configure alternative versions of source code
- You need to modify `dspace/config/spring/api/identifier-service.xml`
- Activate: `DOIIdentifierProvider` or `VersionedDOIIdentifierProvider` and the `DataCiteConnector` (just remove comment signs `<!-- -->`)
- Configure if DataCite's test or production system should be used:

```
<property name='DATACITE_HOST' value='mds.test.datacite.org' />
```

Or

```
<property name='DATACITE_HOST' value='mds.datacite.org' />
```



Mint DOIs!

- Once DSpace is configured to mint DOIs and you have restarted tomcat, submit an item
- If you have configured the collection to use review steps, make sure the item has been accepted in all review steps and is archived in the repository
- You will not see the DOI in the Item view, as the DOI is not stored as metadata yet
- Check via the command line, if the DOI is queued to be sent to DataCite:
- `[dspace-instal1]/bin/dspace doi-organiser -l`

Command line options and cronjobs

- `[dspace-install]/bin/dspace` contains the command line tool of DSpace
- `[dspace-install]/bin/dspace doi-organiser` is the command line tool to handle DOIs
- `[dspace-install]/bin/dspace doi-organiser --help` prints the online help
- `[dspace-install]/bin/dspace doi-organiser -l` lists up all internally stored DOIs, whose state should be changed
- There are commands to reserve DOIs (send metadata to DataCite, as updates or for the first time) or to register them (send URLs or metadata and URLs to DataCite)
- Run this as a cronjob automatically

Cronjob

```
[dspace-install]/bin/dspace doi-organiser -u -q
```

```
[dspace-install]/bin/dspace doi-organiser -s -q
```

```
[dspace-install]/bin/dspace doi-organiser -r -q
```

```
[dspace-install]/bin/dspace doi-organiser -d -q
```

```
[dspace-install]/bin/dspace doi-organiser -u -q
```



Caveats

- When a DOI is registered successfully, the DOI is stored as metadata in the item
- When an item is changed, DSpace wants to send the updated metadata to DataCite
- When a DOI gets registered, it will be immediately marked as needing to send a metadata update to DSpace, because of the added DOI
- Just run `dspace doi-organiser -u` again

Metadata

Sending Metadata to DataCite

- The cronjob you activated before, sends metadata to DataCite
- DSpace stores metadata fields for each item, DataCite has its own schema to represent metadata information (see <https://schema.datacite.org>)
- DSpace produces XML representing the metadata of an item and XSLT to transform this XML into XML that DataCite expects
- XSLT is a functional language to transform XML into another XML structure or even other formats
 - We cannot cover this here, but there is a ton of information about this on the internet
- To change what information is sent to DataCite you can edit the file `dspace/config/crosswalks/DIM2DataCite.xsl`



DIM2DataCite.xsl

```
<?xml version="1.0" encoding="UTF-8"?>

<!--
Document   : DIM2DataCite.xsl
Created on : January 23, 2013, 1:26 PM
Updated on : January 30, 2024, 3:00 PM
Author    : pbecker, ffuerste, ypaulsen
Description: Converts metadata from DSpace Intermediate Format (DIM) into
            metadata following the DataCite Metadata Schema 4.5
-->

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:dspace="http://www.dspace.org/xmlns/dspace/dim"
                xmlns="http://datacite.org/schema/kernel-4"
                version="2.0">

  <!-- CONFIGURATION -->
  <!-- The parameters prefix, publisher, datamanager and hostinginstitution
        moved to DSpace's configuration. They will be substituted automatically.
        Please take a look into the DSpace documentation for details on how to
        change those. -->
  <!-- This file handles the transformation of metadata into the DataCite
        Schema. You should customize it to match your local metadata schema
        and submission forms. Please note that this must produce valid XML
        according to the DataCite Schema. Otherwise you will not be able
        to register DOIs anymore. Please follow and reuse the examples
        included in this file. For more information on the DataCite
        Schema, see https://schema.datacite.org. -->

  <!-- We need the prefix to determine DOIs that were minted by ourself. -->
  <xsl:param name="prefix">10.5072/dspace</xsl:param>
```



DIM2DataCite.xsl

```
<!--  
  MANDATORY PROPERTIES  
-->  
  
<!--  
  DataCite (1)  
  Template Call for DOI identifier  
  Occ: 1  
-->  
<!--  
  dc.identifier.uri may contain  
  repository contains an item th  
  company as well. We have to en  
  as primary identifiers only.  
-->  
<xsl:apply-templates select="//dsp -->  
  <xsl:template match="dspace:field[@mdschema='dc' and @element='identifier' and @qualifier and (contains(., $prefix))]">  
    <identifier identifierType="DOI">  
      <xsl:if test="starts-with(string(text()), 'https://doi.org/')">  
        <xsl:value-of select="substring(., 17)"/>  
      </xsl:if>  
      <xsl:if test="starts-with(string(text()), 'http://dx.doi.org/')">  
        <xsl:value-of select="substring(., 19)"/>  
      </xsl:if>  
    </identifier>  
  </xsl:template>
```



DIM2DataCite.xsl

```
<!--
  DataCite (2)
  Add creator information.
  Occ: 1-n
-->
<creators>
  <xsl:choose>
    <xsl:when test="//dspace:field[@mdschema='dc' and @element='contributor' and @qualifier='author']">
      <xsl:apply-templates select="//dspace:field[@mdschema='dc' and @element='contributor' and @qualifier='author']" />
    </xsl:when>
    <xsl:otherwise>
      <creator>
        <creatorName>(:unkn) unknown</creatorName>
      </creator>
    </xsl:otherwise>
  </xsl:choose>
</creators>

<!-- DataCite (2) :: Creator -->
<xsl:template match="//dspace:field[@mdschema='dc' and @element='contributor' and @qualifier='author']">
  <creator>
    <creatorName>
      <xsl:value-of select="." />
    </creatorName>
  </creator>
</xsl:template>
```



Changing the metadata transformation

- You can change the DIM2DataCite.xsl, to use or add information from local metadata fields
- To see the XML the XSLT processor works on, run:
- `[dspace-install]/bin/dspace dsrun org.dspace.content.crosswalk.XSLTDisseminationCrosswalk dim 123456789/3`
 - This expects that an item with the handle 123456789/3 exists
 - Of course, you can use any handle of any item in your repository

Testing the metadata transformation

- The XSLT transformation must produce XML that is valid DataCite XML
- Read the information at <https://schema.datacite.org>
- Create the XML that will be sent to DataCite by running:
- `[dspace-install]/bin/dspace dsrun org.dspace.content.crosswalk.XSLTDisseminationCrosswalk DataCite 123456789/3`
 - If the DOI is not stored as metadata of this item, it will not be in the XML
 - DSpace checks automatically if the DOI is in the metadata and will add it if necessary before it sends the metadata to DataCite

Item filter

Do we want to mint DOIs for all items?

- How about bibliographic entries without files? -> No
- How about previously published items that already have a DOI? -> Yes
 - We run repositories, to ensure that their contents are accessible
 - We want to have DOIs that are not forwarding to paywalls, but to Open Access Repositories
 - If we publish pre- or post prints, the page count may differ from the original publication
- How about restricted content, that can be read only within the campus? -> Probably No



Mint DOIs selectively

- Different repositories require different criteria for whether to mint DOIs for a given item
- Some will mint DOIs depending on the item's owning collection, some depending on metadata, some on permissions, some depending on number of files
- Inspired by filters in XOAI, The Library Code introduced a logic framework to DSpace
 - Spring service that is reusable anywhere
 - Logical statements are defined as Spring beans in XML



Logic, Filters, and Conditions...

- Filters define rules, by using a single operator (AND / OR / NOT), or a single condition
- Each condition references a Java class that inspects an item and returns true / false
- Each operator can include any number of conditions or operators
- Complete boolean algebra is possible in this configuration

- Filters are use logical operators to combine different conditions to create complex rules
- Filters can be used by other parts of DSpace, such as the DOIIdentifierProvider



dspace/config/spring/api/identifier-service.xml

```
<bean id="org.dspace.identifier.DOIIdentifierProvider"  
  class="org.dspace.identifier.DOIIdentifierProvider"  
  scope="singleton">  
  <property name="configurationService"  
    ref="org.dspace.services.ConfigurationService" />  
  <property name="DOIConnector"  
    ref="org.dspace.identifier.doi.DOIConnector" />  
  <property name="filter" ref="doi-filter" />  
</bean>
```

dspace/config/spring/api/item-filters.xml

```
<!-- Example DOI Filter. An item has to pass the filter (filter returns true) to get a DOI.
      If the filter returns false on an item, minting of a new DOI for that item is prevented. -->
<bean id="doi-filter" class="org.dspace.content.logic.DefaultFilter">
  <property name="statement">
    <bean class="org.dspace.content.logic.operator.And">
      <property name="statements">
        <list>
          <!-- Make sure the item is archived -->
          <ref bean="is-archived_condition"/>
          <!-- Make sure the item is not withdrawn -->
          <bean class="org.dspace.content.logic.operator.Not">
            <property name="statements" ref="is-withdrawn_condition"/>
          </bean>
          <!-- Don't create new DOIs for items that already have one -->
          <bean class="org.dspace.content.logic.operator.Not">
            <property name="statements" ref="dc-identifier-uri-contains-doi_condition"/>
          </bean>
          <!-- Create DOIs for items only that do have at least one bitstream. -->
          <ref bean="has-at-least-one-bitstream_condition"/>
        </list>
      </property>
    </bean>
  </property>
</bean>
```



dspace/config/spring/api/item-filters.xml

```
<bean id="is-archived_condition" class="org.dspace.content.logic.condition.IsArchivedCondition">
  <property name="parameters">
    <map></map>
  </property>
</bean>
<bean id="is-withdrawn_condition" class="org.dspace.content.logic.condition.IsWithdrawnCondition">
  <property name="parameters">
    <map></map>
  </property>
</bean>
<!--
  A filter that checks if any value of dc.identifier.uri contains "10.12345/".
-->
<bean id="dc-identifier-uri-contains-doi_condition"
  class="org.dspace.content.logic.condition.MetadataValueMatchCondition">
  <property name="parameters">
    <map>
      <entry key="field" value="dc.identifier.uri" />
      <entry key="pattern" value="10.12345/" />
    </map>
  </property>
</bean>
```

Example of conditions

- **MetadataValuesMatchCondition** takes a list of regular expressions and a field.
True if at least one metadata value for the field matches one of the patterns.
Use cases: “any value”, types, identifier prefixes, subject, etc.
- **ReadableByGroupCondition** takes a group name and action.
True if the action (READ, ADD, DELETE, etc) is allowed by the group.
Use cases: public access, restricted items - if extended to collection DSOs, could be used with workflow permissions
- **BitstreamCountCondition** takes a minimum and maximum and a bundle name.
True if the bundle has at least the minimum and at most the maximum. If just a minimum or maximum was specified, the other is unbounded
Use cases: “has file”, “has one thumbnail”, “has more than three files”

Example of filters

- OpenAIRE filter has a list of required conditions to return true only if an item is compliant with OpenAIRE guidelines
- DOI filter returns true only if an item identifier URI does not contain the DOI prefix and the item contains at least one bitstream
- Filters can reference other filters, since filters, operators and conditions all implement the LogicalStatement class
- ... the possibilities are endless!

Filters in practice

- A filter must return true for an item to get a DOI
 - Administrators can manually assign DOIs to item that were filtered out (Edit Item -> assign DOI)
 - It's up to you, to adapt the filters to your local requirements
 - If the is-archived_condition is still part of the doi-filter, please remove it, as a new item is not archived when the filter is applied.
 - Documentation in the DSpace manual: <https://wiki.lyrasis.org/x/2wsoCw>
1. Define your filters in item-filters.xml
 2. Test them on the command line (see link in the manual above)
 3. Configure the DOIIdentifierProvider to use the filter (in identifier-service.xml, see Slide 30)



Identifiers in Submission

Adding DOIs to your PDFs



- Authors might want to add the DOI of the document to the PDF itself: „Cite as: ...“
- How can we add the DOI, if it is minted after the item was submitted and archived?
- DSpace can assign a DOI to an item that is still in submission
 - It shows the DOI the item will get, once it is archived
 - If the item gets archived, it will get this DOI and no other one
 - No information will be sent to DataCite before the item gets archived, DataCite will not charge for the DOI until it is registered
 - A DOI assigned to a workspace item has the status “PENDING” in DSpace’s database (table ‘doi’)

But if items get filtered?

- Assume only items that include at least one public file should get DOIs
- We want to show the DOI in the submission, even if no file was uploaded yet
- If the item is submitted without a file, no DOI should be minted/registered at DataCite

- You can define two filters in DSpace:
 - One filter checks whether a DOI should be shown in the submission
 - The other filter checks if this DOI should be registered when the item gets archived
 - You can change the message in the message catalog, e.g. “If you upload a file, this item will get the following DOI ... Please include that in your PDF”
 - You can ensure that DOIs are not assigned in submission to entities that should never get a DOI

General Configuration

- To be able to register DOIs for workspace/workflow items, you must change the default configuration in [dspace]/config/modules/identifiers.cfg or add it to local.cfg:
`identifiers.submission.register = true`
- You can name a filter that is defined in item-filters.xml and decides if a DOI should be shown in submission or not:
`identifiers.submission.filter.workspace = always_true_filter`
- Imagine an item matches this “workspace-filter” and is then changed in submission, so that it doesn’t match anymore. You can configure DSpace to either still show the DOI or hide it, until the item matches the filter again:
`identifiers.submission.strip_pending_during_submission = true`
- You can define whether to show handles and/or DOIs in submission:
`identifiers.submission.display = handle`
`identifiers.submission.display = doi`



Submission Configuration

- Activate the submission step “identifiers” in your submission-process in `dspace/config/item-submission.xml` by removing the comment signs or adding the step to your submission-process:

```
<submission-definitions>
```

```
<!--This "traditional" process defines the DEFAULT item submission process -->
```

```
<submission-process name="traditional">
```

```
<!--Uncomment to display the SAMPLE step as your first step -->
```

```
<!--<step id="sample"/> -->
```

```
<step id="collection"/>
```

```
<!-- To include the 'Show Identifiers' step, uncomment the line below. Make sure  
that identifiers.cfg and identifier-service.xml are properly configured. -->
```

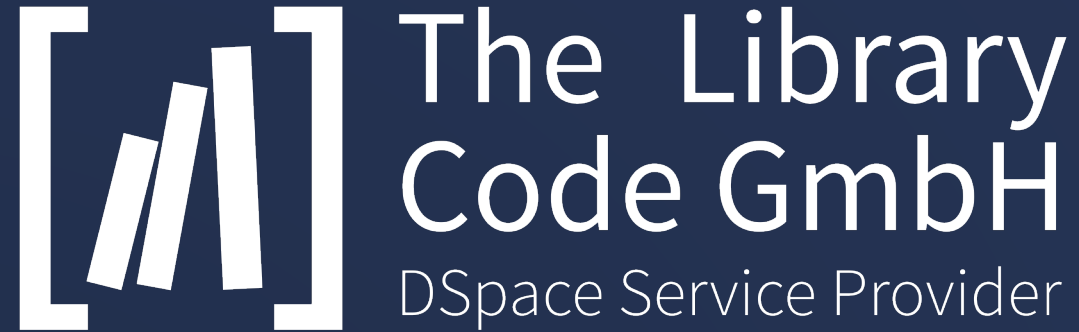
```
<!--<step id="identifiers"/>-->
```

```
<!--Step will be to Describe the item. -->
```

```
<step id="traditionalpageone"/>
```

```
<step id="traditionalpagetwo"/>
```





contact@the-library-code.de
<https://www.the-library-code.de>



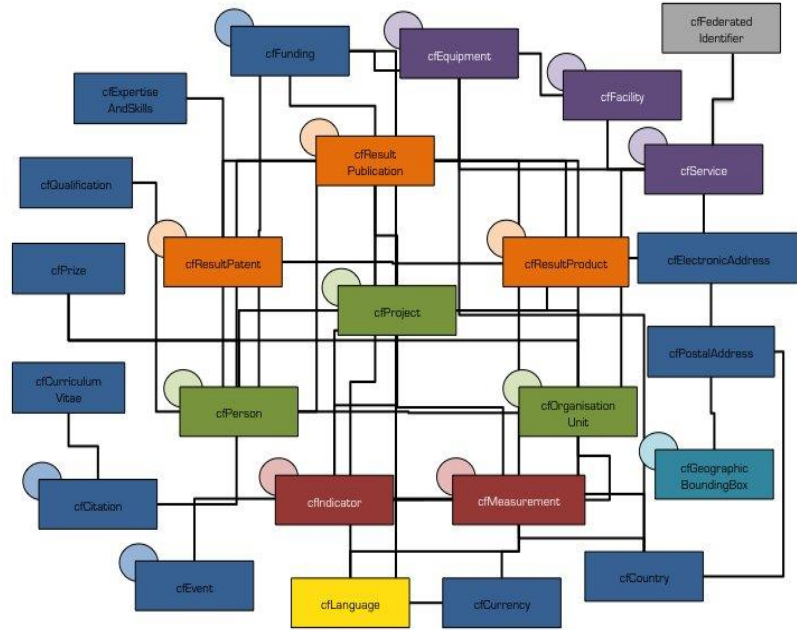
Managing DOIs within DSpace-CRIS 7

- DSpace-CRIS? What is it?
- DOI & DSpace-CRIS
 - Configurations






DSpace-CRIS? What is it? Extended data model

- Extended Data Model as a core feature for mapping the whole research domain
- Based on the OpenAire information space
- Pre-configured features and workflows to support to broad scope of a modern repository: publications, datasets, projects, researchers profiles, organizations, etc.



DSpace-CRIS what is it? At the Core of the Research Ecosystem

- ROR integration  ROR
- Full ORCID integration  ORCID
- DataCite integration  DataCite

DOI & DSpace-CRIS

- Integration built on top of DSpace integration
[DOI Digital Object Identifier - DSpace 7.x Documentation - LYRISIS Wiki](#)
- **On DSpace-CRIS, then it's «only a matter of configuration»:**
dspace/config/spring/api/identifier-service.xml
 - Customizing the DOI namespace generation
 - Exposing ROR and ORCID to DataCite (connecting research outputs to further entities)
 - Compliancy with DataCite Metadata Schema 4.5
- Import of metadata from DataCite available since DSpace-CRIS 2023.01.00 (will be in DSpace 8)

Configurations: configurable DOI namespace generation

```
<!-- Namespace definition -->
<bean id="defaultValueNamespace" class="org.dspace.identifier.generators.FixedConfigurationValueNamespaceGenerator">
  <property name="configurationValue" value="{identifier.doi.namespaceseparator}" />
</bean>

<bean id="unitsCustomNamespace" class="org.dspace.identifier.generators.FixedConfigurationValueNamespaceGenerator">
  <property name="configurationValue" value="units/custom/" />
</bean>

<bean id="customSeparatorNamespace" class="org.dspace.identifier.generators.MetadataValueNamespaceGenerator">
  <constructor-arg name="postValue" value="/" />
  <constructor-arg name="namespaceSeparator" value="{identifier.doi.namespaceseparator}" />
  <constructor-arg name="metadataFields">
    <set>
      <value>dc.identifier.issn</value>
      <value>dc.identifier.eissn</value>
      <value>dc.identifier.isbn</value>
      <value>dc.identifier.eisbn</value>
    </set>
  </constructor-arg>
</bean>

<!-- Below there are defined the generation strategy -->
<bean id="defaultDoiGenerationStrategy" class="org.dspace.identifier.generators.ConfigurableDoiGenerationStrategy">
  <constructor-arg name="filter" ref="always_true_filter" />
  <constructor-arg name="doiNamespaceGenerator" ref="defaultValueNamespace" />
  <constructor-arg name="generationType" value="DEFAULT" />
</bean>

<bean id="communityDoiGenerationStrategy" class="org.dspace.identifier.generators.ConfigurableDoiGenerationStrategy">
  <constructor-arg name="filter" ref="belongsToComm" />
  <constructor-arg name="doiNamespaceGenerator" ref="unitsCustomNamespace" />
  <constructor-arg name="generationType" value="CUSTOM" />
</bean>

<bean id="collectionDoiGenerationStrategy" class="org.dspace.identifier.generators.ConfigurableDoiGenerationStrategy">
  <constructor-arg name="filter" ref="belongsToColl" />
  <constructor-arg name="doiNamespaceGenerator" ref="customSeparatorNamespace" />
  <constructor-arg name="generationType" value="CUSTOM" />
</bean>
```

Examples

Giulio Monteverde as a master portraitist

Gardonio, Matteo

2018



ISSN 1827-269X

e-ISSN 2499-6750

DOI 10.13137/2499-6750/22490

<http://hdl.handle.net/10077/22490>

Book

Umanità in mostra. Esposizioni etniche e invenzioni esotiche in Italia (1880-1940). Seconda edizione riveduta e accresciuta

ABBATTISTA, GUIDO

2021



ISBN 978-88-5511-265-9

e-ISBN 978-88-5511-266-6

DOI 10.13137/978-88-5511-266-6/32371

<http://hdl.handle.net/10077/32371>

Book

Configurations: DOI generation strategies

```
<bean id="publicationCollectionCond" class="org.dspace.content.logic.condition.InCollectionCondition">
  <property name="parameters">
    <map>
      <entry key="collections">
        <list>
          <value>123456789/6</value>
        </list>
      </entry>
    </map>
  </property>
</bean>
<bean id="publicationFilter" class="org.dspace.content.logic.DefaultFilter">
  <property name="statement" ref="publicationCollectionCond"/>
</bean>

<bean id="publicationGenerator" class="org.dspace.identifier.generators.FixedConfigurationValueNamespaceGenerator">
  <property name="configurationValue" value="publications/dspace-local/" />
</bean>

<bean id="thesisDoiGenerationStrategy" class="org.dspace.identifier.generators.ConfigurableDoiGenerationStrategy">
  <constructor-arg name="filter" ref="publicationFilter" />
  <constructor-arg name="doiNamespaceGenerator" ref="publicationGenerator" />
  <constructor-arg name="generationType" value="CUSTOM" />
</bean>
```

Configurations: DOI templates

dspace > config > crosswalks > template > publication-datacite-xml.template

```
1 <?xml version="1.0" encoding="utf-8"?>
2 {
3 <resource xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://datacite.org/schema/kernel-4" xsi:schemaLocation="http://datacite.org/schema/kernel-4 http://schema.datacite.org/meta/kernel-4.5/metadata.xsd">
4 <identifier identifierType="DOI">@virtual.primary-doi.dc-identifier-doi@</identifier>
5 <creators>
6 |
7 | @group.dc-contributor-author.start@
8 | <creator>
9 | | <<creatorName>@dc.contributor.author@</creatorName>
10 | | @relation.dc-contributor-author.start@
11 | | <nameIdentifier nameIdentifierScheme="ORCID" schemeURI="http://orcid.org/">@person.identifier.orcid@</nameIdentifier>
12 | | @relation.dc-contributor-author.end@
13 | | @relation.oairecerif-author-affiliation.start@
14 | | <affiliation>
15 | | | @if.metadata.organization-identifier-ror.start@
16 | | | | affiliationIdentifier="@organization.identifier.ror@" affiliationIdentifierScheme="ROR"
17 | | | @if.metadata.organization-identifier-ror.end@
18 | | | >dc.title@</affiliation>
19 | | | @relation.oairecerif-author-affiliation.end@
20 | | | @if.not.authority.oairecerif-author-affiliation.start@
21 | | | <affiliation>@oairecerif.author.affiliation@</affiliation>
22 | | | @if.not.authority.oairecerif-author-affiliation.end@
23 | | </creator>
24 | | @group.dc-contributor-author.end@
25 | </creators>
26 <titles>
27 | <title>@dc.title@</title>
28 </titles>
29 <publisher>@dc.publisher@</publisher>
30 <publicationYear>@virtual.date.dc-date-issued.YYYY@</publicationYear>
31 <subjects>
32 | <subject>@dc.subject@</subject>
33 </subjects>
34 <dates>
35 | <date dateType="Issued">@dc.date.issued@</date>
36 | <date dateType="Available">@datacite.available@</date>
37 </dates>
38 </resource>
39 }
```

DataCite 4.5

ORCID

ROR

Template based on DSpace-CRIS Refer Crosswalk Engine

Configurations: Item Export

- In order to generate the metadata deposited in Datacite, DSpace-CRIS uses the Item Export functionality and the ReferCrosswalk instead than XSLT
- Easier to debug and immediate to check also performing an export from the UI
- The item export functionality allows users to export the metadata of one or multiple items in a specific format among those configured. Based on the type and number of entities to be exported, it is possible to obtain results with different formats (XML, JSON, PDF, CSV etc...).
- Single item export and the Multiple items export.

Configurations: Item Export

- With the same type of entity to be exported, the cardinality of the items to be exported (one or many) can affect the available export result formats.
- If you decide to export a single researcher profile the available formats could be XML, JSON, PDF or RTF, while if you choose to export many profiles you could do it in XML, JSON, CSV and XLS.
- The available formats are not static but can be configured: it is possible to establish both the information to export and the structure of the file itself. The configuration of export formats is based on a series of editable text files that act as templates.

StreamDisseminationCrosswalk and ItemExportCrosswalk

- The export logic is implemented by a set of classes that implement the **ItemExportCrosswalk** interface and that serialize in a specific format one or multiple items.
- This interface extends **StreamDisseminationCrosswalk** interface and it allows to distinguish the classes that can be used for exporting the item in the various formats available.

StreamDisseminationCrosswalk and ItemExportCrosswalk

The current implementations used by the export functionality to obtain the items in a specific format are:

- **ReferCrosswalk:** Generates a textual representation of the item/items starting from a template file in which there is a set of placeholders.
- **DocumentCrosswalk:** Generates a document starting from the chosen item in the configured format (such as PDF or RTF). This implementation is based on an XSL transformation made from a template file written with the XSL-FO (Formatting Objects) language.
- **TabularCrosswalk:** Abstract implementation that, starting from the items chosen for export, generates a table structure with configurable headings starting from a template file. The actual format of the table is determined by the classes that extend this abstract class. Currently available implementations are:
 - **XlsCrosswalk:** the data of the items in the tabular form is written into an xls file
 - **CsvCrosswalk:** items metadata are exported in csv format
- **CSLItemDataCrosswalk:** Generates textual representation using the Citation Style Language (CSL), an XML-based format to describe the formatting of citations, notes and bibliographies.

Refer Crosswalk

- The **ReferCrosswalk** allows to serialize the metadata of an item in a textual format that mirrors that of the configured template
- Examples of properties to be configured are:
 - **templateFileName**: the path of the template to use relative to the DSpace configuration folder
 - **contentType**: the format of the file obtained by processing the item; it should be consistent with the configured template.
 - **fileName**: the default name of the file that can be generated starting from the ReferCrosswalk result
 - **entityType**: the type of the items that can be processed by this instance of the ReferCrosswalk

Refer Crosswalk

- The template that is used to produce the result in a given format is a text file in which can be placed a series of placeholders: the result of the process corresponds to a file similar to the template in which the data relative to the processed items are inserted instead of these placeholders.
- Each line of the template can contain at most one placeholder; in case one line does not contain a placeholder, this line will be reported identical in the generated result. If a placeholder needs to be replaced by multiple values (for example due to multiple values of a metadata) the entire row is duplicated for each value to be written.

Refer Crosswalk

- The placeholders are marked with the @ and depending on the type the effect on the output may be different. There are 5 type of placeholders:
 - **metadata:** can be used to indicate that the specified metadata value must be entered instead of the placeholder. The syntax of this placeholder is **@<metadata-field>@**, where <metadata-field> represents a metadata field with the various sections divided by a period. Examples: @dc.title@, @dc.date.issued@

```
<titles>  
  <title>@dc.title@</title>  
</titles>
```

Refer Crosswalk

- **metadata-group**: placeholder with which some lines of the template can be delimited to indicate that the whole section must be repeated for each set of nested metadata identified. The syntax of this placeholder is **@group.<metadata-field>.start@** to delimit the beginning of the section to be replicated and **@group.<metadata-field>.end@** to indicate the end, where <metadata-field> is the metadata representing the group with its various sections separated by “-”

Refer Crosswalk

```
@group.dc-contributor-author.start@
<creator>
  <creatorName>@dc.contributor.author@</creatorName>
  @relation.dc-contributor-author.start@
  <nameIdentifier nameIdentifierScheme="ORCID" schemeURI="http://orcid.org/">@person.identifier.orcid@</nameIdentifier>
  @relation.dc-contributor-author.end@
  @relation.oairecerif-author-affiliation.start@
  <affiliation
    affiliationIdentifier="@organization.identifier.ror@" affiliationIdentifierScheme="ROR" schemeURI="https://ror.org/"
  >@dc.title@</affiliation>
  @relation.oairecerif-author-affiliation.end@
  @if.not.authority.oairecerif-author-affiliation.start@
  <affiliation>@oairecerif.author.affiliation@</affiliation>
  @if.not.authority.oairecerif-author-affiliation.end@
</creator>
@group.dc-contributor-author.end@
```

Refer Crosswalk

- **virtual field**: placeholder to be replaced with the results of the specified virtual field. The syntax of this placeholder is **@virtual.<name>.<qualifiers>@**, where name represents the virtual field identifiers and the qualifiers represents a set of info useful for the virtual field processing divided by period.

```
<resourceType resourceTypeGeneral="@virtual.mapConverter.type2datacite.dc-type@">  
@virtual.mapConverter.type2datacitelabel.dc-type@  
</resourceType>
```

Refer Crosswalk

- **relation**: placeholder with which some lines of the template can be delimited to indicate that the whole section must be repeated for each item which has a specific relationship with the item being written. The syntax of this placeholder is **@relation.<relationName>.start@** to delimit the beginning of the section to be replicated and **@relation.<relationName>.end@** to indicate the end, where <relationName> could be:
 - the metadata that contains the relationship with the other item through authority, with the various sections separated by “-”
 - the last section of one of the discovery configuration named **RELATION.<entityType>.<relationName>**, where entityType is the type of the item being written.

Refer Crosswalk

```
@relation.oairecerif-author-affiliation.start@  
  <affiliation
```

```
  affiliationIdentifier="@organization.identifier.ror@"  
  affiliationIdentifierScheme="ROR"  
  schemeURI="https://ror.org/"  
    >@dc.title@</affiliation>  
  @relation.oairecerif-author-affiliation.end@
```

Refer Crosswalk

- **if:** placeholder with which some lines of the template can be delimited to indicate that the whole section must be printed or not based on a condition evaluation. The syntax of this placeholder is **@if.[not.]<conditonName>.[qualifiers.]start@** to delimit the beginning of the section to be replicated and **@if.[not.]<condtionName>.[qualifiers.]end@** to indicate the end, where:
 - not is an optional section to negate the whole evaluation result
 - qualifiers are a set of data that can be used to evaluate the condition, separated by period
 - conditonName is the name of the particular condition evaluator to be used to evaluate the condition.

Refer Crosswalk

```
<dates>
  <date dateType="Issued">@dc.date.issued@</date>
  @if.metadata.datacite-available.start@
  <date dateType="Available">@datacite.available@</date>
  @if.metadata.datacite-available.end@
  @if.not.metadata.datacite-available.start@
  <date dateType="Available">@dc.date.available@</date>
  @if.not.metadata.datacite-available.end@
</dates>
```

Bug fixes and improvements to be donated to DSpace

- DOI Organizer: if the doi table contains hundreds or thousands of operations (create, update, delete) the process stucks, usually failing with an out-of-memory or taking too long: fixed introducing pagination
- The ItemServiceImpl doesn't flag the DOI of deleted item in the proper way in the doi table. This leads to subsequent failures when the doi-organiser -d command is executed: fixed delegating deletion to the identifier service

What's Next?

- DOI generation with "multiple prefixes" in the same DSpace-CRIS:
 - standard
 - thesis
 - report
 - etc.

