DSpace 7.x Documentation

DSpace 7.x Documentation

Exported on 08/03/2021

Table of Contents

1 Introduction

DSpace is an open source software platform that enables organisations to:

- capture and describe digital material using a submission workflow module, or a variety of programmatic ingest options
- distribute an organisation's digital assets over the web through a search and retrieval system
- preserve digital assets over the long term

This system documentation includes a functional overview of the system(see page 35), which is a good introduction to the capabilities of the system, and should be readable by non-technical folk. Everyone should read this section first because it introduces some terminology used throughout the rest of the documentation.

For people actually running a DSpace service, there is an installation guide(see page 53), and sections on configuration(see page 552) and the directory structure(see page 636). Support options are available in the DSpace Support Guide[1].

For those interested in the details of how DSpace works, and those potentially interested in modifying the code for their own purposes, there is a detailed architecture section(see page 652).

Other good sources of information are:

- The DSpace Support Guide[2] lists various places to ask for help, report bugs or security issues, etc.
- The DSpace REST API contract[3] which documents the REST API behavior, etc. If you want source code docs, we also provide JavaDocs for the Java API layer which can be built by running `mvn javadoc:javadoc`
- The DSpace Wiki[4] contains stacks of useful information about the DSpace platform and the work people are doing with it. You are strongly encouraged to visit this site and add information about your own work. Useful Wiki areas are:
  - A list of DSpace resources[5] (Web sites, mailing lists etc.)
  - Technical FAQ[6]
  - Registry of projects using DSpace[7]
  - Guidelines for contributing back to DSpace[8]
- www.dspace.org[9] has announcements and contains useful information about bringing up an instance of DSpace at your organization.
- The DSpace Community List[10]. Join DSpace-Community to ask questions or join discussions about non-technical aspects of building and running a DSpace service. It is open to all DSpace users. Ask questions, share news, and spark discussion about DSpace with people managing other DSpace sites. Watch DSpace-Community for news of software releases, user conferences, and announcements about DSpace.
- The DSpace Technical List[11]. DSpace developers & fellow community members help answer installation and technology questions, share information and help each other solve technical problems through the DSpace-Tech mailing list. Post questions or contribute your expertise to other developers working with the system.
- The DSpace Development List[12]. Join Discussions among DSpace Developers. The DSpace-Dev listserv is for DSpace developers working on the DSpace platform to share ideas and discuss code changes to the open source platform. Join other developers to shape the evolution of the DSpace software. The DSpace

1 https://wiki.lyrasis.org/display/DSPACE/Support
2 https://wiki.lyrasis.org/display/DSPACE/Support
3 https://github.com/DSpace/Rest7Contract/blob/main/README.md
4 http://wiki.dspace.org/
5 https://wiki.lyrasis.org/display/DSPACE/DSpaceResources
6 https://wiki.lyrasis.org/display/DSPACE/TechnicalFAQ
7 http://registry.duraspace.org/registry/dspace
8 https://wiki.lyrasis.org/display/DSPACE/Code+Contribution+Guidelines
9 http://www.dspace.org/
10 https://groups.google.com/d/forum/dspace-community
11 https://groups.google.com/d/forum/dspace-tech
12 https://groups.google.com/d/forum/dspace-devel

community depends on its members to frame functional requirements and high-level architecture, and to facilitate programming, testing, documentation and to the project.

1.1 Release Notes

Try out DSpace 7.0!

To try out DSpace 7.0 immediately, see Try out DSpace 7[13]. This includes instructions for a quick-install via Docker, as well as information on our sandbox/demo site for DSpace 7[14].

DSpace 7 includes a separate backend (REST API) & frontend (User Interface). Full installation instructions are available at Installing DSpace(see page 53).

- Download DSpace 7.0 Backend: https://github.com/DSpace/DSpace/releases/tag/dspace-7.0
- Download DSpace 7.0 User Interface: https://github.com/DSpace/dspace-angular/releases/tag/dspace-7.0

Upgrade from any past version of DSpace!

Installing DSpace(see page 53) provides an overview of the DSpace 7 installation process and all prerequisite software. You should review this before attempting an upgrade, in order to ensure you are running the required versions of Java, Node, etc.

Upgrading DSpace(see page 75) provides a guide for upgrading from *any old version* of DSpace to v7. As in the past, your data migrates automatically, no matter which older version you are running. *However, as the old XMLUI and JSPUI user interfaces are no longer supported, you must switch to using the new User Interface.*

13 https://wiki.lyrasis.org/display/DSPACE/Try+out+DSpace+7
14 https://demo7.dspace.org/

1.1.1 7.0 Release Notes

> Brief videos of some DSpace 7 features are available at https://duraspace.org/dspace/dspace-7/

DSpace 7.0 is the largest release in the history of DSpace software. While retaining the "out-of-the-box" aspects DSpace is known for, it represents a major evolution of the platform including:

- **A completely new User Interface (**demo site[15]**).** This is the new Javascript-based frontend, built on Angular.io[16] (with support for SEO provided by Angular Universal). This new interface is also customizable via HTML and CSS (Sass) and Bootstrap. For early theme building tips see User Interface Customization(see page 378)
- **A completely new, fully featured REST API (**demo site[17]**)**, provided via a single "server" webapp backend. This new backend is not only a REST API, but also still supports OAI-PMH, SWORD (v1 or v2) and RDF. Anything you can do from the User Interface is now also possible in our REST API. See REST API(see page 502) documentation for more details.
- **A newly designed search box.** Search from the header of any page (click the magnifying glass). The search results page now features automatic search highlight, expandable & searchable filters, and optional thumbnail-based results (click on the “grid” view).
- **A new MyDSpace area to manage your submissions & reviews**, MyDSpace includes a new drag & drop area to start a new submission, and easily search your workflow tasks or in progress submissions to find what you were working on. (Login, click on your user profile icon, click “MyDSpace”). Find workflow tasks to claim by selecting “All tasks” in the “Show” dropdown.
- **A new configurable submission user interface,** featuring a one-page, drag & drop submission form. This form is completely configurable and can be prepopulated by dragging & dropping a metadata file (e.g. ArXiv, CSV/TSV, Endnote, PubMed, or RIS. etc) or by importing via external APIs (e.g ORCID, PubMed, Sherpa Journals or Sherpa Publishers, etc) (video[18]). Local controlled vocabularies are also still supported (video[19]). See Submission User Interface(see page 260) for more details.
- **Optional, new** Configurable Entities(see page 134) **feature**. DSpace now supports “entities”, which are DSpace Items of a specific ‘type’ which may have relationships to other entities. These entity types and relationships are configurable, with two examples coming out-of-the-box: a set of Journal hierarchy entities (Journal, Volume, Issue, Publication) and a set of Research entities (Publication, Project, Person, OrgUnit). For more information see Configurable Entities(see page 134).
- **Dynamic user interface translations** (Click the globe, and select a language). Interested in adding more translations? See DSpace 7 Translation - Internationalization (i18n) - Localization (l10n)[20].
- **A new Admin sidebar.** Login as an Administrator, and an administrative sidebar appears. Features available include:
    - Quickly create or edit objects from anywhere in the system. Either browse to the object first, or search for it using the Admin sidebar.
    - Processes UI (video[21]) allows Administrators to run backend scripts/processes while monitoring their progress & completion. (Login as an Admin, select "Processes" in sidebar)
    - Administrative Search (video[22]) combines retrieval of withdrawn items and private items, together with a series of quick action buttons.

15 https://demo7.dspace.org/
16 http://Angular.io
17 https://api7.dspace.org/server/
18 https://www.youtube.com/watch?v=mGRDl0khzrQ
19 https://youtu.be/OfEIoxOJK-8
20 https://wiki.lyrasis.org/pages/viewpage.action?pageId=117735441
21 https://www.youtube.com/watch?v=vcsWkWQONkY
22 https://www.youtube.com/watch?v=JV8Rb-9cByo&t=1s

    - Administer Active Workflows (video[23]) allows Administrators to see every submission that is currently in the workflow approval process.
    - Bitstream Editing (video[24]) has a drag-and-drop interface for re-ordering bitstreams and makes adding and editing bitstreams more intuitive.
    - Metadata Editing (video[25]) introduces suggest-as-you-type for field name selection of new metadata.
    - Login As (Impersonate) another account allows Administrators to debug issues that a specific user is seeing, or do some work on behalf of that user. (Login as an Admin, Click "Access Control" in sidebar, Click "People". Search for the user account & edit it. Click the "Impersonate EPerson" button. You will be authenticated as that user until you click "Stop Impersonating EPerson" in the upper right.)
- **Improved GDPR alignment** (video[26])
    - User Agreement required for all authenticated users to read and agree to. (Login for first time, and sample user agreement will display. After agreeing to it, it will not appear again.)
    - Cookie Preferences are now available for all users (anonymous or authenticated). A cookie preference popup appears when first accessing the site. Users are given information on what cookies added by DSpace, including a Privacy Statement which can be used to describe how their data is used.
    - User Accounts can be deleted *even if they've submitted content* in the past.
- **Support for OpenAIREv4** Guidelines for Literature Repositories[27] in OAI-PMH (See the new “openaire4” context in OAI-PMH).
- **Search Engine Optimization**: Tested and approved by the Google Scholar team, DSpace still includes all the SEO features you require: a robots.txt, Sitemaps and Google Scholar "citation" tags.
- **Video/Image Content Streaming** *(Kindly donated by* Zoltán Kanász-Nagy[28] *and* Dániel Péter Sipos[29] *of Qulto):* When enabled, DSpace can now stream videos & view images full screen, using an embedded viewer. (See the "mediaViewer" settings in the environment.common.ts[30] to enable.)
- **Basic Usage Statistics** (video[31]) are available for the entire site (See "Statistics" menu at top of homepage), or specific Communities, Collections or Items (Click on that same "Statistics" menu after browsing to a specific object
- Additional features are listed in the Beta release notes below. Also, give it a try on our demo site[32] & see what you discover!

> **DSpace 7 does not yet include all the features of DSpace 6.x**
>
> DSpace 7.0 represents a major evolution of the platform into a new, modern web architecture. This means there are tons of new and redesigned features in 7.0. However, in order to get this release in your hands sooner, DSpace Steering decided to delay some 6.x features for later 7.x releases. So, if you don't see a 6.x feature yet in 7.0, it'll likely be coming soon in a later 7.x release. For a prioritized list of upcoming features see "What features are coming in a later 7.x release?" on our DSpace Release 7.0 Status[33] page.

Additional major changes to be aware of in the 7.x platform (not an exhaustive list):

---

23 https://www.youtube.com/watch?v=CjH8VS2WDjE
24 https://youtu.be/s1msEKK0f68
25 https://youtu.be/6KVB2ugUgjI
26 https://www.youtube.com/watch?v=bKqFmb6Ywng
27 https://guidelines.openaire.eu/en/latest/literature/index.html
28 https://github.com/kanasznagyzoltan
29 https://github.com/dsipos-dev
30 https://github.com/DSpace/dspace-angular/blob/main/src/environments/environment.common.ts#L273-L276
31 https://youtu.be/T2g74zs_wmM
32 https://demo7.dspace.org/
33 https://wiki.lyrasis.org/display/DSPACE/DSpace+Release+7.0+Status

- **XMLUI and JSPUI are no longer supported or distributed with DSpace.** All users should immediately migrate to and utilize the new Angular User Interface[34]. There is no migration path from either the XMLUI or JSPUI to the new User interface. However, the new user interface can be themed via HTML and CSS (SCSS).
- **The old REST API ("rest" webapp from DSpace v4.x-6.x) is deprecated and will be removed in v8.x.** The new REST API(see page 502) (provided in the "server" webapp) replaces all functionality available in the older REST API. If you have tools that rely on the old REST API, you can still (optionally) build & deploy it alongside the "server" webapp via the "-Pdspace-rest" Maven flag. See REST API v6 (deprecated)(see page 506)
- **The Submission Form configuration has changed.** The "item-submission.xml" file has changed its structure, and the "input-forms.xml" has been replaced by a "submission-forms.xml". See Submission User Interface(see page 260)
- **ElasticSearch Usage Statistics have been removed.** Please use SOLR Statistics(see page 338) or DSpace Google Analytics Statistics(see page 363).
- **The traditional, 3-step Workflow system has been removed in favor of the** Configurable Workflow System(see page 250)**.** For most users, you should see no effect or difference. The default setup for this Configurable Workflow System is identical to the traditional, 3-step workflow ("Approve/Reject", "Approve/Reject/Edit Metadata", "Edit Metadata")
- **The old BTE import framework in favor of** Live Import Framework(see page 274) (features of BTE have been ported to Live Import)
- **Apache Solr is no longer embedded within the DSpace installer.** Solr now MUST be installed as a separate dependency alongside the DSpace backend. See Installing DSpace(see page 53).
- **A large number of old/obsolete configurations were removed.** "7.0 Configurations Removed" section below.
- See Upgrading DSpace(see page 75) for more hints on the upgrade from any old version of DSpace to 7.x

Additional Resources

- Video presentations / Workshops from OR2021 (June 2021) showing off many of the new features & configurations of DSpace 7: DSpace 7 at OR2021[35]

1.1.2 7.0 Configurations Removed

With the removal of the JSPUI and XMLUI, a large number of server-side (backend) configurations were made obsolete and were therefore removed between the 6.x and 7.0 release. Those Configurations removed included:

- Within the `[dspace]/config/` directory, these are the **configuration files which were deleted:**
  - `dc2mods.cfg`
  - `input-forms.xml / dtd` (*REPLACED BY submission-forms.xml, see* Submission User Interface(see page 260))
  - `log4j.properties` (REPLACED BY `log4j2.xml`)
  - `log4j-console.properties` (REPLACED BY `log4j-console.xml`)
  - `log4j-solr.properties` (no replacement as Solr now must be installed separately)
  - `news-side.html`
  - `news-top.html`
  - `news-xmlui.xml`
  - `workflow.xml` (REPLACED BY `./spring/api/workflow.xml`)
  - `xmlui.xconf / dtd`
  - `emails/bte_*` (BTE import framework was removed in favor of Live Import from external sources(see page 274))
  - `modules/controlpanel.cfg`
  - `modules/elastic-search-statistics.cfg` (Elastic Search support was removed in favor of Solr)
  - `modules/fetchccdata.cfg`

34 https://github.com/DSpace/dspace-angular/
35 https://wiki.lyrasis.org/display/DSPACE/DSpace+7+at+OR2021

    - `modules/publication-lookup.cfg`
    - `spring/api/bte.xml` (BTE import framework was removed in favor of Live Import from external sources(see page 274))
    - `spring/oai/*` (OAI is now part of the backend "server webapp" and needs no separate configurations)
    - `spring/xmlui/*`
- Within the `dspace.cfg` main configuration file, the following **settings were removed:**
    - `log.init.config` (replaced by `log4j2.xml`)
    - `webui.submit.blocktheses`
    - `webui.submit.upload.html5`
    - `webui.submission.restrictstep.enableAdvancedForm`
    - `webui.submission.restrictstep.groups`
    - `webui.submit.enable-cc`
    - `webui.browse.thumbnail.*`
    - `webui.item.thumbnail.*`
    - `webui.preview.enabled`
    - `webui.strengths.show`
    - `webui.browse.author-field`
    - `webui.browse.author-limit`
    - `webui.browse.render-scientific-formulas`
    - `recent.submissions.*`
    - `webui.collectionhome.*`
    - `plugin.sequence.org.dspace.plugin.SiteHomeProcessor`
    - `plugin.sequence.org.dspace.plugin.CommunityHomeProcessor`
    - `plugin.sequence.org.dspace.plugin.CollectionHomeProcessor`
    - `plugin.sequence.org.dspace.plugin.ItemHomeProcessor`
    - `plugin.single.org.dspace.app.webui.search.SearchRequestProcessor`
    - `plugin.single.org.dspace.app.xmlui.aspect.administrative.mapper.SearchRequestProcessor`
    - `plugin.named.org.dspace.app.webui.json.JSONRequest`
    - `plugin.single.org.dspace.app.webui.util.StyleSelection`
    - `webui.bitstream.order.*`
    - `webui.itemdisplay.*`
    - `webui.resolver.*`
    - `webui.preferred.identifier`
    - `webui.identifier.*`
    - `webui.mydspace.*`
    - `webui.suggest.*`
    - `webui.controlledvocabulary.enable`
    - `webui.session.invalidate`
    - `itemmap.*`
    - `jspui.*`
    - `xmlui.*`
    - `mirage2.*`

A full list of all changes / bug fixes in 7.x is available in the Changes in 7.x(see page 697) section.

1.1.3 7.0 Acknowledgments

1.1.3.1 Major Contributing Institutions

The following institutions have been major code contributors to the DSpace 7 release (in general)

- Atmire[36] - also hosts/maintains DSpace 7 UI demo at https://demo7.dspace.org
- 4Science[37] - also hosts/maintains DSpace 7 REST demo at https://api7.dspace.org/server/
- FCT[38] / RCAAP[39]

1.1.3.2 Financial Contributors

We gratefully recognize the following institutions who together have generously contributed financially to support the DSpace 7 staged release program (see DSpace 7 Release Goals[40]), and individuals who devoted time to fundraising:

- Auburn University
- Cornell University
- Pascal Becker
- Dalhousie University
- Duke University
- ETH Zurich, ETH Library
- Fraunhofer Gesellschaft
- Imperial College London
- Indiana University–Purdue University, Indianapolis
- LYRASIS
- National Library of Finland
- Beate Rajski
- Staats- und Universitätsbibliothek Hamburg – Carl von Ossietzky
- Technische Universität Berlin
- Technische Universität Hamburg (TUHH)
- The DSpace-Konsortium Deutschland
- The Helmut-Schmidt-Universität/Universität der Bundeswehr Hamburg
- The Library Code GmbH
- The Ohio State University
- Texas Digital Library
- University of Arizona
- University of Edinburgh
- University of Kansas
- University of Minnesota
- University of Missouri
- University of Toronto
- World Bank
- ZHAW

36 https://www.atmire.com/
37 https://www.4science.it/
38 https://www.fct.pt/
39 https://www.rcaap.pt/
40 https://wiki.lyrasis.org/display/DSPACE/DSpace+7+Release+Goals

1.1.3.3 Frontend / User Interface Acknowledgments

The following 55 individuals have contributed directly to the new DSpace (Angular) User Interface in this release (*ordered by number of GitHub commits*): Giuseppe Digilio (atarix83), Kristof De Langhe (Atmire-Kristof), Lotte Hofstede (LotteHofstede), Art Lowel (artlowel), Marie Verdonck (MarieVerdonck), Julius Gruber (Flusspferd123), Yura Bondarenko (ybnd), William Welling (wwelling and wellingWilliam), Yana De Pauw (YanaDePauw), Tim Donohue (tdonohue), Alessandro Martelli (alemarte), Michael Spalti (mspalti), Jonas Van Goolen (jonas-atmire), Laura Henze (lhenze), Dániel Péter Sipos (dsipos-dev), Samuel Cambien (samuelcambien), Bruno Roemers (bruno-atmire), Matteo Perelli (sourcedump), Bram Luyten (bram-atmire), Ben Bosman (benbosman), Terry Brady (terrywbrady), Raf Ponsaerts (Raf-atmire), Danilo Di Nuzzo (ddinuzzo), Andrea Chiapparelli (andreachiapparelli), Antoine Snyers (antoine-atmire), Corrado Lombardi (corrad82-4s), Courtney Pattison (courtneypattison), Àlex Magaz Graça (rivaldi8), Chris Wilper (cwilper), Christian Scheible (christian-scheible), Andrew Wood (AndrewZWood), Reeta Kuuskoski (reetagithub), Vítor Silvério Rodrigues (vitorsilverio), Alexander Sulfrian (AlexanderS), muiltje, José Carvalho (josekarvalho), Claudia Jürgen (cjuergen), fernandaruizm, Ivan Masar (helix84), Paulo Graça (paulo-graca), Philip Vissenaekens (PhilipVis), Nagy Akos (akoscomp), Kevin Van de Velde (KevinVdV), Sascha Szott (saschaszott), Mohamed Mohideen Abdul Rasheed (mohideen), David Cavrenne (davidatmire), Hardy Pottinger (hardyoyo), Luca Giamminonni (LucaGiamminonni), Mateus Mercer (MatMercer), Denijs Balodis (Denijsb), Pascal-Nicolas Becker (pnbecker), Mikus Zarins (MixonZ), marciofoz, Andrea Bollini (abollini), Martin Walk (MW3000).

Out of the above list, the following individuals contributed a translation of the new interface (ordered alphabetically by language): Ivan Masar (Czech), Marina Muilwijk (Dutch), Reeta Kuuskoski (Finnish), David Cavrenne (French), Claudia Jürgen and Sasha Szott (German), Nagy Akos and Transylvanian Museum Society (Hungarian), Mikus Zarins (Latvian), Vítor Silvério Rodrigues and marciofoz (Brazilian Portuguese), José Carvalho (Portuguese) and Maria Fernanda Ruiz (Spanish).

The above contributor lists were determined based on historical contributions to the "dspace-angular" project in GitHub until 7.0: https://github.com/DSpace/dspace-angular/graphs/contributors?from=2016-11-27&to=2021-07-29&type=c

1.1.3.4 Backend / REST API Acknowledgments

The following 55 individuals have contributed directly to the DSpace backend (REST API, Java API, OAI-PMH, etc) in this release (*ordered by number of GitHub commits*): Raf Ponsaerts (Raf-atmire), Tim Donohue (tdonohue), Andrea Bollini (abollini), Michele Boychuk (Micheleboychuk), Mark Wood (mwoodiupui), Marie Verdonck (MarieVerdonck), Ben Bosman (benbosman), Luigi Andrea Pascarelli (lap82), Terry Brady (terrywbrady), Tom Desair (tomdesair), Yana De Pauw (YanaDePauw), Chris Wilper (cwilper), Peter Nijs (peter-atmire), Kevin Van de Velde (KevinVdV), Bruno Roemers (bruno-atmire), Giuseppe Digilio (atarix83), Pasquale Cavallo (pasqualecvl), Jelle Pelgrims (jpelgrims-atmire), Andrew Wood (AndrewZWood), Samuel Cambien (samuelcambien), Antoine Snyers (antoine-atmire), Kim Shepherd (kshepherd), Yura Bondarenko (ybnd), Michael Spalti (mspalti), Alessandro Martelli (alemarte), Oliver Goldschmidt (olli-gold), Jonas Van Goolen (jonas-atmire), Kristof De Langhe (Atmire-Kristof), Alexander Sulfrian (AlexanderS), Patrick Trottier (PTrottier), Pablo Prieto (ppmdo), Hardy Pottinger (hardyoyo), Pascal-Nicolas Becker (pnbecker), William Tantzen (tantz001), Paulo Graça (paulo-graca), Luca Giamminonni (LucaGiamminonni), Ivan Masar (helix84), Hrafn Malmquist (J4bbi), Ian Little (ilittle-cnri), Anis Moubarik (anis-moubarik), Claudia Jürgen (cjuergen), Alan Orth (alanorth), xuejiangtao, Danilo Di Nuzzo (ddinuzzo), James Creel (jcreel), Marsa Haoua (marsaoua), Philip Vissenaekens (PhilipVis), Miika Nurminen (minurmin), Bram Luyten (bram-atmire), Christian Scheible (christian-scheible), Nicholas Woodward (nwoodward), József Marton (jmarton), Mohamed Mohideen Abdul Rasheed (mohideen), Saiful Amin (saiful-semantic), Àlex Magaz Graça (rivaldi8)

The above contributor list was determined based on contributions to the "DSpace" project in GitHub between 6.0 (after Oct 24, 2016) and 7.0: https://github.com/DSpace/DSpace/graphs/contributors?from=2016-10-24&to=2021-07-29&type=c Therefore this list may include individuals who contributed to later 6.x releases, but *only if their bug fix was also applied to 7.0*.

1.1.3.5 Additional Thanks

Additional thanks to our DSpace Leadership Group[41] and DSpace Steering Group[42] for their ongoing DSpace support and advice. Thanks also to LYRASIS[43] for your leadership, collaboration & support in helping to speed up the development process of DSpace 7.

Thanks also to the various developer & community Working Groups who have worked diligently to help make DSpace 7 a reality. These include:

- DSpace 7 Working Group[44] - This is the team behind the code
- DSpace 7 Entities Working Group[45] - This team designed & implemented Configurable Entities(see page 134)
- DSpace 7 Marketing Working Group[46] - This team did all our DSpace 7marketing, press releases & announcements.
- DSpace Community Advisory Team[47] (DCAT) - This team helped organize/lead the DSpace 7.0 Testathon[48] (to bang on the system to find any last bugs), and they also provided us with advice on features, etc.

We apologize to any contributor accidentally left off this list. DSpace has such a large, active development community that we sometimes lose track of all our contributors. Acknowledgments to those left off will be made in future releases.

1.1.4 7.0 Beta 1-5 Release Notes

DSpace 7.0 was developed via a series of Beta releases from 2020-21. The release notes for each Beta are retained here for reference.

1.1.4.1 7.0 Beta 5 Release Notes

Released April 2021

Included in Beta 5

- **Support for custom theme(s) in UI & accessibility cleanup of base theme**. See early information at DSpace UI Design principles and guidelines[49] and the "themes" section of the environment.common.ts[50]
  - Updated the "base" theme (default Bootstrap look & feel) for consistency and better accessibility. (Additional accessibility analysis will be performed during Testathon)
  - Added a simple "dspace" theme (this is the new default theme, and primarily shows an example of customizing color scheme & homepage)
  - Added a "custom" theme folder with all necessary files. These files can be directly modified to create a completely custom theme.
- **Major performance improvements to UI** by making better use of caching & smart reloading

41 https://duraspace.org/dspace/community/leadership-group/
42 https://duraspace.org/dspace/community/dspace-steering-group/
43 https://www.lyrasis.org/
44 https://wiki.lyrasis.org/display/DSPACE/DSpace+7+Working+Group
45 https://wiki.lyrasis.org/display/DSPACE/DSpace+7+Entities+Working+Group
46 https://wiki.lyrasis.org/display/DSPACE/DSpace+7+Marketing+Working+Group
47 https://wiki.lyrasis.org/display/cmtygp/DSpace+Community+Advisory+Team
48 https://wiki.lyrasis.org/display/DSPACE/DSpace+Release+7.0+Testathon+Page
49 https://wiki.lyrasis.org/display/DSPACE/DSpace+UI+Design+principles+and+guidelines
50 https://github.com/DSpace/dspace-angular/blob/main/src/environments/environment.common.ts#L230-L267

- **Video/Image Content Streaming** *(Kindly donated by* Zoltán Kanász-Nagy[51] *and* Dániel Péter Sipos[52] *of Qulto):* When enabled, DSpace can now stream videos & view images full screen, using an embedded viewer.
    - See the new "mediaViewer" settings in the environment.common.ts[53] to enable. S[54]ample screenshots of the feature can also be found at https://github.com/DSpace/dspace-angular/issues/885
- New Administrative Features
    - Add ability to modify Community/Collection resource policies (i.e. permissions). Edit a Community or Collection and look at the "Authorizations" tab.
    - Add ability to edit/delete user Groups.
    - Add private/withdrawn item badges for Administrators to quickly see which Items are private or withdrawn. These are viewable throughout the browse/search when logged in as an Administrative user.
- Configurable Entities Improvements
    - Entities now report their Entity type in the URL path (e.g. Person entities use URL path /entities/person/[uuid] and Publication entities use the URL path /entities/publication/[uuid])
    - Each Entity type now has a custom Submission form.
        - These can be most easily seen in the Demo site. Submitting to the "People" collection[55] uses the "Person" Entity Form. Submitting to the "Articles" collection[56] uses the "Publication" Entity Form. The full list of Entity-specific Collection submission mappings can be found in the example in item-submission.xml[57] (this example is enabled on our Demo Site)
    - General performance improvements for Entities. Introduction of "tilted" relationships[58] for Configurable Entities that may have hundreds or thousands of relationships.
- Improvements to Upgrade process
    - Added a new Submission form migration script[59] to help DSpace 5/6 institutions migrate their old Submission configuration files to the new/updated format for v7.
- Security fixes
    - Added CSRF (Cross-Site Request Forgery)[60] protection to REST API. UI (and any other clients) now must be *trusted* to login to the REST API.
    - Improved permissions checks/validation in UI for Administrator, Community/Collection Administrator and Submitter roles.
    - Fixed several other security issues auto-reported by LGTM[61]
- Many bug fixes
    - Fixed issue where mapped items were not appearing
    - Fixed issue where Handles were not redirecting
    - Fixed issues with Sherpa and ORCID integrations
    - Fixed several small issues with OpenAIRE v4 support in OAI-PMH
    - Fixed many bugs in MyDSpace and Submission UI
    - Fixed several bugs in CSV import/export process.
    - Fixes to search/browse pagination & breadcrumb trail
    - Improved performance of Browse by Community/Collection hierarchy
    - LDAP Authentication support is working again
- Many dependency upgrades

---

51 https://github.com/kanasznagyzoltan
52 https://github.com/dsipos-dev
53 https://github.com/DSpace/dspace-angular/blob/main/src/environments/environment.common.ts#L273-L276
54 https://github.com/DSpace/dspace-angular/pull/888
55 https://demo7.dspace.org/collections/9398affe-a977-4992-9a1d-6f00908a259f
56 https://demo7.dspace.org/collections/282164f5-d325-4740-8dd1-fa4d6d3e7200
57 https://github.com/DSpace/DSpace/blob/main/dspace/config/item-submission.xml#L23-L49
58 https://github.com/DSpace/DSpace/pull/3134
59 https://github.com/DSpace/DSpace/pull/3076
60 https://owasp.org/www-community/attacks/csrf
61 https://lgtm.com/

    - Upgrade UI to Angular v10
    - Upgrade UI to Node v12 or v14 support
    - Upgrade Backend to Solr v8 support
    - Upgrade to ORCID v3 support
    - Upgrade to SHERPA v2 support
- Removal of obsolete features
    - Removal of the old BTE framework in favor of Live Import Framework(see page 274) (features of BTE have been ported to Live Import)
    - Removal of Traditional/Basic workflow in favor of Configurable Workflow(see page 250) (default workflow is still the same as in DSpace 6)

Changelog

- All User Interface changes: https://github.com/DSpace/dspace-angular/issues?q=is%3Aclosed+milestone%3A7.0beta5
- All Backend changes: https://github.com/DSpace/DSpace/issues?q=is%3Aclosed+milestone%3A7.0beta5

1.1.4.2 7.0 Beta 4 Release Notes

Released October 2020

Included in Beta 4

- **Live Import framework** (video[62]) support has been added to the Submission Form (and REST API `/api/integration/externalsources` endpoint)
    - Search an external site for works to import (From your MyDSpace page, click the "Import metadata from external source" button in upper right). Currently supports Library of Congress Names, ORCID, PubMed, Sherpa Journals or Sherpa Publishers.
    - Drag and drop a bibliographic file into Submission form or MyDSpace page to prepopulate metadata. Supported formats include ArXiv, CSV (or TSV), Endnote, PubMed, or RIS.
- **Controlled Vocabulary support** (video[63]) in Submission Form. Depending on the field configuration, this can include autocomplete of known terms (see default "Subject Keywords" field), dropdown support (see default "Type" field) and hierarchical tree views
    - Includes support for Controlled Vocabs, Authority Control and "Value-Pairs" (from submission configs)
- **Curation Tasks** are now supported via the Admin UI and the Processes UI. (Login as an Admin, select "Curation Tasks")
- **Import / Export metadata from/to CSV** (i.e. Batch Metadata Editing[64]) is now available from the Admin UI. (Login as an Admin, select "Export" > "Metadata", select "Import" > "Metadata")
- **Basic Usage Statistics** (video[65]) are available for the entire site (See "Statistics" menu at top of homepage), or specific Communities, Collections or Items (Click on that same "Statistics" menu after browsing to a specific object).
    - Support for exchanging usage data to IRUS[66] was added. See new "irus-statistics.cfg" and DS-626[67]
- Improved GDPR Alignment (video[68])
    - **User Agreement** required for all authenticated users to read and agree to. (Login for first time, and sample user agreement will display. After agreeing to it, it will not appear again.)

62 https://youtu.be/irL7RO1HhFU
63 https://youtu.be/OfEIoxOJK-8
64 https://wiki.lyrasis.org/display/DSDOC6x/Batch+Metadata+Editing
65 https://youtu.be/T2g74zs_wmM
66 https://irus.jisc.ac.uk/
67 https://jira.lyrasis.org/browse/DS-626
68 https://www.youtube.com/watch?v=bKqFmb6Ywng

    - **Cookie Preferences** are now available for all users (anonymous or authenticated). A cookie preference popup appears when first accessing the site. Users are given information on what cookies added by DSpace, including a **Privacy Statement** which can be used to describe how their data is used.
    - User Accounts can be deleted *even if they've submitted content* in the past.
        - When a user is deleted, their past submissions are kept but the submitter field is set to empty (null).
        - Users cannot be deleted if they are the only member of a workflow approval group. Admins must either delete that group first, or assign another member to the group. This ensures Workflows are kept even if a user account needs to be deleted.
- **Language preferences** are now kept for all users (anonymous or logged in). By default, DSpace will try to use your browser's preferred language (if found in Accept-Language header and a translation in that language exists). Users can override it by either saving a preferred language in their user profile, or by manually selecting a different language from the globe icon (upper right).
- **IP-based authorization** lets you restrict (or provide access to) objects based on the user's IP address. This uses the same "authentication-ip.cfg" configuration as DSpace 6, allowing you to map IP ranges to specific DSpace Groups. Users within that IP range are added to the mapped DSpace Group for the remainder of their session.
- **Search Engine Optimization**: Addition of robots.txt, Sitemaps and Google Scholar "citation" tags. These optimizations are being tested by the Google Scholar team and may be improved further in the upcoming beta 5 release.
    - For improved SEO, Sitemaps are now enabled by default and automatically update once per day.
- Security Fixes and Dependency upgrades
    - Enhancements to new `/api/authz/features` endpoint in REST API to provide additional feature-specific permission checks
    - Flyway[69] database engine was upgraded to version 6.5.5
    - Indexing enhancements (some objects were being indexed twice, see PR#2960[70])
    - Fixes to Shibboleth login
    - Additional bug fixes to both UI and REST API

Changelog

- All User Interface changes: https://github.com/DSpace/dspace-angular/issues?q=is%3Aclosed+milestone%3A7.0beta4
- All Backend changes: https://github.com/DSpace/DSpace/issues?q=is%3Aclosed+milestone%3A7.0beta4

1.1.4.3 7.0 Beta 3 Release Notes

Released July 2020

Included in Beta 3

- **Processes Admin UI** (video[71]) allows Administrators to run backend scripts/processes while monitoring their progress & completion. (Login as an Admin, select "Processes" in sidebar)
    - Currently supported processes include "index-discovery" (reindex site), "metadata-export" (batch metadata editing CSV export), and "metadata-import" (batch metadata editing CSV import).
- **Manage Account Profile** allows logged in users to update their name, language or password. (Login, click on the account icon, and select "Profile")
- **New User Registration** (video[72]) and password reset on the Login Screen

69 https://flywaydb.org/
70 https://github.com/DSpace/DSpace/pull/2960
71 https://www.youtube.com/watch?v=vcsWkWQONkY
72 https://www.youtube.com/watch?v=C2k8vn1rWNE

- **Login As (Impersonate)** another account allows Administrators to debug issues that a specific user is seeing, or do some work on behalf of that user. (Login as an Admin, Click "Access Control" in sidebar, Click "People". Search for the user account & edit it. Click the "Impersonate EPerson" button. You will be authenticated as that user until you click "Stop Impersonating EPerson" in the upper right.)
    - Requires "webui.user.assumelogin=true" to be set in your local.cfg on backend. Also be aware that you can only "impersonate" a user who is *not* a member of the Administrator group.
- **Manage Authorization Policies** of an Item allows Administrators to directly change/update the access policies of an Item, its Bundles or Bitstreams. (Login as an Admin, Click "Edit" → "Item" in sidebar, and search for the Item. Click the "Authorization.." button on its "Status" tab.
- **Manage Item Templates** of a Collection allows Administrators to create/manage template metadata that all new Items will start with when submitted to that Collection. (Login as an Admin, Click "Edit" → "Collection" in sidebar and search for the Collection. Click the "Add" button under "Template Item" to get started.)
    - NOTE: unfortunately there's a known bug that while you can create these templates, the submission process is not yet using them. See https://github.com/DSpace/dspace-angular/issues/748
- **Administer Active Workflows** (video[73]) allows Administrators to see every submission that is currently in the workflow approval process. From there, they have the option to delete Items (if they are no longer needed), or send them back to the workflow pool (to allow another user to review them). (Login as an Admin, Click "Administer Workflow" in sidebar)
- **CC License** step allows your users to select a Creative Commons License as part of their submission. Once enabled in the "item-submission.xml" (on the backend) it appears as part of the submission form.
- **Angular CLI** compatibility was added to the User Interface. This allows developers to easily update the User Interface using standard Angular commandline tools. More information (including tutorials) is available at https://cli.angular.io/
- English, Latvian, Dutch, German, French, Portuguese, Spanish and Finnish language catalogs
- Numerous bugs were fixed based on early user testing. (*Thanks to all who've tested Beta 1 or Beta 2 and reported your feedback!*) Some bugs fixed include:
    - Login/Logout session fixes (including compatibility with Firefox and Safari browsers)
    - Improved Community/Collection tree browsing performance
    - Fixes to editing Communities, Collections and Items. This includes improved drag & drop reordering of bitstreams in an Item.
    - Improved performance of Collection dropdown in submission
    - Ability to download restricted bitstreams (previously these would error out)
    - Authorization & security improvements in both REST API and UI
- Upgraded all REST API dependencies (Spring, Spring Boot, HAL Browser) and enhanced our automated testing via additional Integration Tests.
- *All features previous mentioned in* 7.0 Beta 2 Release Notes(see page 33) *and* 7.0 Beta 1 Release Notes(see page 33) *below*

Learn More: New videos are available highlighting features of the MyDSpace area:

- Manage Submissions in MyDSpace (video[74])
- Manage Tasks in MyDSpace (video[75])

Changelog

- All User Interface changes: https://github.com/DSpace/dspace-angular/issues?q=is%3Aclosed+milestone%3A7.0beta3
- All Backend changes: https://github.com/DSpace/DSpace/issues?q=is%3Aclosed+milestone%3A7.0beta3

73 https://www.youtube.com/watch?v=CjH8VS2WDjE
74 https://youtu.be/uM1q6W6k6lY
75 https://youtu.be/R0v1WNFDbmI

1.1.4.4 7.0 Beta 2 Release Notes

Released April 2020

Included in Beta 2

- **Administrative Search** (video[76]) combines retrieval of withdrawn items and private items, together with a series of quick action buttons.
- **EPeople, Groups and Roles** can now be viewed, created and updated.
  - Manage Groups (Login as an Admin → Access Control → Groups)
  - Manage EPeople (Login as an Admin → Access Control → EPeople)
  - Manage Community/Collection Roles (Login as an Admin → Edit Community/Collection → Assign Roles). Note: this feature is Admin-only in beta 2, but will be extended to Community/Collection Admins in beta 3.
- **Bitstream Editing** (video[77]) has a drag-and-drop interface for re-ordering bitstreams and makes adding and editing bitstreams more intuitive.
- **Metadata Editing** (video[78]) introduces suggest-as-you-type for field name selection of new metadata.
- **Update Profile / Change Password** (Login → Select user menu in upper right → Profile)
- Shibboleth Authentication[79]
- **Viewing Item Version History** (requires upgrading from a 6.x site that includes Item Versioning[80])
- Collection and Community (video[81]) creation and edit pages.
- English, Latvian, Dutch, German, French, Portuguese and Spanish language catalogs
- Security and authorization improvements, including REST API support hiding specific metadata fields (metadata.hide property) and upgrades of different software packages on which DSpace 7 depends.
- *All features previous mentioned in* 7.0 Beta 1 Release Notes(see page 33) *below*

A full list of all changes / bug fixes in 7.x is available in the Changes in 7.x(see page 697) section.

1.1.4.5 7.0 Beta 1 Release Notes

Released March 2020

New features to look for

- **A completely new User Interface (**demo site[82]**).** This is the new Javascript-based frontend, built on Angular.io[83] (with support for SEO provided by Angular Universal). This new interface is also via HTML and CSS (SCSS). For early theme building training, see the "Getting Started with DSpace 7 Workshop" from the North American User Group meeting: slides[84] or video recording[85].
- **A completely new, fully featured REST API (**demo site[86]**)**, provided via a single "server" webapp backend. This new backend is not only a REST API, but also still supports OAI-PMH, SWORD (v1 or v2) and RDF. See the REST API's documentation / contract at https://github.com/DSpace/Rest7Contract/blob/master/README.md

76 https://youtu.be/XoYStblYZWY
77 https://youtu.be/s1msEKK0f68
78 https://youtu.be/6KVB2ugUgjI
79 https://wiki.lyrasis.org/display/DSPACE/DSpace+7+Shibboleth+Configuration
80 https://wiki.lyrasis.org/display/DSDOC6x/Item+Level+Versioning
81 https://youtu.be/wnrUOHRS5WA
82 https://dspace7-demo.atmire.com/
83 http://Angular.io
84 https://tinyurl.com/na-dsug2019-dspace7
85 https://umn.zoom.us/recording/play/Mk3gWE1AGEErGg6fLaZ_u-rg_8pEC7MWu_2uWa5l8Q03fKM7ra9sm1-MntSRtNti?startTime=1569247043000
86 https://dspace7.4science.cloud/server/

- **A newly designed search box.** Search from the header of any page (click the magnifying glass). The search results page now features automatic search highlight, expandable & searchable filters, and optional thumbnail-based results (click on the "grid" view).
- **A new MyDSpace area**, including a new, one-page, drag & drop submission form, a new workflow approval process, and searchable past submissions. (Login, click on your user profile icon, click "MyDSpace"). Find workflow tasks to claim by selecting "All tasks" in the "Show" dropdown.
- **Dynamic user interface translations** (Click the globe, and select a language). Anyone interested in adding more translations? See DSpace 7 Translation - Internationalization (i18n) - Localization (l10n)[87].
- **A new Admin sidebar.** Login as an Administrator, and an administrative sidebar appears. Use this to create a new Community/Collection/Item, edit existing ones, and manage registries. (*NOTE: A number of Administrative tools are still missing or greyed out. They will be coming in future Beta releases.*)
- Optional, new Configurable Entities(see page 134) **feature**. DSpace now supports "entities", which are DSpace Items of a specific 'type' which may have relationships to other entities. These entity types and relationships are configurable, with two examples coming out-of-the-box: a set of Journal hierarchy entities (Journal, Volume, Issue, Publication) and a set of Research entities (Publication, Project, Person, OrgUnit). For more information see "The Power of Configurable Entities" from OR2019: slides[88] or video recording[89]. Additionally, a test data set featuring both out-of-the-box examples can be used when trying out DSpace 7 via Docker[90]. Early documentation is available at Configurable Entities(see page 134).
- **Support for OpenAIREv4** Guidelines for Literature Repositories[91] in OAI-PMH (See the new "openaire4" context in OAI-PMH).

Additional major changes to be aware of in the 7.x platform (not an exhaustive list):

- **XMLUI and JSPUI are no longer supported or distributed with DSpace.** All users should immediately migrate to and utilize the new Angular User Interface[92]. There is no migration path from either the XMLUI or JSPUI to the new User interface. However, the new user interface can be themed via HTML and CSS (SCSS).
- **The old REST API ("rest" webapp from DSpace v4.x-6.x) is deprecated and will be removed in v8.x.** The new REST API (provided in the "server" webapp) replaces all functionality available in the older REST API. If you have tools that rely on the old REST API, you can still (optionally) build & deploy it alongside the "server" webapp via the "-Pdspace-rest" Maven flag.
- **The Submission Form configuration has changed.** The "item-submission.xml" file has changed its structure, and the "input-forms.xml" has been replaced by a "submission-forms.xml". For early documentation see Configuration changes in the submission process[93]
- **ElasticSearch Usage Statistics have been removed.** Please use SOLR Statistics(see page 338) or DSpace Google Analytics Statistics(see page 363).
- **The traditional, 3-step Workflow system has been removed in favor of the** Configurable Workflow System(see page 250)**.** For most users, you should see no effect or difference. The default setup for this Configurable Workflow System is identical to the traditional, 3-step workflow ("Approve/Reject", "Approve/Reject/Edit Metadata", "Edit Metadata")
- **Apache Solr is no longer embedded within the DSpace installer (and has been upgraded to Solr v7).** Solr now MUST be installed as a separate dependency alongside the DSpace backend. See Installing DSpace(see page 53).
- **Some command-line tools/scripts are enabled in the new REST API** (e.g. index-discovery): See new Scripts endpoint: https://github.com/DSpace/Rest7Contract/blob/master/scripts-endpoint.md
- **DSpace now has a single, backend "server" webapp to deploy in Tomcat (or similar).** In DSpace 6.x and below, different machine interfaces (OAI-PMH, SWORD v1 or v2, RDF, REST API) were provided via separate deployable webapps. Now, all those interfaces along with the new REST API are in a single, "server" webapp

87 https://wiki.lyrasis.org/pages/viewpage.action?pageId=117735441
88 https://www.slideshare.net/Atmire/dspace-7-the-power-of-configurable-entities
89 https://lecture2go.uni-hamburg.de/l2go/-/get/v/24831
90 https://wiki.lyrasis.org/display/DSPACE/Try+out+DSpace+7#TryoutDSpace7-InstallviaDocker
91 https://guidelines.openaire.eu/en/latest/literature/index.html
92 https://github.com/DSpace/dspace-angular/
93 https://wiki.lyrasis.org/display/DSPACE/Configuration+changes+in+the+submission+process

built on Spring Boot[94]. You can now control which interfaces are enabled, and what path they appear on via configuration (e.g. "oai.enabled=true" and "oai.path=oai"). See https://jira.lyrasis.org/browse/DS-4257
- Configuration(see page 552) **has been upgraded to Apache Commons Configuration version 2.** For most users, you should see no effect or difference. *No DSpace configuration files were modified during this upgrade and no configurations or settings were renamed or changed.* However, if you locally modified or customized the `[dspace]/config/config-definition.xml` (DSpace's Apache Commons Configuration settings), you will need to ensure those modifications are compatible with Apache Commons Configuration version 2. See the Apache Commons Configuration's configuration definition file reference[95] for more details.
- **Handle Server has been upgraded to version 9.x :** https://jira.lyrasis.org/browse/DS-4205
- **DSpace now has sample Docker images (configurations)** which can be used to try out DSpace quickly. See Try out DSpace 7[96] ("Install via Docker" section).

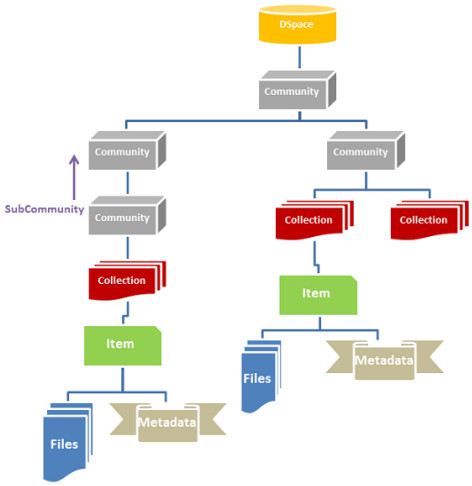
1.2 Functional Overview

The following sections describe the various functional aspects of the DSpace system.

94 https://spring.io/projects/spring-boot
95 https://commons.apache.org/proper/commons-configuration/userguide/howto_combinedbuilder.html#Configuration_definition_file_reference
96 https://wiki.lyrasis.org/display/DSPACE/Try+out+DSpace+7

1.2.1 Online access to your digital assets

The online presentation of your content in an organized tree of Communities and Collections is a main feature of DSpace. Users can access pages for individual items, these are metadata descriptions together with files available for download. The structure is summarised in this diagram (click to see the image at full size).



1.2.1.1 Full-text search

DSpace can process uploaded text based contents for full-text searching. This means that not only the metadata you provide for a given file will be searchable, but all of its contents will be indexed as well. This allows users to search for specific keywords that only appear in the actual content and not in the provided description.

1.2.1.2 Navigation

DSpace allows users to find their way to relevant content in a number of ways, including:

- **Searching** for one or more keywords in metadata or extracted full-text
- **Faceted browsing** through any field provided in the item description.
- Through **external reference**, such as a Handle

- By clicking on Community and Collection titles to explore their contents

Another important mechanism for discovery in DSpace is the browse. This is the process whereby the user views a particular index, such as the title index, and navigates around it in search of interesting items. The browse subsystem provides a simple API for achieving this by allowing a caller to specify an index, and a subsection of that index. The browse subsystem then discloses the portion of the index of interest. Indices that may be browsed are item title, item issue date, item author, and subject terms. Additionally, the browse can be limited to items within a particular collection or community.

For more information on Search/Browse functionality in DSpace, see Discovery(see page 386).

1.2.1.3 Supported file types

DSpace can accommodate any type of uploaded file. While DSpace is most known for hosting text based materials including scholarly communication and electronic theses and dissertations (ETDs), there are many stakeholders in the community who use DSpace for multimedia, data and learning objects. While some restrictions apply, DSpace can even serve as a store for HTML Archives(see page 201).

Files that have been uploaded to DSpace are often referred to as "Bitstreams". The reason for this is mainly historic and tracks back to the technical implementation. After ingestion, files in DSpace are stored on the file system as a stream of bits without the file extension.

By default, DSpace only recognizes specific file types, as defined in its Bitstream Format Registry. The default Bitstream Format Registry(see page 640) recognizes many common file formats, but it can be enhanced at your local institution via the Admin User Interface.

1.2.1.4 Optimized for Google Indexing

The Duraspace community fosters a close relation with Google to ensure optimal indexing of DSpace content, primarily in the Google Search and Google Scholar products. For the purpose of Google Scholar indexing, DSpace added specific metadata in the page head tags facilitating indexing in Scholar. More information can be retrieved on the Google Scholar Metadata Mappings page(see page 492). Popular DSpace repositories often generate over 60% of their visits from Google pages.

1.2.1.5 OpenURL Support

DSpace supports the OpenURL protocol[97] in a rather simple fashion. If your institution has an SFX server[98], DSpace will display an OpenURL link on every item page, automatically using the Dublin Core metadata. Additionally, DSpace can respond to incoming OpenURLs. Presently it simply passes the information in the OpenURL to the search subsystem. A list of results is then displayed, which usually gives the relevant item (if it is in DSpace) at the top of the list.

1.2.1.6 Support for modern browsers

The DSpace developer community aims to rely on modern web standards and well tested libraries where possible. As a rule of thumb, users can expect that the DSpace web interfaces work on modern web browsers. DSpace developers routinely test new interface developments on recent versions of Firefox, Safari, Chrome and Microsoft Edge. Because of fast moving, automatic, incremental updates to these browsers, support is no longer targeted at specific versions of these browsers. (Please note that we do not recommend or support using Internet Explorer as it is considered "end of life" by Microsoft.)

---

97 https://en.wikipedia.org/wiki/OpenURL
98 http://www.exlibrisgroup.com/category/SFXOverview

1.2.2 Metadata Management

1.2.2.1 Metadata

Broadly speaking, DSpace holds three sorts of metadata about archived content:

- **Descriptive Metadata**: DSpace can support multiple flat metadata schemas for describing an item. A qualified Dublin Core metadata schema loosely based on the Library Application Profile[99] set of elements and qualifiers is provided by default. This default schema is described in more detail in Metadata and Bitstream Format Registries(see page 640). However, you can configure multiple schemas and select metadata fields from a mix of configured schemas to describe your items. Other descriptive metadata about items (e.g. metadata described in a hierarchical schema) may be held in serialized bitstreams.
- **Administrative Metadata**: This includes preservation metadata, provenance and authorization policy data. Most of this is held within DSpace's relational DBMS schema. Provenance metadata (prose) is stored in Dublin Core records. Additionally, some other administrative metadata (for example, bitstream byte sizes and MIME types) is replicated in Dublin Core records so that it is easily accessible outside of DSpace.
- **Structural Metadata**: This includes information about how to present an item, or bitstreams within an item, to an end-user, and the relationships between constituent parts of the item. As an example, consider a thesis consisting of a number of TIFF images, each depicting a single page of the thesis. Structural metadata would include the fact that each image is a single page, and the ordering of the TIFF images/pages. Structural metadata in DSpace is currently fairly basic; within an item, bitstreams can be arranged into separate bundles as described above. A bundle may also optionally have a *primary bitstream*. This is currently used by the HTML support to indicate which bitstream in the bundle is the first HTML file to send to a browser. In addition to some basic technical metadata, a bitstream also has a 'sequence ID' that uniquely identifies it within an item. This is used to produce a 'persistent' bitstream identifier for each bitstream. Additional structural metadata can be stored in serialized bitstreams, but DSpace does not currently understand this natively.

1.2.2.2 Choice Management and Authority Control

This is a configurable framework that lets you define plug-in classes to control the choice of values for specified DSpace metadata fields. It also lets you configure fields to include "authority" values along with the textual metadata value. The choice-control system includes a user interface in both the Configurable Submission UI and the Admin UI (edit Item pages) that assists the user in choosing metadata values.

Introduction and Motivation

Definitions

Choice Management

This is a mechanism that generates a list of choices for a value to be entered in a given metadata field. Depending on your implementation, the exact choice list might be determined by a proposed value or query, or it could be a fixed list that is the same for every query. It may also be closed (limited to choices produced internally) or open, allowing the user-supplied query to be included as a choice.

Authority Control

This works in addition to choice management to supply an authority key along with the chosen value, which is also assigned to the Item's metadata field entry. Any authority-controlled field is also inherently choice-controlled.

About Authority Control

99 http://www.dublincore.org/documents/library-application-profile/

The advantages we seek from an authority controlled metadata field are:

1. **There is a simple and positive way to test whether two values are identical**, by comparing authority keys.
    - Comparing plain text values can give false positive results e.g. when two different people have a name that is written the same.
    - It can also give false negative results when the same name is written different ways, e.g. "J. Smith" vs. "John Smith".
2. **Help in entering correct metadata values.** The submission and admin UIs may call on the authority to check a proposed value and list possible matches to help the user select one.
3. **Improved interoperability.** By sharing a name authority with another application, your DSpace can interoperate more cleanly with other applications.
    - For example, a DSpace institutional repository sharing a naming authority with the campus social network would let the social network construct a list of all DSpace Items matching the shared author identifier, rather than by error-prone name matching.
    - When the name authority is shared with a campus directory, DSpace can look up the email address of an author to send automatic email about works of theirs submitted by a third party. That author does not have to be an EPerson.
4. Authority keys are normally invisible in the public web UIs. They are only seen by administrators editing metadata. The value of an authority key is not expected to be meaningful to an end-user or site visitor.

Authority control is different from the controlled vocabulary of keywords already implemented in the submission UI:

1. **Authorities are external to DSpace.** The source of authority control is typically an external database or network resource.
    - Plug-in architecture makes it easy to integrate new authorities without modifying any core code.
2. This authority proposal impacts all phases of metadata management.
    - The keyword vocabularies are only for the submission UI.
    - Authority control is asserted everywhere metadata values are changed, including unattended/batch submission, SWORD package submission, and the administrative UI.

Some Terminology

Authority	An authority is a source of fixed values for a given domain, each unique value identified by a key.
.	For example, the OCLC LC Name Authority Service.
Authority Record	The information associated with one of the values in an authority; may include alternate spellings and equivalent forms of the value, etc.
Authority Key	An opaque, hopefully persistent, identifier corresponding to exactly one record in the authority.

1.2.3 Licensing

DSpace offers support for licenses on different levels

1.2.3.1 Collection and Community Licenses

Each community and collection in the hierarchy of a DSpace repository can contain its own license terms. This allows an institution to use the repository both for collections where certain rights are reserved and others from which the content may be accessed and distributed more freely.

1.2.3.2 License granted by the submitter to the repository

At the end of the manual submission process, the submitter is asked to grant the repository service an appropriate distribution license. This license can be easily customized on a per collection basis. In its most common form, the submitter grants to the repository service a non-exclusive distribution license, meaning that he officially gives the repository service the right to share his or her work with the world.

1.2.3.3 Creative Commons Support for DSpace Items

DSpace provides support for Creative Commons licenses to be attached to items in the repository. They represent an alternative to traditional copyright. To learn more about Creative Commons, visit their website[100]. Support for license selection is controlled by a site-wide configuration option, and since license selection involves interaction with the Creative Commons website, additional parameters may be configured to work with a proxy server. If the option is enabled, users may select a Creative Commons license during the submission process, or select to don't assign a Creative Commons license at all. If a selection is made, metadata and a copy of the license in the RDF format is stored along with the item in the repository. There is also an indication - text and a Creative Commons icon - in the item display page of the web user interface when an item is licensed under Creative Commons. The RDF license is embedded in the html page of the item to allow machine understanding of the licensing terms. For specifics of how to configure and use Creative Commons licenses, see the configuration section(see page 601).

1.2.4 Persistent URLs and Identifiers

1.2.4.1 Handles

Researchers require a stable point of reference for their works. The simple evolution from sharing of citations to emailing of URLs broke when Web users learned that sites can disappear or be reconfigured without notice, and that their bookmark files containing critical links to research results couldn't be trusted in the long term. To help solve this problem, a core DSpace feature is the creation of a persistent identifier for every item, collection and community stored in DSpace. To persist identifiers, DSpace requires a storage- and location- independent mechanism for creating and maintaining identifiers. DSpace uses the CNRI Handle System[101] for creating these identifiers. The rest of this section assumes a basic familiarity with the Handle system.

DSpace uses Handles primarily as a means of assigning globally unique identifiers to objects. Each site running DSpace needs to obtain a unique Handle 'prefix' from CNRI, so we know that if we create identifiers with that prefix, they won't clash with identifiers created elsewhere.

Presently, Handles are assigned to communities, collections, and items. Bundles and bitstreams are not assigned Handles, since over time, the way in which an item is encoded as bits may change, in order to allow access with future technologies and devices. Older versions may be moved to off-line storage as a new standard becomes de facto. Since it's usually the *item* that is being preserved, rather than the particular bit encoding, it only makes sense

100 http://creativecommons.org/
101 http://www.handle.net/

to persistently identify and allow access to the item, and allow users to access the appropriate bit encoding from there.

Of course, it may be that a particular bit encoding of a file is explicitly being preserved; in this case, the bitstream could be the only one in the item, and the item's Handle would then essentially refer just to that bitstream. The same bitstream can also be included in other items, and thus would be citable as part of a greater item, or individually.

The Handle system also features a global resolution infrastructure; that is, an end-user can enter a Handle into any service (e.g. Web page) that can resolve Handles, and the end-user will be directed to the object (in the case of DSpace, community, collection or item) identified by that Handle. In order to take advantage of this feature of the Handle system, a DSpace site must also run a 'Handle server' that can accept and resolve incoming resolution requests. All the code for this is included in the DSpace source code bundle.

Handles can be written in two forms:

```
hdl:1721.123/4567
http://hdl.handle.net/1721.123/4567
```

The above represent the same Handle. The first is possibly more convenient to use only as an identifier; however, by using the second form, any Web browser becomes capable of resolving Handles. An end-user need only access this form of the Handle as they would any other URL. It is possible to enable some browsers to resolve the first form of Handle as if they were standard URLs using CNRI's Handle Resolver plug-in[102], but since the first form can always be simply derived from the second, DSpace displays Handles in the second form, so that it is more useful for end-users.

It is important to note that DSpace uses the CNRI Handle infrastructure only at the 'site' level. For example, in the above example, the DSpace site has been assigned the prefix '1721.123'. It is still the responsibility of the DSpace site to maintain the association between a full Handle (including the '4567' local part) and the community, collection or item in question.

1.2.4.2 Bitstream 'Persistent' Identifiers

Similar to handles for DSpace items, bitstreams also have 'Persistent' identifiers. They are more volatile than Handles, since if the content is moved to a different server or organization, they will no longer work (hence the quotes around 'persistent'). However, they are more easily persisted than the simple URLs based on database primary key previously used. This means that external systems can more reliably refer to specific bitstreams stored in a DSpace instance.

Each bitstream has a sequence ID, unique within an item. This sequence ID is used to create a persistent ID, of the form:

dspace url/bitstream/handle/sequence ID/filename

For example:

```
https://dspace.myu.edu/bitstream/123.456/789/24/foo.html
```

The above refers to the bitstream with sequence ID 24 in the item with the Handle hdl:123.456/789[103].
The *foo.html* is really just there as a hint to browsers: Although DSpace will provide the appropriate MIME type, some browsers only function correctly if the file has an expected extension.

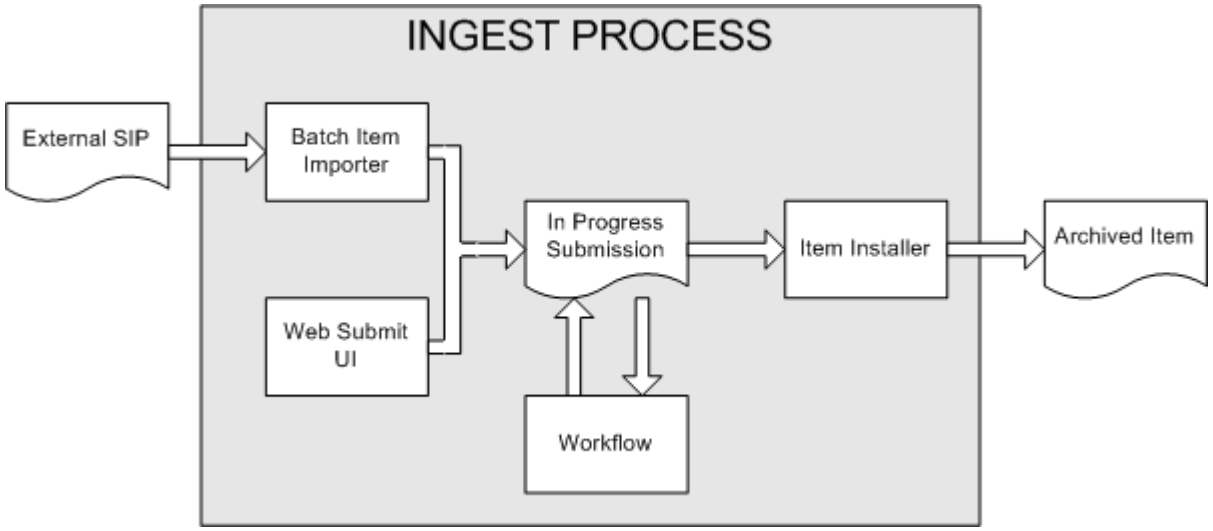
---

102 http://www.handle.net/resolver/index.html
103 http://hdl:123.456

1.2.5 Getting content into DSpace

1.2.5.1 The Manual DSpace Submission and Workflow System

Rather than being a single subsystem, ingesting is a process that spans several. Below is a simple illustration of the current ingesting process in DSpace.



DSpace Ingest Process

The batch item importer is an application, which turns an external SIP (an XML metadata document with some content files) into an "in progress submission" object. The Web submission UI is similarly used by an end-user to assemble an "in progress submission" object.

Depending on the policy of the collection to which the submission in targeted, a workflow process may be started. This typically allows one or more human reviewers or 'gatekeepers' to check over the submission and ensure it is suitable for inclusion in the collection.

When the Batch Ingester or Submission UI completes the InProgressSubmission object, and invokes the next stage of ingest (be that workflow or item installation), a provenance message is added to the Dublin Core which includes the filenames and checksums of the content of the submission. Likewise, each time a workflow changes state (e.g. a reviewer accepts the submission), a similar provenance statement is added. This allows us to track how the item has changed since a user submitted it.

Once any workflow process is successfully and positively completed, the InProgressSubmission object is consumed by an "item installer", that converts the InProgressSubmission into a fully blown archived item in DSpace. The item installer:

- Assigns an accession date
- Adds a "date.available" value to the Dublin Core metadata record of the item
- Adds an issue date if none already present
- Adds a provenance message (including bitstream checksums)
- Assigns a Handle persistent identifier
- Adds the item to the target collection, and adds appropriate authorization policies
- Adds the new item to the search and browse index

Workflow Steps

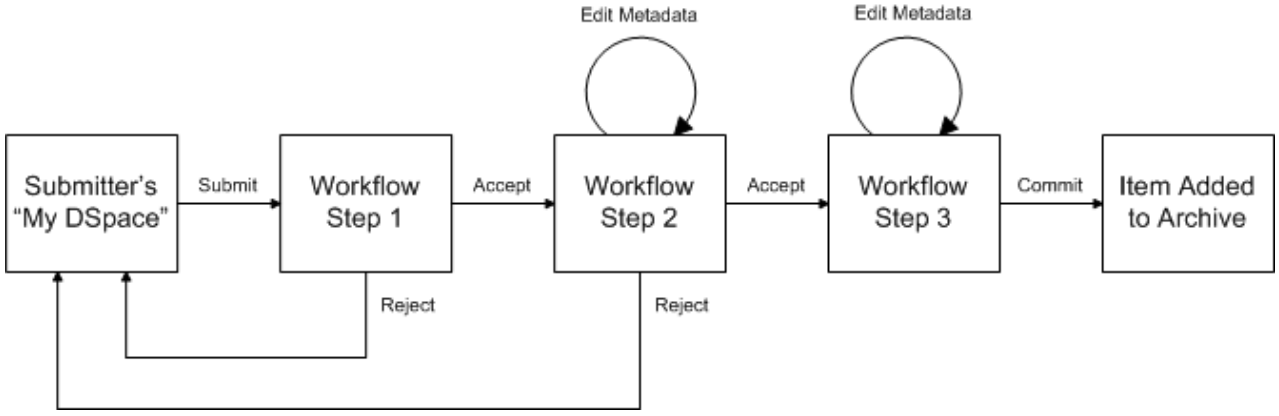
By default, a collection's workflow may have up to three steps. Each collection may have an associated e-person group for performing each step; if no group is associated with a certain step, that step is skipped. If a collection has no e-person groups associated with any step, submissions to that collection are installed straight into the main archive. Keep in mind, however, that this is only the **default** behavior, and the workflow process can be configured/customized easily, see Configurable Workflow(see page 250).

In other words, the default sequence is this: The collection receives a submission. If the collection has a group assigned for workflow step 1, that step is invoked, and the group is notified. Otherwise, workflow step 1 is skipped. Likewise, workflow steps 2 and 3 are performed if and only if the collection has a group assigned to those steps.

When a step is invoked, the submission is put into the 'task pool' of the step's associated group. One member of that group takes the task from the pool, and it is then removed from the task pool, to avoid the situation where several people in the group may be performing the same task without realizing it.

The member of the group who has taken the task from the pool may then perform one of three actions:

Workflow Step	Possible actions
1	Can accept submission for inclusion, or reject submission.
2	Can edit metadata provided by the user with the submission, but cannot change the submitted files. Can accept submission for inclusion, or reject submission.
3	Can edit metadata provided by the user with the submission, but cannot change the submitted files. Must then commit to archive; may not reject submission.



Submission Workflow in DSpace

If a submission is rejected, the reason (entered by the workflow participant) is e-mailed to the submitter, and it is returned to the submitter's 'My DSpace' page. The submitter can then make any necessary modifications and re-submit, whereupon the process starts again.

If a submission is 'accepted', it is passed to the next step in the workflow. If there are no more workflow steps with associated groups, the submission is installed in the main archive.

One last possibility is that a workflow can be 'aborted' by a DSpace site administrator. This is accomplished using the Administration UI.

1.2.5.2 Command line import facilities

DSpace includes batch tools to import items in a simple directory structure, where the Dublin Core metadata is stored in an XML file. This may be used as the basis for moving content between DSpace and other systems. For more information see Item Importer and Exporter(see page 0).

DSpace also includes various package importer tools, which support many common content packaging formats like METS. For more information see Package Importer and Exporter(see page 0). Additionally, DSpace can import/export Archival Information Packages (AIPs), see AIP Backup and Restore(see page 411).

1.2.5.3 Registration for externally hosted files

Registration is an alternate means of incorporating items, their metadata, and their bitstreams into DSpace by taking advantage of the bitstreams already being in accessible computer storage. An example might be that there is a repository for existing digital assets. Rather than using the normal interactive ingest process or the batch import to furnish DSpace the metadata and to upload bitstreams, registration provides DSpace the metadata and the location of the bitstreams. DSpace uses a variation of the import tool to accomplish registration.

1.2.5.4 SWORD Support

SWORD (Simple Web-service Offering Repository Deposit) is a protocol that allows the remote deposit of items into repositories. SWORD was further developed in SWORD version 2 to add the ability to retrieve, update, or delete deposits. DSpace supports the SWORD protocol via the 'sword' web application and SWord v2 via the swordv2 web application. The specification and further information can be found at http://swordapp.org[104]. See also SWORDv1 Server(see page 216) and SWORDv2 Server(see page 202).

1.2.6 Getting content out of DSpace

1.2.6.1 OAI Support

The Open Archives Initiative[105] has developed a protocol for metadata harvesting[106]. This allows sites to programmatically retrieve or 'harvest' the metadata from several sources, and offer services using that metadata, such as indexing or linking services. Such a service could allow users to access information from a large number of sites from one place.

DSpace exposes the Dublin Core metadata for items that are publicly (anonymously) accessible. Additionally, the collection structure is also exposed via the OAI protocol's 'sets' mechanism. OCLC's open source OAICat[107] framework is used to provide this functionality.

104 http://swordapp.org/
105 http://www.openarchives.org/
106 http://www.openarchives.org/OAI/openarchivesprotocol.html
107 http://www.oclc.org/research/software/oai/cat.shtm

You can also configure the OAI service to make use of any crosswalk plugin to offer additional metadata formats, such as MODS.

DSpace's OAI service does support the exposing of deletion information for withdrawn items, but not for items that are 'expunged' (see above). DSpace also supports OAI-PMH resumption tokens. See OAI(see page 179) for more information.

1.2.6.2 Command Line Export Facilities

DSpace includes batch tools to export items in a simple directory structure, where the Dublin Core metadata is stored in an XML file. This may be used as the basis for moving content between DSpace and other systems. For more information see Item Importer and Exporter(see page 0).

DSpace also includes various package exporter tools, which support many common content packaging formats like METS. For more information see Package Importer and Exporter(see page 0). Additionally, DSpace can import/export Archival Information Packages (AIPs), see AIP Backup and Restore(see page 411).

1.2.6.3 Packager Plugins

Packagers are software modules that translate between DSpace Item objects and a self-contained external representation, or "package". A *Package Ingester* interprets, or *ingests*, the package and creates an Item. A *Package Disseminator* writes out the contents of an Item in the package format.

A package is typically an archive file such as a Zip or "tar" file, including a *manifest* document which contains metadata and a description of the package contents. The IMS Content Package[108] is a typical packaging standard. A package might also be a single document or media file that contains its own metadata, such as a PDF document with embedded descriptive metadata.

Package ingesters and package disseminators are each a type of named plugin (see Plugin Manager(see page 0)), so it is easy to add new packagers specific to the needs of your site. You do not have to supply both an ingester and disseminator for each format; it is perfectly acceptable to just implement one of them.

Most packager plugins call upon Crosswalk Plugins(see page 45) to translate the metadata between DSpace's object model and the package format.

More information about calling Packagers to ingest or disseminate content can be found in the Package Importer and Exporter(see page 0) section of the System Administration documentation.

1.2.6.4 Crosswalk Plugins

Crosswalks are software modules that translate between DSpace object metadata and a specific external representation. An *Ingestion Crosswalk* interprets the external format and crosswalks it to DSpace's internal data structure, while a *Dissemination Crosswalk* does the opposite.

For example, a MODS ingestion crosswalk translates descriptive metadata from the MODS format to the metadata fields on a DSpace Item. A MODS dissemination crosswalk generates a MODS document from the metadata on a DSpace Item.

Crosswalk plugins are named plugins (see Plugin Manager(see page 0)), so it is easy to add new crosswalks. You do not have to supply both an ingester and disseminator for each format; it is perfectly acceptable to just implement one of them.

---

108 http://www.imsglobal.org/content/packaging/

There is also a special pair of crosswalk plugins which use XSL stylesheets to translate the external metadata to or from an internal DSpace format. You can add and modify XSLT crosswalks simply by editing the DSpace configuration and the stylesheets, which are stored in files in the DSpace installation directory.

The Packager plugins and OAH-PMH server make use of crosswalk plugins.

1.2.6.5 Supervision and Collaboration

In order to facilitate, as a primary objective, the opportunity for thesis authors to be supervised in the preparation of their e-theses, a supervision order system exists to bind groups of other users (thesis supervisors) to an item in someone's pre-submission workspace. The bound group can have system policies associated with it that allow different levels of interaction with the student's item; a small set of default policy groups are provided:

- Full editorial control
- View item contents
- No policies
  Once the default set has been applied, a system administrator may modify them as they would any other policy set in DSpace

This functionality could also be used in situations where researchers wish to collaborate on a particular submission, although there is no particular collaborative workspace functionality.

1.2.7 User Management

Although many of DSpace's functions such as document discovery and retrieval can be used anonymously, some features (and perhaps some documents) are only available to certain "privileged" users. E-People and Groups are the way DSpace identifies application users for the purpose of granting privileges. This identity is bound to a session of a DSpace application such as the Web UI or one of the command-line batch programs. Both E-People and Groups are granted privileges by the authorization system described below.

1.2.7.1 User Accounts (E-Person)

DSpace holds the following information about each e-person:

- E-mail address
- First and last names
- Whether the user is able to log in to the system via the Web UI, and whether they must use an X509 certificate to do so;
- A password (encrypted), if appropriate
- A list of collections for which the e-person wishes to be notified of new items
- Whether the e-person 'self-registered' with the system; that is, whether the system created the e-person record automatically as a result of the end-user independently registering with the system, as opposed to the e-person record being generated from the institution's personnel database, for example.
- The network ID for the corresponding LDAP record, if LDAP authentication is used for this E-Person.

1.2.7.2 Subscriptions

As noted above, end-users (e-people) may 'subscribe' to collections in order to be alerted when new items appear in those collections. Each day, end-users who are subscribed to one or more collections will receive an e-mail giving brief details of all new items that appeared in any of those collections the previous day. If no new items appeared in any of the subscribed collections, no e-mail is sent. Users can unsubscribe themselves at any time. RSS feeds of new items are also available for collections and communities.

1.2.7.3 Groups

Groups are another kind of entity that can be granted permissions in the authorization system. A group is usually an explicit list of E-People; anyone identified as one of those E-People also gains the privileges granted to the group.

However, an application session can be assigned membership in a group *without* being identified as an E-Person. For example, some sites use this feature to identify users of a local network so they can read restricted materials not open to the whole world. Sessions originating from the local network are given membership in the "LocalUsers" group and gain the corresponding privileges.

Administrators can also use groups as "roles" to manage the granting of privileges more efficiently.

1.2.8 Access Control

1.2.8.1 Authentication

Authentication is when an application session positively identifies itself as belonging to an E-Person and/or Group. In DSpace, it is implemented by a mechanism called *Stackable Authentication*: the DSpace configuration declares a "stack" of authentication methods. An application (like the Web UI) calls on the Authentication Manager, which tries each of these methods in turn to identify the E-Person to which the session belongs, as well as any extra Groups. The E-Person authentication methods are tried in turn until one succeeds. Every authenticator in the stack is given a chance to assign extra Groups. This mechanism offers the following advantages:

- Separates authentication from the Web user interface so the same authentication methods are used for other applications such as non-interactive Web Services
- Improved modularity: The authentication methods are all independent of each other. Custom authentication methods can be "stacked" on top of the default DSpace username/password method.
- Cleaner support for "implicit" authentication where username is found in the environment of a Web request, e.g. in an X.509 client certificate.

For more information see Authentication Plugins(see page 87)

1.2.8.2 Authorization

DSpace's authorization system is based on associating actions with objects and the lists of EPeople who can perform them. The associations are called Resource Policies, and the lists of EPeople are called Groups. There are two built-in groups: 'Administrators', who can do anything in a site, and 'Anonymous', which is a list that contains all users. Assigning a policy for an action on an object to anonymous means giving everyone permission to do that action. (For example, most objects in DSpace sites have a policy of 'anonymous' READ.) Permissions must be explicit - lack of an explicit permission results in the default policy of 'deny'. Permissions also do not 'commute'; for example, if an e-person has READ permission on an item, they might not necessarily have READ permission on the bundles and bitstreams in that item. Currently Collections, Communities and Items are discoverable in the browse and search systems regardless of READ authorization.

The following actions are possible:

Collection

ADD/REMOVE	add or remove items (ADD = permission to submit items)

DEFAULT_ITEM_READ	inherited as READ by all submitted items
DEFAULT_BITSTREAM_READ	inherited as READ by Bitstreams of all submitted items. Note: only affects Bitstreams of an item at the time it is initially submitted. If a Bitstream is added later, it does *not* get the same default read policy.
COLLECTION_ADMIN	collection admins can edit items in a collection, withdraw items, map other items into this collection.

Item

ADD/REMOVE	add or remove bundles
READ	can view item (item metadata is always viewable)
WRITE	can modify item

Bundle

ADD/REMOVE	add or remove bitstreams to a bundle

Bitstream

READ	view bitstream
WRITE	modify bitstream

Note that there is no 'DELETE' action. In order to 'delete' an object (e.g. an item) from the archive, one must have REMOVE permission on all objects (in this case, collection) that contain it. The 'orphaned' item is automatically deleted.

Policies can apply to individual e-people or groups of e-people.

1.2.9 Usage Metrics

DSpace is equipped with SOLR based infrastructure to log and display pageviews and file downloads.

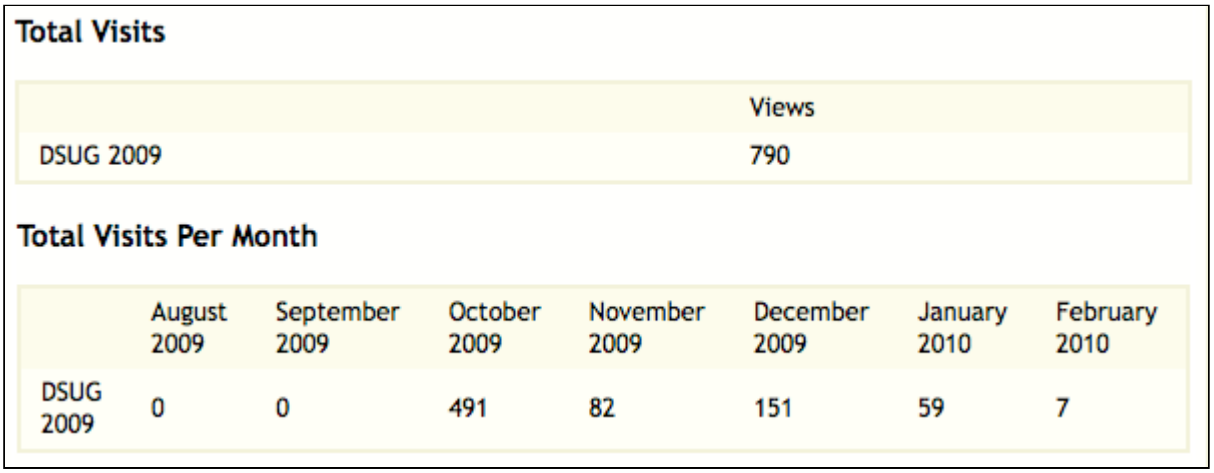
1.2.9.1 Item, Collection and Community Usage Statistics

Usage statistics can be retrieved from individual item, collection and community pages. These Usage Statistics pages show:

- Total page visits (all time)
- Total Visits per Month

- File Downloads (all time)*
- Top Country Views (all time)
- Top City Views (all time)

*File Downloads information is only displayed for item-level statistics. Note that downloads from separate bitstreams are also recorded and represented separately. DSpace is able to capture and store File Download information, even when the bitstream was downloaded from a direct link on an external website.



Total Visits

	Views
DSUG 2009	790

Total Visits Per Month

	August 2009	September 2009	October 2009	November 2009	December 2009	January 2010	February 2010
DSUG 2009	0	0	491	82	151	59	7

1.2.9.2 System Statistics

Various statistical reports about the contents and use of your system can be automatically generated by the system. These are generated by analyzing DSpace's log files. Statistics can be broken down monthly.

The report includes following sections

- A customizable general overview of activities in the archive, by default including:
  - Number of items archived
  - Number of bitstream views
  - Number of item page views
  - Number of collection page views
  - Number of community page views
  - Number of user logins
  - Number of searches performed
  - Number of license rejections
  - Number of OAI Requests
- Customizable summary of archive contents
- Broken-down list of item viewings
- A full break-down of all performed actions
- User logins
- Most popular searches
- Log Level Information
- Processing information!stats_genrl_overview.png!
  The results of statistical analysis can be presented on a by-month and an in-total report, and are available via the user interface. The reports can also either be made public or restricted to administrator access only.

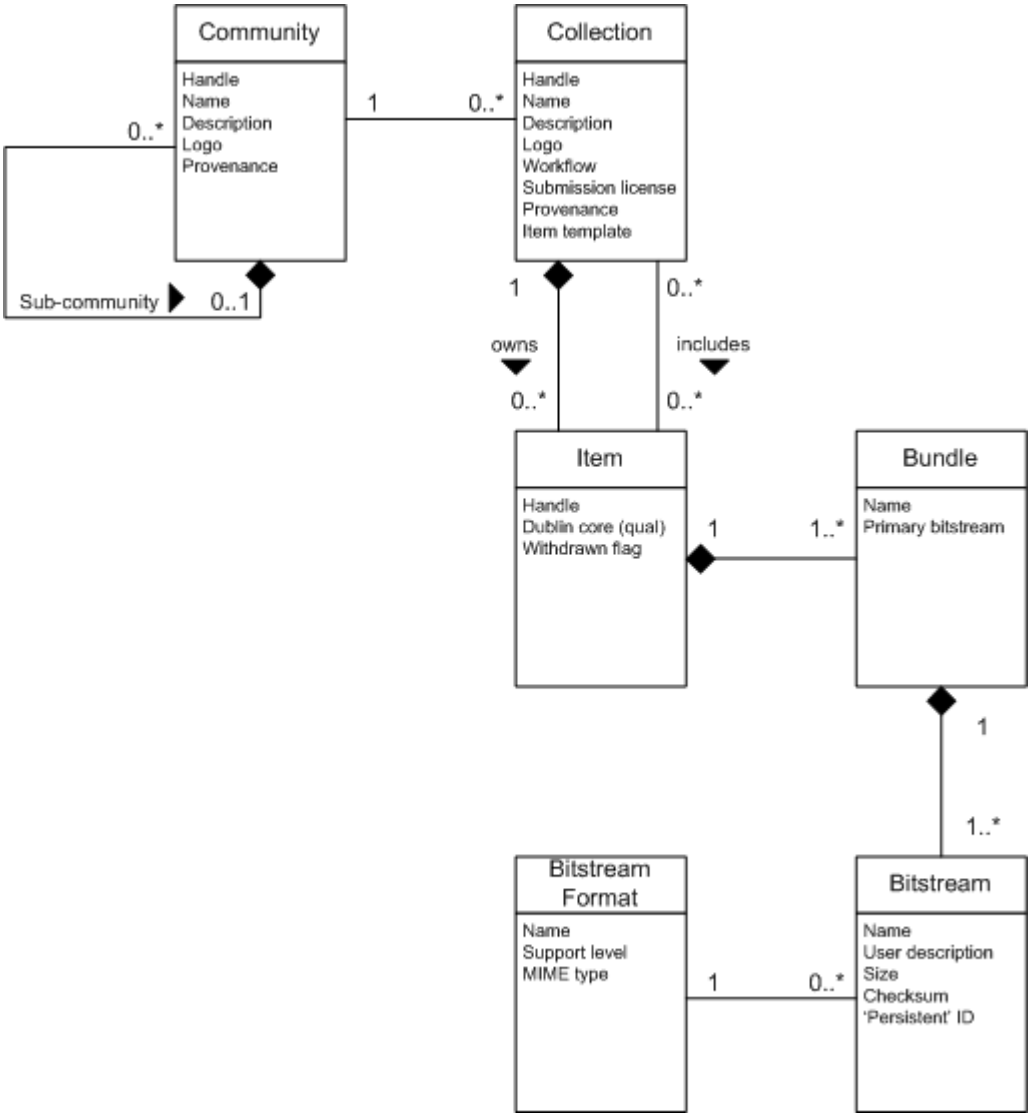
1.2.10 Digital Preservation

1.2.10.1 Checksum Checker

The purpose of the checker is to verify that the content in a DSpace repository has not become corrupted or been tampered with. The functionality can be invoked on an ad-hoc basis from the command line, or configured via cron or similar. Options exist to support large repositories that cannot be entirely checked in one run of the tool. The tool is extensible to new reporting and checking priority approaches.

1.2.11 System Design

1.2.11.1 **Data Model**



Data Model Diagram

The way data is organized in DSpace is intended to reflect the structure of the organization using the DSpace system. Each DSpace site is divided into *communities*, which can be further divided into *sub-communities* reflecting the typical university structure of college, department, research center, or laboratory.

Communities contain *collections*, which are groupings of related content. A collection may appear in more than one community.

Each collection is composed of *items*, which are the basic archival elements of the archive. Each item is owned by one collection. Additionally, an item may appear in additional collections; however every item has one and only one owning collection.

Items are further subdivided into named *bundles* of *bitstreams*. Bitstreams are, as the name suggests, streams of bits, usually ordinary computer files. Bitstreams that are somehow closely related, for example HTML files and images that compose a single HTML document, are organized into bundles.

In practice, most items tend to have these named bundles:

- *ORIGINAL* – the bundle with the original, deposited bitstreams
- *THUMBNAILS* – thumbnails of any image bitstreams
- *TEXT* – extracted full-text from bitstreams in ORIGINAL, for indexing
- *LICENSE* – contains the deposit license that the submitter granted the host organization; in other words, specifies the rights that the hosting organization have
- *CC_LICENSE* – contains the distribution license, if any (a icenommons[109] license) associated with the item. This license specifies what end users downloading the content can do with the content

Each bitstream is associated with one *Bitstream Format*. Because preservation services may be an important aspect of the DSpace service, it is important to capture the specific formats of files that users submit. In DSpace, a bitstream format is a unique and consistent way to refer to a particular file format. An integral part of a bitstream format is an either implicit or explicit notion of how material in that format can be interpreted. For example, the interpretation for bitstreams encoded in the JPEG standard for still image compression is defined explicitly in the Standard ISO/IEC 10918-1. The interpretation of bitstreams in Microsoft Word 2000 format is defined implicitly, through reference to the Microsoft Word 2000 application. Bitstream formats can be more specific than MIME types or file suffixes. For example, *application/ms-word* and *.doc* span multiple versions of the Microsoft Word application, each of which produces bitstreams with presumably different characteristics.

Each bitstream format additionally has a *support level*, indicating how well the hosting institution is likely to be able to preserve content in the format in the future. There are three possible support levels that bitstream formats may be assigned by the hosting institution. The host institution should determine the exact meaning of each support level, after careful consideration of costs and requirements. MIT Libraries' interpretation is shown below:

Supported	The format is recognized, and the hosting institution is confident it can make bitstreams of this format usable in the future, using whatever combination of techniques (such as migration, emulation, etc.) is appropriate given the context of need.
Known	The format is recognized, and the hosting institution will promise to preserve the bitstream as-is, and allow it to be retrieved. The hosting institution will attempt to obtain enough information to enable the format to be upgraded to the 'supported' level.

109 http://www.creativecommons.org/

Unsupported	The format is unrecognized, but the hosting institution will undertake to preserve the bitstream as-is and allow it to be retrieved.

Each item has one qualified Dublin Core metadata record. Other metadata might be stored in an item as a serialized bitstream, but we store Dublin Core for every item for interoperability and ease of discovery. The Dublin Core may be entered by end-users as they submit content, or it might be derived from other metadata as part of an ingest process.

Items can be removed from DSpace in one of two ways: They may be 'withdrawn', which means they remain in the archive but are completely hidden from view. In this case, if an end-user attempts to access the withdrawn item, they are presented with a 'tombstone,' that indicates the item has been removed. For whatever reason, an item may also be 'expunged' if necessary, in which case all traces of it are removed from the archive.

Object	**Example**
Community	Laboratory of Computer Science; Oceanographic Research Center
Collection	LCS Technical Reports; ORC Statistical Data Sets
Item	A technical report; a data set with accompanying description; a video recording of a lecture
Bundle	A group of HTML and image bitstreams making up an HTML document
Bitstream	A single HTML file; a single image file; a source code file
Bitstream Format	Microsoft Word version 6.0; JPEG encoded image format

1.2.11.2 Amazon S3 Support

DSpace offers two means for storing bitstreams. The first is in the file system on the server. The second is using Amazon S3. For more information, see Storage Layer(see page 686)

2 Installing DSpace

2.1 Installation Overview

> **Try out DSpace 7 before you install**
>
> If you'd like to quickly try out DSpace 7 before a full installation, see Try out DSpace 7[110] for instructions on a quick install via Docker.

As of version 7 (and above), the DSpace application is split into a "frontend" (User Interface) and a "backend" (Server API). Most institutions will want to install BOTH. However, you can decide whether to run them on the same machine or separate machines.

- The DSpace Frontend consists of a User Interface built on Angular.io[111]. It cannot be run alone, as it *requires* a valid DSpace Backend to function. The frontend provides all user-facing functionality
- The DSpace Backend consists of a Server API ("server" webapp), built on Spring Boot[112]. It can be run standalone, however it has no user interface. The backend provides all machine-based interfaces, including the REST API, OAI-PMH, SWORD (v1 and v2) and RDF.

We recommend installing the Backend **first**, as the Frontend requires a valid Backend to run properly.

2.2 Installing the Backend (Server API)

2.2.1 Backend Requirements

- UNIX-like OS or Microsoft Windows(see page 54)
- Java JDK 11 (OpenJDK or Oracle JDK)(see page 54)

110 https://wiki.lyrasis.org/display/DSPACE/Try+out+DSpace+7
111 https://angular.io/
112 https://spring.io/projects/spring-boot

- Apache Maven 3.3.x or above (Java build tool)(see page 55)
  - Configuring a Maven Proxy(see page 55)
- Apache Ant 1.10.x or later (Java build tool)(see page 55)
- Relational Database (PostgreSQL or Oracle)(see page 56)
  - PostgreSQL 11.x, 12.x or 13.x (with pgcrypto installed)(see page 56)
  - Oracle 10g or later(see page 57)
- Apache Solr 8.x (full-text index/search service)(see page 57)
- Servlet Engine (Apache Tomcat 9, Jetty, Caucho Resin or equivalent)(see page 57)
- (Optional) IP to City Database for Location-based Statistics(see page 58)
- Git (code version control)(see page 59)

2.2.1.1 UNIX-like OS or Microsoft Windows

- UNIX-like operating system (Linux, HP/UX, Mac OSX, etc.) : Many distributions of Linux/Unix come with some of the dependencies below pre-installed or easily installed via updates. You should consult your particular distribution's documentation or local system administrators to determine what is already available.
- Microsoft Windows: While DSpace can be run on Windows servers, most institutions tend to run it on a UNIX-like operating system.

2.2.1.2 Java JDK 11 (OpenJDK or Oracle JDK)

- OpenJDK download and installation instructions can be found here http://openjdk.java.net/install/. Most operating systems provide an easy path to install OpenJDK. Just be sure to install the full JDK (development kit), and not the JRE (which is often the default example).
- Oracle's Java can be downloaded from the following location: http://www.oracle.com/technetwork/java/javase/downloads/index.html. Make sure to download the appropriate version of the Java SE JDK.

Make sure to install the JDK and not just the JRE

At this time, DSpace requires the full JDK (Java Development Kit) be installed, rather than just the JRE (Java Runtime Environment). So, please be sure that you are installing the full JDK and not just the JRE.

Only JDK11 is fully supported

Older versions of Java are unsupported. This includes JDK v7-10.

Newer versions of Java may work (e.g. JDK v12-16), but we do not recommend running them in Production. We highly recommend running only Java LTS (Long Term Support) releases in Production, as non-LTS releases may not receive ongoing security fixes. As of this DSpace release, JDK11 is the most recent Java LTS release, with the next one (JDK17) being due sometime around September 2021. As soon as the next Java LTS release is available, we will analyze it for compatibility with this release of DSpace. For more information on Java releases, see the Java roadmaps for Oracle[113] and/or OpenJDK[114].

113 https://www.oracle.com/technetwork/java/java-se-support-roadmap.html
114 https://adoptopenjdk.net/support.html#roadmap

2.2.1.3 Apache Maven 3.3.x or above (Java build tool)

Maven is necessary in the first stage of the build process to assemble the installation package for your DSpace instance. It gives you the flexibility to customize DSpace using the existing Maven projects found in the *[dspace-source]/dspace/modules* directory or by adding in your own Maven project to build the installation package for DSpace, and apply any custom interface "overlay" changes.

Maven can be downloaded from http://maven.apache.org/download.html It is also provided via many operating system package managers.

Configuring a Maven Proxy

You can configure a proxy to use for some or all of your HTTP requests in Maven. The username and password are only required if your proxy requires basic authentication (note that later releases may support storing your passwords in a secured keystore, in the meantime, please ensure your *settings.xml* file (usually *${user.home}/.m2/settings.xml*) is secured with permissions appropriate for your operating system).

Example:

```
<settings>
  .
  .
  <proxies>
   <proxy>
      <active>true</active>
      <protocol>http</protocol>
      <host>proxy.somewhere.com</host>
      <port>8080</port>
      <username>proxyuser</username>
      <password>somepassword</password>
      <nonProxyHosts>www.google.com|*.somewhere.com</nonProxyHosts>
    </proxy>
  </proxies>
  .
  .
</settings>
```

2.2.1.4 Apache Ant 1.10.x or later (Java build tool)

> While Apache Ant recommends using v1.10.x for Java 11, we've also had some success with recent versions of 1.9.x (specifically v1.9.15 seems to work fine with Java 11). That said, earlier versions of v1.9.x are not compatible with Java 11.

Apache Ant is required for the second stage of the build process (deploying/installing the application). First, Maven is used to construct the installer (`[dspace-source]/dspace/target/dspace-installer`), after which Ant is used to install/deploy DSpace to the installation directory.

Ant can be downloaded from the following location: http://ant.apache.org[115] It is also provided via many operating system package managers.

2.2.1.5 Relational Database (PostgreSQL or Oracle)

PostgreSQL 11.x, 12.x or 13.x (with pgcrypto installed)

> **PostgreSQL v9.4 to v10.x may work, but those versions are less well tested.**
>
> Active development/testing on DSpace 7 has occurred on PostgreSQL v11.x, v12.x and v13.x. However, it is likely that the backend would also function on PostgreSQL v9.4 - v10.x. At this time we have not performed sufficient testing on these earlier versions to add them to the prerequisites listing.
>
> DSpace 7 will *definitely not function on versions below 9.4* as DSpace requires installing and running the pgcrypto extension[116] (see below) v1.1, which was not available until PostgreSQL v9.4.

- PostgreSQL can be downloaded from http://www.postgresql.org/. It is also provided via many operating system package managers.
  - If the version of Postgres provided by your package manager is outdated, you may wish to use one of the official PostgreSQL provided repositories:
    - Linux users can select their OS of choice for detailed instructions on using the official PostgreSQL apt or yum repository: http://www.postgresql.org/download/linux/
    - Windows users will need to use the windows installer: http://www.postgresql.org/download/windows/
    - Mac OSX users can choose their preferred installation method: http://www.postgresql.org/download/macosx/
- Install the pgcrypto extension.[117] It will also need to be enabled on your DSpace Database (see Installation instructions below for more info). The pgcrypto extension allows DSpace to create UUIDs (universally unique identifiers) for all objects in DSpace, which means that (internal) object identifiers are now globally unique and no longer tied to database sequences.
  - On most Linux operating systems (Ubuntu, Debian, RedHat), this extension is provided in the "postgresql-contrib" package in your package manager. So, ensure you've installed "postgresql-contrib".
  - On Windows, this extension should be provided automatically by the installer (check your "[PostgreSQL]/share/extension" folder for files starting with "pgcrypto")
- Unicode (specifically UTF-8) support must be enabled (but this is enabled by default).
- Once installed, you need to enable TCP/IP connections (DSpace uses JDBC):
  - In `postgresql.conf`: uncomment the line starting: `listen_addresses = 'localhost'`. This is the default, in recent PostgreSQL releases, but you should at least check it.
  - Then tighten up security a bit by editing `pg_hba.conf` and adding this line:

```
host dspace dspace 127.0.0.1 255.255.255.255 md5
```

115 http://ant.apache.org/
116 http://www.postgresql.org/docs/9.4/static/pgcrypto.html
117 http://www.postgresql.org/docs/9.4/static/pgcrypto.html

This should appear *before* any lines matching `all` databases, because the first matching rule governs.
- Then restart PostgreSQL.

Oracle 10g or later

> ⚠ Please be aware that all active development occurs on PostgreSQL at this time. However, we provide Oracle as a secondary option if you are less comfortable with PostgreSQL.

- Details on acquiring Oracle can be downloaded from the following location: http://www.oracle.com/database/. You will need to create a database for DSpace. Make sure that the character set is one of the Unicode character sets. DSpace uses UTF-8 natively, and it is suggested that the Oracle database use the same character set. You will also need to create a user account for DSpace (e.g. *dspace*) and ensure that it has permissions to add and remove tables in the database. Refer to the Quick Installation for more details.
  - **NOTE:** If the database server is not on the same machine as DSpace, you must install the Oracle client to the DSpace server and point `tnsnames.ora` and `listener.ora` files to the database the Oracle server.

2.2.1.6 Apache Solr 8.x (full-text index/search service)

> ⚠ Make sure to install Solr with Authentication disabled (which is the default). DSpace does not yet support authentication to Solr (see https://github.com/DSpace/DSpace/issues/3169). Instead, we recommend placing Solr behind a firewall and/or ensuring port 8983 (which Solr runs on) is not available for public/anonymous access on the web. Solr only needs to be accessible to requests from the DSpace backend.

Solr can be obtained at the Apache Software Foundation site for Lucene and Solr[118]. You may wish to read portions of the quick-start tutorial[119] to make yourself familiar with Solr's layout and operation. Unpack a Solr .tgz or .zip archive in a place where you keep software that is not handled by your operating system's package management tools, and arrange to have it running whenever DSpace is running. You should ensure that Solr's index directories will have plenty of room to grow. You should also ensure that port 8983 is not in use by something else, or configure Solr to use a different port.

If you are looking for a good place to put Solr, consider `/opt` or `/usr/local`. You can simply unpack Solr in one place and use it. Or you can configure Solr to keep its indexes elsewhere, if you need to – see the Solr documentation for how to do this.

It is not necessary to dedicate a Solr instance to DSpace, if you already have one and want to use it. Simply copy DSpace's cores to a place where they will be discovered by Solr. See below.

2.2.1.7 Servlet Engine (Apache Tomcat 9, Jetty, Caucho Resin or equivalent)

- **Apache Tomcat 9.** Tomcat can be downloaded from the following location: http://tomcat.apache.org[120]. It is also provided via many operating system package managers.
  - *The Tomcat owner* (i.e. the user that Tomcat runs as) ***must have read/write access*** *to the DSpace installation directory* (i.e. `[dspace]`). There are a few common ways this may be achieved:

118 https://lucene.apache.org/solr
119 http://lucene.apache.org/solr/guide/7_7/solr-tutorial.html
120 http://tomcat.apache.org/whichversion.html

    - One option is to specifically give the Tomcat user (often named "tomcat") ownership of the [dspace] directories, for example:

      ```
      # Change [dspace] and all subfolders to be owned by "tomcat"
      chown -R tomcat:tomcat [dspace]
      ```

    - Another option is to have Tomcat itself *run as* a new user named "dspace" (see installation instructions below). Some operating systems make modifying the Tomcat "run as" user easily modifiable via an environment variable named TOMCAT_USER. This option may be more desirable if you have multiple Tomcat instances running, and you do not want all of them to run under the same Tomcat owner.
  - You need to ensure that Tomcat a) has enough memory to run DSpace, and b) uses UTF-8 as its default file encoding for international character support. So ensure in your startup scripts (etc) that the following environment variable is set: *JAVA_OPTS="-Xmx512M -Xms64M -Dfile.encoding=UTF-8"*
  - **Modifications in *[tomcat]/conf/server.xml*** : You also need to alter Tomcat's default configuration to support searching and browsing of multi-byte UTF-8 correctly. You need to add a configuration option to the *<Connector>* element in *[tomcat]/config/server.xml*: *URIEncoding="UTF-8"* e.g. if you're using the default Tomcat config, it should read:

    ```
    <!-- Define a non-SSL HTTP/1.1 Connector on port 8080 -->
    <Connector port="8080"
                   minSpareThreads="25"
                   enableLookups="false"
                   redirectPort="8443"
                   connectionTimeout="20000"
                   disableUploadTimeout="true"
                   URIEncoding="UTF-8"/>
    ```

    You may change the port from 8080 by editing it in the file above, and by setting the variable *CONNECTOR_PORT* in *server.xml*. You should set the URIEncoding even if you are running Tomcat behind a proxy (Apache HTTPD, Nginx, etc.) via AJP.
- **Jetty or Caucho Resin**
  - DSpace 7 has not been tested with Jetty or Caucho Resin, after the switch to Java 11
  - Older versions of DSpace were able to run on a Tomcat-equivalent servlet Engine, such as Jetty (https://www.eclipse.org/jetty/) or Caucho Resin (http://www.caucho.com/). If you choose to use a different servlet container, please ensure that it supports Servlet Spec 3.1 (or above).
  - Jetty and Resin are configured for correct handling of UTF-8 by default.

2.2.1.8 (Optional) IP to City Database for Location-based Statistics

Optionally, if you wish to record the geographic locations of clients in DSpace usage statistics records, you will need to install (and regularly update) one of the following:

- Either, a copy of MaxMind's GeoLite City database[121] (in MMDB format)
  - NOTE: Installing MaxMind GeoLite2 is *free.* However, you **must** sign up for a (free) MaxMind account in order to obtain a license key to use the GeoLite2 database.
  - You may download GeoLite2 directly from MaxMind, or many Linux distributions provide the `geoipupdate` tool directly via their package manager. You will still need to configure your license key prior to usage.

121 https://dev.maxmind.com/geoip/geoip2/geolite2/

    - Once the "GeoLite2-City.mmdb" database file is installed on your system, you will need to configure its location as the value of `usage-statistics.dbfile` in your `local.cfg` configuration file.
    - See the "Managing the City Database File" section of SOLR Statistics(see page 338) for more information about using a City Database with DSpace.
- Or, you can alternatively use/install DB-IP's City Lite database[122] (in MMDB format)
    - This database is also free to use, but does **not** require an account to download.
    - Once the "dbip-city-lite.mmdb" database file is installed on your system, you will need to configure its location as the value of `usage-statistics.dbfile` in your `local.cfg` configuration file.
    - See the "Managing the City Database File" section of SOLR Statistics(see page 338) for more information about using a City Database with DSpace.

2.2.1.9 Git (code version control)

Currently, there is a known bug in DSpace where a third-party Maven Module expects `git` to be available (in order to support the `./dspace version` commandline tool). We are working on a solution within this ticket:
DS-3418[123] - DSpace Build Error When Git is Not Present **VOLUNTEER NEEDED**

For the time being, you can work around this problem by installing Git locally: https://git-scm.com/downloads

2.2.2 Backend Installation

1. Install all the Backend Requirements(see page 53) listed above.
2. **Create a DSpace operating system user *(optional)*** . As noted in the prerequisites above, Tomcat (or Jetty, etc) **must run as** an operating system user account that has full read/write access to the DSpace installation directory (i.e. `[dspace]`). Either you must ensure the Tomcat owner also owns `[dspace]`, OR you can create a new "dspace" user account, and ensure that Tomcat also runs as that account:

   ```
   useradd -m dspace
   ```

   The choice that makes the most sense for you will probably depend on how you installed your servlet container (Tomcat/Jetty/etc). If you installed it from source, you will need to create a user account to run it, and that account can be named anything, e.g. 'dspace'. If you used your operating system's package manager to install the container, then a user account should have been created as part of that process and it will be much easier to use that account than to try to change it.
3. **Download** the latest DSpace release[124] from the DSpace GitHub Repository. You can choose to either download the zip or tar.gz file provided by GitHub, or you can use "git" to checkout the appropriate tag (e.g. `dspace-7.0`) or branch.
4. **Unpack the DSpace software**. After downloading the software, based on the compression file format, choose one of the following methods to unpack your software:
    a. **Zip file**. If you downloaded *dspace-7.0.zip* do the following:

    ```
    unzip dspace-7.0.zip
    ```

    b. **.gz file**. If you downloaded *dspace-7.0.tar.gz* do the following:

122 https://db-ip.com/db/download/ip-to-city-lite
123 https://jira.lyrasis.org/browse/DS-3418?src=confmacro
124 https://github.com/DSpace/DSpace/releases

```
gunzip -c dspace-7.0.tar.gz | tar -xf -
```

For ease of reference, we will refer to the location of this unzipped version of the DSpace release as *[dspace-source]* in the remainder of these instructions. After unpacking the file, the user may wish to change the ownership of the dspace-7.x folder to the "dspace" user. (And you may need to change the group).

5. **Database Setup**
    - *PostgreSQL:*
        - Create a `dspace` database user (this user can have any name, but we'll assume you name it "dspace"). This is entirely separate from the `dspace` operating-system user created above:

          ```
          createuser --username=postgres --no-superuser --pwprompt dspace
          ```

          You will be prompted (twice) for a password for the new `dspace` user. Then you'll be prompted for the password of the PostgreSQL superuser (`postgres`).
        - Create a `dspace` database, owned by the `dspace` PostgreSQL user. Similar to the previous step, this can only be done by a "superuser" account in PostgreSQL (e.g. `postgres`):

          ```
          createdb --username=postgres --owner=dspace --encoding=UNICODE dspace
          ```

          You will be prompted for the password of the PostgreSQL superuser (`postgres`).
        - Finally, you MUST enable the pgcrypto extension[125] on your new dspace database. Again, this can only be enabled by a "superuser" account (e.g. `postgres`)

          ```
          # Login to the database as a superuser, and enable the pgcrypto extension on this
           database
          psql --username=postgres dspace -c "CREATE EXTENSION pgcrypto;"
          ```

          The "CREATE EXTENSION" command should return with no result if it succeeds. If it fails or throws an error, it is likely you are missing the required pgcrypto extension (see Database Prerequisites[126] above).
            - ***Alternative method: How to enable pgcrypto via a separate database schema.*** While the above method of enabling pgcrypto is perfectly fine for the majority of users, there may be some scenarios where a database administrator would prefer to install extensions into a database schema that is *separate from* the DSpace tables. Developers also may wish to install pgcrypto into a separate schema if they plan to "clean" (recreate) their development database frequently. Keeping extensions in a separate schema from the DSpace tables will ensure developers would NOT have to continually re-enable the extension each time you run a "`./dspace database clean`". If you wish to install pgcrypto in a separate schema here's how to do that:

125 http://www.postgresql.org/docs/9.4/static/pgcrypto.html
126 https://wiki.duraspace.org/display/DSDOC6x/Installing+DSpace#InstallingDSpace-RelationalDatabase:(PostgreSQLorOracle)

```
# Login to the database as a superuser
psql --username=postgres dspace
# Create a new schema in this database named "extensions" (or whatever you want
to name it)
CREATE SCHEMA extensions;
# Enable this extension in this new schema
CREATE EXTENSION pgcrypto SCHEMA extensions;
# Grant rights to call functions in the extensions schema to your dspace user
GRANT USAGE ON SCHEMA extensions TO dspace;


# Append "extensions" on the current session's "search_path" (if it doesn't
already exist in search_path)
# The "search_path" config is the list of schemas that Postgres will use
SELECT set_config('search_path',current_setting('search_path') || ',extensions',f
alse) WHERE current_setting('search_path') !~ '(^|,)extensions(,|$)';
# Verify the current session's "search_path" and make sure it's correct
SHOW search_path;
# Now, update the "dspace" Database to use the same "search_path" (for all
future sessions) as we've set for this current session (i.e. via set_config()
above)
ALTER DATABASE dspace SET search_path FROM CURRENT;
```

- *Oracle:*
  - Setting up DSpace to use Oracle is a bit different now. You will need still need to get a copy of the Oracle JDBC driver, but instead of copying it into a lib directory you will need to install it into your local Maven repository. (You'll need to download it first from this location: http://www.oracle.com/technetwork/database/enterprise-edition/jdbc-112010-090769.html.) Run the following command (all on one line):

    ```
    mvn install:install-file
        -Dfile=ojdbc6.jar
        -DgroupId=com.oracle
        -DartifactId=ojdbc6
        -Dversion=11.2.0.4.0
        -Dpackaging=jar
        -DgeneratePom=true
    ```

  - You need to compile DSpace with an Oracle driver (ojdbc6.jar) corresponding to your Oracle version - update the version in *[dspace-source]/pom.xml* E.g.:

    ```
    <dependency>
      <groupId>com.oracle</groupId>
      <artifactId>ojdbc6</artifactId>
      <version>11.2.0.4.0</version>
    </dependency>
    ```

  - Create a database for DSpace. Make sure that the character set is one of the Unicode character sets. DSpace uses UTF-8 natively, and it is required that the Oracle database use the same character set. Create a user account for DSpace (e.g. *dspace*) and ensure that it has permissions to add and remove tables in the database.

- NOTE: You will need to ensure the proper `db.*` settings are specified in your `local.cfg` file (see next step), as the defaults for all of these settings assuming a PostgreSQL database backend.

```
db.url = jdbc:oracle:thin:@host:port/SID
# e.g. db.url = jdbc:oracle:thin:@//localhost:1521/xe
# NOTE: in db.url, SID is the SID of your database defined in tnsnames.ora
# the default Oracle port is 1521
# You may also use a full SID definition, e.g.
# db.url = jdbc:oracle:thin:@(description=(address_list=(address=(protocol=TCP)
(host=localhost)(port=1521)))(connect_data=(service_name=DSPACE)))

# Oracle driver and dialect
db.driver = oracle.jdbc.OracleDriver
db.dialect = org.hibernate.dialect.Oracle10gDialect

# Specify DB username, password and schema to use
db.username =
db.password =
db.schema = ${db.username}
# For Oracle, schema is equivalent to the username of your database account,
# so this may be set to ${db.username} in most scenarios
```

- Later, during the Maven build step, don't forget to specify `mvn -Ddb.name`[127]`=oracle package`

6. **Initial Configuration (local.cfg):** Create your own `[dspace-source]/dspace/config/local.cfg` configuration file. You may wish to simply copy the provided `[dspace-source]/dspace/config/local.cfg.EXAMPLE`. This local.cfg file can be used to store *any* configuration changes that you wish to make which are local to your installation (see local.cfg configuration file(see page 560) documentation). ANY setting may be copied into this local.cfg file from the dspace.cfg or any other *.cfg file in order to override the default setting (see note below). For the initial installation of DSpace, there are some key settings you'll likely want to override. Those are provided in the `[dspace-source]/dspace/config/local.cfg.EXAMPLE`. (NOTE: Settings followed with an asterisk (*) are highly recommended, while all others are optional during initial installation and may be customized at a later time.)
    - `dspace.dir*` - must be set to the *[dspace]* (installation) directory (*NOTE: On Windows be sure to use forward slashes for the directory path!* For example: "`C:/dspace`" is a valid path for Windows.)
    - `dspace.server.url*` - complete URL of this DSpace backend (including port and any subpath). **Do not end with '/'**. For example: http://localhost:8080/server
    - `dspace.ui.url*` - complete URL of the DSpace frontend (including port and any subpath). REQUIRED for the REST API to fully trust requests from the DSpace frontend. **Do not end with '/'**. For example: http://localhost:4000
    - `dspace.name` - Human-readable, "proper" name of your server, e.g. "My Digital Library".
    - `solr.server*` - complete URL of the Solr server. DSpace makes use of Solr[128] for indexing purposes. http://localhost:8983/solr unless you changed the port or installed Solr on some other host.
    - `default.language` – Default language for all metadata values (defaults to "en_US")
    - `db.url*` – The full JDBC URL to your database (examples are provided in the `local.cfg.EXAMPLE`)

127 http://Ddb.name
128 http://lucene.apache.org/solr/

- `db.driver*` – Which database driver to use, based on whether you are using PostgreSQL or Oracle

- `db.dialect*` – Which database dialect to use, based on whether you are using PostgreSQL or Oracle
- `db.username*` - the database username used in the previous step.
- `db.password*` - the database password used in the previous step.
- `db.schema*` - the database schema to use (examples are provided in the local.cfg.EXAMPLE)
- `mail.server` - fully-qualified domain name of your outgoing mail server.
- `mail.from.address` - the "From:" address to put on email sent by DSpace.
- `mail.feedback.recipient` - mailbox for feedback mail.
- `mail.admin` - mailbox for DSpace site administrator.
- `alert.recipient` - mailbox for server errors/alerts (not essential but very useful!)
- `registration.notify`- mailbox for emails when new users register (optional)

> **Your local.cfg file can override ANY settings from other *.cfg files in DSpace**
>
> The provided `local.cfg.EXAMPLE` only includes a small subset of the configuration settings available with DSpace. It provides a good starting point for your own `local.cfg` file.
>
> However, you should be aware that ANY configuration can now be copied into your `local.cfg` to override the default settings. This includes ANY of the settings/ configurations in:
>
> - The primary dspace.cfg file (`[dspace]/config/dspace.cfg`)
> - Any of the module configuration files (`[dspace]/config/modules/*.cfg` files)
> - Any of the Spring Boot settings (`[dspace-src]/dspace-server-webapp/src/main/resources/application.properties`)
>
> Individual settings may also be commented out or removed in your `local.cfg`, in order to re-enable default settings.
>
> See the Configuration Reference(see page 552) section for more details.

7. **DSpace Directory:** Create the directory for the DSpace backend installation (i.e. `[dspace]`). As *root* (or a user with appropriate permissions), run:

```
mkdir [dspace]
chown dspace [dspace]
```

(Assuming the *dspace* UNIX username.)

8. **Build the Installation Package:** As the *dspace* UNIX user, generate the DSpace installation package.

```
cd [dspace-source]
mvn package
```

> **Building with Oracle Database Support**
>
> Without any extra arguments, the DSpace installation package is initialized for PostgreSQL. If you want to use Oracle instead, you should build the DSpace installation package as follows:
> `mvn -Ddb.name`[129]`=oracle package`

9. **Install DSpace Backend:** As the *dspace* UNIX user, install DSpace to `[dspace]`:

```
cd [dspace-source]/dspace/target/dspace-installer
ant fresh_install
```

> To see a complete list of build targets, run: `ant help` *The most likely thing to go wrong here is the test of your database connection. See the* Common Installation Issues(see page 71) *Section below for more details.*

10. **Initialize your Database:** While this step is optional (as the DSpace database should auto-initialize itself on first startup), it's always good to verify one last time that your database connection is working properly. To initialize the database run:

```
[dspace]/bin/dspace database migrate
```

11. **Deploy Server web application:** The DSpace backend consists of a single "server" webapp (in `[dspace]/webapps/server`). You need to deploy this webapp into your Servlet Container (e.g. Tomcat). Generally, there are two options (or techniques) which you could use...either configure Tomcat to find the DSpace "server" webapp, or copy the "server" webapp into Tomcat's own webapps folder.
    - *Technique A.* Tell your Tomcat/Jetty/Resin installation where to find your DSpace web application(s). As an example, in the directory `[tomcat]/conf/Catalina/localhost` you could add files similar to the following (but replace `[dspace]` with your installation location):

      DEFINE A CONTEXT PATH FOR DSpace Server webapp: server.xml

      ```
      <?xml version='1.0'?>
      <Context
          docBase="[dspace]/webapps/server"/>
      ```

      The name of the file (not including the suffix ".xml") will be the name of the context, so for example `server.xml` defines the context at `http://host:8080/server`. To define the *root context* (`http://host:8080/`), name that context's file `ROOT.xml`. Optionally, you can also choose to install the old, deprecated "rest" webapp if you
    - *Technique B.* Simple and complete. You copy only (or all) of the DSpace Web application(s) you wish to use from the [dspace]/webapps directory to the appropriate directory in your Tomcat/Jetty/Resin installation. For example:
      `cp -R [dspace]/webapps/* [tomcat]/webapps*` (This will copy all the web applications to Tomcat).
      `cp -R [dspace]/webapps/server [tomcat]/webapps*` (This will copy only the Server web application to Tomcat.)

129 http://Ddb.name

To define the *root context* (`http://host:8080/`), name that context's directory `ROOT`.

12. **Optionally, also install the deprecated DSpace 6.x REST API web application.** If you previously used the DSpace 6.x REST API, for backwards compatibility the old, deprecated "rest" webapp is still available to install (in `[dspace]/webapps/rest`). It is NOT used by the DSpace frontend. So, most users should skip this step.
13. **Copy Solr cores:** DSpace installation creates a set of four empty Solr cores already configured.
    a. Copy them from `[dspace]`/solr to the place where your Solr instance will discover them. For example:

    ```
    cp -R [dspace]/solr/* [solr]/server/solr/configsets
    chown -R solr:solr [solr]/server/solr/configsets
    ```

    b. Start (or re-start) Solr. For example:

    ```
    [solr]/bin/solr restart
    ```

    c. You can check the status of Solr and your new DSpace cores by using its administrative web interface. Browse to `http://localhost:8983/` to see if Solr is running well, then look at the cores by selecting (on the left) Core Admin or using the Core Selector drop list.
14. **Create an Administrator Account:** Create an initial administrator account from the command line:

    ```
    [dspace]/bin/dspace create-administrator
    ```

15. **Initial Startup!** Now the moment of truth! Start up (or restart) Tomcat/Jetty/Resin.
    a. *REST API Interface* - (e.g.) http://dspace.myu.edu:8080/server/
    b. *OAI-PMH Interface* - (e.g.) http://dspace.myu.edu:8080/server/oai/request?verb=Identify
    c. For an example of what the default backend looks like, visit the Demo Backend: https://api7.dspace.org/server/
16. **Production Installation (adding HTTPS support)**: Running the DSpace Backend on HTTP & port 8080 is only usable for testing/demo environments. **If you want to run DSpace in Production, you MUST run the backend with HTTPS support** (otherwise logins will not work outside of your local domain).
    a. For HTTPS support, we recommend installing either Apache HTTPD[130] or Nginx[131], configuring SSL at that level, and proxying all requests to your Tomcat installation. Keep in mind, if you want to host both the DSpace Backend and Frontend on the same server, you can use one installation of Apache HTTPD or Nginx to manage HTTPS/SSL and proxy to both.
    b. These instructions are specific to Apache HTTPD, but a similar setup can be achieved with Nginx
        i. Install Apache HTTPD[132], e.g. `sudo apt install apache2`
        ii. Install the mod_proxy[133] and mod_proxy_ajp[134] modules, e.g. `sudo en2mod proxy; sudo a2enmod proxy_ajp`
            1. Alternatively, you can choose to use mod_proxy_http[135] to create an http proxy. A separate example is commented out below
        iii. Restart Apache to enable

130 https://httpd.apache.org/
131 https://www.nginx.com/
132 https://httpd.apache.org/
133 https://httpd.apache.org/docs/current/mod/mod_proxy.html
134 https://httpd.apache.org/docs/current/mod/mod_proxy_ajp.html
135 https://httpd.apache.org/docs/current/mod/mod_proxy_http.html

iv. For mod_proxy_ajp to communicate with Tomcat, you'll need to enable Tomcat's AJP connector in your Tomcat's server.xml:

```
<Connector protocol="AJP/1.3" port="8009" redirectPort="8443" URIEncoding="UTF-8" />
```

v. Now, setup a new VirtualHost for your site (using HTTPS / port 443) which proxies all requests to Tomcat's AJP connector (running on port 8009)

```
<VirtualHost _default_:443>
    .. setup your host how you want, including log settings...

    SSLEngine on
    SSLCertificateFile [full-path-to-PEM-cert]
    SSLCertificateKeyFile [full-path-to-cert-KEY]

    # Proxy all HTTPS requests to "/server" from Apache to Tomcat via AJP connector
    ProxyPass /server ajp://localhost:8009/server
    ProxyPassReverse /server ajp://localhost:80009/server

    # If you would rather use mod_proxy_http as an http proxy to port 8080
    # then use these settings instead
    #ProxyPass /server http://localhost:8080/server
    #ProxyPassReverse /server http://localhost:8080/server
    # When using mod_proxy_http, you need to also ensure the X-Forwarded-Proto header
is sent
    # to tell DSpace it is behind HTTPS, otherwise some URLs may continue to use HTTP
    # (requires installing/enabling mod_headers)
    #RequestHeader set X-Forwarded-Proto https
</VirtualHost>
```

c. After switching to HTTPS, make sure to go back and update the URLs (e.g. `dspace.server.url`) in your local.cfg to match the new URL of your backend. This will require briefly rebooting Tomcat.

2.3 Installing the Frontend (User Interface)

2.3.1 Frontend Requirements

- UNIX-like OS or Microsoft Windows(see page 66)
- Node.js (v12.x or v14.x)(see page 67)
- Yarn (v1.x)(see page 67)
- PM2 (or another Process Manager for Node.js apps) (optional, but recommended for Production)(see page 67)
- DSpace 7.x Backend (see above)(see page 67)

2.3.1.1 UNIX-like OS or Microsoft Windows

- UNIX-like operating system (Linux, HP/UX, Mac OSX, etc.) : Many distributions of Linux/Unix come with some of the dependencies below pre-installed or easily installed via updates. You should consult your particular distribution's documentation or local system administrators to determine what is already available.

- Microsoft Windows: While DSpace can be run on Windows servers, most institutions tend to run it on a UNIX-like operating system.

2.3.1.2 Node.js (v12.x or v14.x)

- Node.js can be found at https://nodejs.org/. It may be available through your Linux distribution's package manager. We recommend running a Long Term Support (LTS) version[136] (even numbered releases). Non-LTS versions (odd numbered releases) are not recommended.
- Node.js is a Javascript runtime that also provides npm[137] (Node Package Manager). It is used to both build and run the frontend.

2.3.1.3 Yarn (v1.x)

- Yarn v1.x is available at https://classic.yarnpkg.com/. It can usually be install via NPM (or through your Linux distribution's package manager). *We do NOT currently support Yarn v2.*

```
# You may need to run this command using "sudo" if you don't have proper privileges
npm install --global yarn
```

- Yarn is used to build/install the frontend.

2.3.1.4 PM2 (or another Process Manager for Node.js apps) (*optional, but recommended for Production*)

- In Production scenarios, we *highly recommend* starting/stopping the User Interface using a Node.js process manager. There are several available, but our current favorite is PM2[138]. The rest of this installation guide assumes you are using PM2.
- PM2[139] is very easily installed via NPM

```
# You may need to run this command using "sudo" if you don't have proper privileges
npm install --global pm2
```

2.3.1.5 DSpace 7.x Backend (see above)

- The DSpace User Interface (Frontend) cannot function without an installed DSpace Backend. Follow the instructions above.
- The Frontend and Backend *do not need to be installed on the same machine/server*. They may be installed on separate machines as long as the two machines can connect to one another via HTTP or HTTPS.

2.3.2 Frontend Installation

1. First, install all the Frontend Requirements(see page 66) listed above & verify the backend/REST API is publicly accessible.

136 https://nodejs.org/en/about/releases/
137 https://www.npmjs.com/
138 https://pm2.keymetrics.io/
139 https://pm2.keymetrics.io/

2. Download the latest dspace-angular release[140] from the DSpace GitHub repository. You can choose to either download the zip or tar.gz file provided by GitHub, or you can use "git" to checkout the appropriate tag (e.g. `dspace-7.0`) or branch.
3. Install all necessary local dependencies by running the following from within the unzipped "dspace-angular" directory

```
# change directory to our repo
cd dspace-angular

# install the local dependencies
yarn install
```

4. Create a Production Configuration file at `[dspace-angular]/src/environments/environment.prod.ts`. You may wish to use the `environment.template.ts` as a starting point. This `environment.prod.ts` file can be used to override any of the default configurations specified in the `environment.common.ts` (in that same directory). *At a minimum* this file MUST include the "ui" and "rest" sections similar to the following (keep in mind, you only need to include settings that you need to modify):

```
export const environment = {
  // The "ui" section defines where you want Node.js to run/respond. It may correspond to your
primary URL, but it also may not (if you are running behind a proxy).
  // In this example, we are setting up our UI to just use localhost, port 4000.
  // This is a common setup for when you want to use Apache or Nginx to handle HTTPS and proxy
requests to Node on port 4000
  ui: {
      ssl: false,
      host: 'localhost',
      port: 4000,
      // NOTE: Space is capitalized because 'namespace' is a reserved string in TypeScript
      nameSpace: '/'
  },
  // This example is valid if your Backend is publicly available at https://api.mydspace.edu/server/
  // The REST settings MUST correspond to the primary URL of the backend. Usually, this means they
must be kept in sync
  // with the value of "dspace.server.url" in the backend's local.cfg
  rest: {
      ssl: true,
      host: 'api.mydspace.edu',
      port: 443,
      // NOTE: Space is capitalized because 'namespace' is a reserved string in TypeScript
      nameSpace: '/server'
  }
};
```

   a. HINT #1: In the "ui" section above, you may wish to start with "ssl: false" and "port: 4000" just to be certain that everything else is working properly. With those settings, you can quickly test your UI by running "`yarn start`" and trying to access it via `http://[mydspace.edu]:4000/` from your web browser. KEEP IN MIND, we highly recommend always using HTTPS for Production.
   b. HINT #2: If Node throws an error saying "listen EADDRNOTAVAIL: address not available", try setting the "host" to "0.0.0.0" or "localhost". Usually that error is a sign that the "host" is not recognized.

140 https://github.com/DSpace/dspace-angular/releases

   c. If there are other settings you know you need to modify in the default `environment.common.ts` configuration file you can also copy them into this same file.
5. Build the User Interface for Production. This uses your `environment.prod.ts` and the source code to create a compiled version of the UI in the `[dspace-angular]/dist` folder

```
yarn run build:prod
```

   a. HINT: if you change/update your `environment.prod.ts`, *then you will need to rebuild* the UI application (i.e. rerun this command).
6. Assuming you are using PM2[141], create a JSON configuration file describing how to run our UI application. This need NOT be in the same directory as the dspace-angular codebase itself (in fact you may want to put the parent directory or another location). Keep in mind the "cwd" setting (on line 5) must be the full path to your `[dspace-angular]` folder.

dspace-angular.json

```
{
    "apps": [
        {
            "name": "dspace-angular",
            "cwd": "/home/dspace/dspace-angular",
            "script": "yarn",
            "args": "run serve:ssr",
           "interpreter": "none"
        }
    ]
}
```

7. Now, start the application using PM2 using the configuration file you created in the previous step

```
# In this example, we are assuming the config is named "dspace-angular.json"
pm2 start dspace-angular.json

# To see the logs, you'd run
# pm2 logs

# To stop it, you'd run
# pm2 stop dspace-angular.json
```

   a. For more PM2 commands see https://pm2.keymetrics.io/docs/usage/quick-start/
   b. HINT: You may also want to install/configure pm2-logrotate[142] to ensure that PM2's log folder doesn't fill up over time.
8. At this point, the User Interface should be available at the URL you configured in your `environment.prod.ts`
   a. For an example of what the default frontend looks like, visit the Demo Frontend: https://demo7.dspace.org/
9. For HTTPS (port 443) support, you have two options

141 https://pm2.keymetrics.io/
142 https://github.com/keymetrics/pm2-logrotate

a. (Recommended) You can install either Apache HTTPD[143] or Nginx[144] , configuring SSL at that level, and proxy requests to PM2 (on port 4000). This is our current recommended approach. Plus, as a bonus, if you want to host the UI and Backend on the same server, you can use just one Apache HTTPD (or Nginx) to proxy to both. These instructions are specific to Apache.
   i. Install Apache HTTPD[145], e.g. `sudo apt install apache2`
   ii. Install the mod_proxy[146] and mod_proxy_http[147] modules, e.g. `sudo en2mod proxy; sudo a2enmod proxy_http`
   iii. Restart Apache to enable
   iv. Now, setup a new VirtualHost for your site (preferably using HTTPS / port 443) which proxies all requests to PM2 running on port 4000.

```
<VirtualHost _default_:443>
    .. setup your host how you want, including log settings...

    SSLEngine on
    SSLCertificateFile [full-path-to-PEM-cert]
    SSLCertificateKeyFile [full-path-to-cert-KEY]

    # Proxy all HTTPS requests from Apache to PM2 on port 4000
    # NOTE that this proxy URL must match the "ui" settings in your environment.*.ts
    ProxyPass / http://localhost:4000/
    ProxyPassReverse / http://localhost:4000/
</VirtualHost>
```

b. (Alternatively) You can use the basic HTTPS support built into dspace-angular node server. (This may currently be better for non-Production environments as it has not been well tested)
   i. Create a `[dspace-angular]/config/ssl/` folder and add a `key.pem` and `cert.pem` to that folder (they must have those exact names)
   ii. Enable "ui.ssl" (set to true)
   iii. Update your "ui.port" to be 443
      1. In order to run Node/PM2 on port 443, you also will likely need to provide node with special permissions, like in this example[148].
   iv. Rebuild and then restart the app in PM2
   v. Keep in mind, while this setup is simple, you may not have the same level of detailed, Production logs as you would with Apache HTTPD or Nginx

10. Additional UI configurations are described in the environment.common.ts[149] and at https://github.com/DSpace/dspace-angular/blob/main/docs/Configuration.md **(More documentation will be coming soon)**

2.4 What Next?

After a successful installation, you may want to take a closer look at

- Scheduled Tasks via Cron(see page 481) : Several DSpace features **require** that a command-line script is run regularly via cron.
- Configuration Reference(see page 552) : Details on the configuration options available to the Backend

143 https://httpd.apache.org/
144 https://www.nginx.com/
145 https://httpd.apache.org/
146 https://httpd.apache.org/docs/current/mod/mod_proxy.html
147 https://httpd.apache.org/docs/current/mod/mod_proxy_http.html
148 https://levelup.gitconnected.com/tws-004-how-to-configure-nodejs-to-use-port-443-86f1ca801c5f
149 https://github.com/DSpace/dspace-angular/blob/main/src/environments/environment.common.ts

- Handle Server installation(see page 464): Optionally, you may wish to enable persistent URLs for your DSpace site using CRNI's Handle.Net Registry
- Statistics and Metrics(see page 338): Optionally, you may wish to configuration one (or more) Statistics options within DSpace, including Google Analytics(see page 363) and (internal) Solr Statistics(see page 338)
- Multilingual Support(see page 407): Optionally, you may wish to enable multilingual support in your DSpace site.
- Using DSpace(see page 87) : Various other pages which describe usage and additional configurations related to other DSpace features.
- System Administration(see page 410): Various other pages which describe additional backend installation options/configurations.

If you've run into installation problems, you may want to...

- Review Commons Installation Issues(see page 71) (see below)
- Ask for Support[150] via one of the support options documented on that page

2.5 Common Installation Issues

2.5.1 Troubleshoot an error or find detailed error messages

See the Troubleshoot an error[151] page, look for the section on "DSpace 7.x". This will provide you hints on locating error messages both in the User Interface (frontend) and in the REST API (backend)

2.5.2 "CORS error" or "Invalid CORS request"

If you are seeing a CORS error in your browser, this means that you are accessing the REST API via an "untrusted" client application. To fix this error, you must change your REST API / Backend configuration to trust the application.

- By default, the DSpace REST API / Backend will only trust the application at `dspace.ui.url`. Therefore, you should first verify that your `dspace.ui.url` setting (in your local.cfg) exactly matches the *primary URL* of your User Interface (i.e. the URL you see in the browser). This must be an exact match: mode (http vs https), domain, port, and subpath(s) all must match.
- If you need to trust *additional* client applications / URLs, those MUST be added to the `rest.cors.allowed-origins` configuration. See REST API(see page 502) for details on this configuration.
- Also, check your Tomcat (or servlet container) log files. If Tomcat throws a syntax or other major error, it may return an error response that triggers a CORS error. In this scenario, the CORS error is only a side effect of a larger error.

If you modify either of the above settings, you will need to restart Tomcat for the changes to take effect.

2.5.3 "403 Forbidden" error with a message that says "Access is denied. Invalid CSRF Token"

First, double check that you are seeing that exact error message. A `403 Forbidden` error may be thrown in a variety of scenarios. For example, a 403 may be thrown if a page requires a login, if you have entered an invalid username or password, or even sometimes when there is a CORS error (see previous installation issue for how to solve that).

150 https://wiki.lyrasis.org/display/DSPACE/Support
151 https://wiki.lyrasis.org/display/DSPACE/Troubleshoot+an+error

If you are seeing the message "Invalid CSRF Token" message (especially on every login), this is usually the result of a configuration / setup issue.

Here's some things you should double check:

1. If you need to be able to login to the REST API from other domains, then **your Backend must be running HTTPS.**
    a. If the REST API Backend is running HTTP, then it will always send the required `DSPACE-XSRF-COOKIE` cookie with a value of `SameSite=Lax`. This setting means that the cookie will *not* be sent (by your browser) to any other domains. Effectively, this will block all logins from any domain that is not the same as the REST API (as this cookie will not be sent back to the REST API as required for CSRF validation). In other words, running the REST API on HTTP is only possible if the User Interface is running on the exact same domain. For example, running both on 'localhost' with HTTP is a common development setup, and this will work fine.
    b. In order to allow for cross-domain logins, you MUST enable HTTPS on the REST API. This will result in the `DSPACE-XSRF-COOKIE` cookie being set to `SameSite=None; Secure`. This setting means the cookie will be sent cross domain, but only for HTTPS requests. It also allows the user interface (or other client applications) to be on any domain, provided that the domain is trusted by CORS (see `rest.cors.allowed-origins` setting in REST API(see page 502))
2. Verify that your User Interface's "rest" section matches the value of "`dspace.server.url`" configuration on the Backend. This simply ensures your UI is sending requests to the correct REST API. Also pay close attention that both specify HTTPS when necessary (see previous bullet).
3. Verify that your "`dspace.server.url`" configuration on the Backend matches the primary URL of the REST API (i.e. the URL you see in the browser). This must be an exact match: mode (http vs https), domain, port, and subpath(s) all must match, and it *must not end in a trailing slash* (e.g. "https://api7.dspace.org/server" is valid, but "https://api7.dspace.org/server/" may cause problems).
4. Verify that your "`dspace.ui.url`" configuration on the Backend matches the primary URL of your User Interface (i.e. the URL you see in the browser). This must be an exact match: mode (http vs https), domain, port, and subpath(s) all must match, and it *must not end in a trailing slash* (e.g. "https://demo7.dspace.org" is valid, but "https://demo7.dspace.org/" may cause problems).
5. Verify that nothing (e.g. a proxy) is blocking Cookies and HTTP Headers from being passed between the UI and REST API. DSpace's CSRF protection relies on the client (User Interface) being able to return both a valid `DSPACE-XSRF-COOKIE` cookie and a matching `X-XSRF-TOKEN` header back to the REST API for validation. See our REST Contract for more details https://github.com/DSpace/RestContract/blob/main/csrf-tokens.md
6. If you are running a custom application, or accessing the REST API from the command-line (or other third party tool like Postman[152]), you MUST ensure you are sending the CSRF token on every modifying request. See our REST Contract for more details https://github.com/DSpace/RestContract/blob/main/csrf-tokens.md

For additional information on how DSpace's CSRF Protection works, see our REST Contract at https://github.com/DSpace/RestContract/blob/main/csrf-tokens.md

2.5.4 Using a Self-Signed SSL Certificate causes the Frontend to not be able to access the Backend

If you setup the backend to use HTTPS with a self-signed SSL certificate, then Node.js (which the frontend runs on) may not "trust" that certificate by default. This will result in the Frontend not being able to make requests to the Backend.

One possible workaround (untested as of yet) is to try setting the NODE_EXTRA_CA_CERTS environment variable[153] (which tells Node.js to trust additional CA certificates).

152 https://www.postman.com/
153 https://nodejs.org/api/cli.html#cli_node_extra_ca_certs_file

Another option is to avoid using a self-signed SSL certificate. Instead, create a real, issued SSL certificate using something like Let's Encrypt[154] (or similar free services)

2.5.5 My REST API is running under HTTPS, but some of its "link" URLs are switching to HTTP?

This scenario may occur when you are running the REST API behind an HTTP proxy (e.g. Apache HTTPD's `mod_proxy_http`, Ngnix's `proxy_pass` or any other proxy that is forwarding from HTTPS to HTTP).

The fix is to ensure the DSpace REST API is sent the `X-Forwarded-Proto` header (by your proxying service), telling it that the forwarded protocol is HTTPS

```
X-Forwarded-Proto: https
```

In general, when running behind a proxy, the DSpace REST API depends on accurate X-Forwarded-* headers to be sent by that proxy.

2.5.6 **Database errors occur when you run** `ant fresh_install`

There are two common errors that occur.

- If your error looks like this:

  ```
  [java] 2004-03-25 15:17:07,730 INFO
         org.dspace.storage.rdbms.InitializeDatabase @ Initializing Database
  [java] 2004-03-25 15:17:08,816 FATAL
         org.dspace.storage.rdbms.InitializeDatabase @ Caught exception:
  [java] org.postgresql.util.PSQLException: Connection refused. Check
         that the hostname and port are correct and that the postmaster is
         accepting TCP/IP connections.
  [java]     at
         org.postgresql.jdbc1.AbstractJdbc1Connection.openConnection(AbstractJd
  bc1Connection.java:204)
  [java]     at org.postgresql.Driver.connect(Driver.java:139)
  ```

  it usually means you haven't yet added the relevant configuration parameter to your PostgreSQL configuration (see above), or perhaps you haven't restarted PostgreSQL after making the change. Also, make sure that the *db.username* and *db.password* properties are correctly set in *[dspace]/config/dspace.cfg*. An easy way to check that your DB is working OK over TCP/IP is to try this on the command line:

  ```
  psql -U dspace -W -h localhost
  ```

  Enter the *dspace* database password, and you should be dropped into the psql tool with a *dspace*=> prompt.
- Another common error looks like this:

[154] https://letsencrypt.org/

```
[java] 2004-03-25 16:37:16,757 INFO
        org.dspace.storage.rdbms.InitializeDatabase @ Initializing Database
[java] 2004-03-25 16:37:17,139 WARN
        org.dspace.storage.rdbms.DatabaseManager @ Exception initializing DB
        pool
[java] java.lang.ClassNotFoundException: org.postgresql.Driver
[java]     at java.net.URLClassLoader$1.run(URLClassLoader.java:198)
[java]     at java.security.AccessController.doPrivileged(Native
       Method)
[java]     at
       java.net.URLClassLoader.findClass(URLClassLoader.java:186)
```

This means that the PostgreSQL JDBC driver is not present in *[dspace]/lib*. See above.

3 Upgrading DSpace

These instructions are valid for any of the following upgrade paths:

- *Upgrading ANY prior version (1.x.x, 3.x, 4.x, 5.x, 6.x or 7.x) of DSpace to DSpace 7.x (latest version)*

For more information about new features or major changes in previous releases of DSpace, please refer to following:

- Releases[155] - Provides links to release notes for all prior releases of DSpace
- Version History(see page 696) - Provides detailed listing of all changes in all prior releases of DSpace

Upgrading database structure/data is now automated!

The underlying DSpace database structure changes and data migrations are now AUTOMATED (using FlywayDB[156]). This means that you no longer need to manually run SQL scripts. Instead, the first time you run DSpace, it will auto-update your database structure (as needed) and migrate all your data to be compatible with the installed version of DSpace. This allows you to concentrate your upgrade efforts on customizing your site without having to worry about migrating your data!

For example, if you were running DSpace 1.4, and you wish to upgrade to DSpace 5, you can follow the simplified instructions below. As soon as you point your DSpace 5 installation against the older DSpace 1.4-compatible database, your database tables (and data) will automatically be migrated to be compatible with DSpace 5.

See below for a specific note on troubleshooting "ignored" migrations (a rare circumstance, but known to happen if you upgrade from DSpace 5 to a later version of DSpace).

Please refrain from customizing the DSpace database tables. It will complicate your next upgrade!

With the addition of our automated database upgrades, *we highly recommend AGAINST customizing the DSpace database tables/structure or backporting any features that change the DSpace tables/structure*. Doing so will often cause the automated database upgrade process to fail (and therefore will complicate your next upgrade).

If you must add features requiring new database tables/structure, we recommend creating new tables (instead of modifying existing ones), as that is usually much less disruptive to our automated database upgrade.

155 https://wiki.lyrasis.org/display/DSPACE/Releases
156 https://flywaydb.org/

Test Your Upgrade Process

In order to minimize downtime, it is always recommended to first perform a DSpace upgrade using a Development or Test server. You should note any problems you may have encountered (and also how to resolve them) before attempting to upgrade your Production server. It also gives you a chance to "practice" at the upgrade. Practice makes perfect, and minimizes problems and downtime. Additionally, if you are using a version control system, such as subversion or git, to manage your locally developed features or modifications, then you can do all of your upgrades in your local version control system on your Development server and commit the changes. That way your Production server can just checkout your well tested and upgraded code.

In the notes below `[dspace]` refers to the install directory for your existing DSpace installation, and `[dspace-source]` to the source directory for DSpace. Whenever you see these path references, be sure to replace them with the actual path names on your local system.

3.1 Release Notes / Significant Changes

DSpace 7.0 features some significant changes which you may wish to be aware of before beginning your upgrade:

- **XMLUI and JSPUI are no longer supported or distributed with DSpace.** All users should install and utilize the new Angular User Interface. See the "Installing the Frontend (User Interface)" instructions in Installing DSpace(see page 53)
- **The old REST API ("rest" webapp from DSpace v4.x-6.x) is deprecated and will be removed in v8.x.** The new REST API(see page 502) (provided in the "server" webapp) replaces all functionality available in the older REST API. If you have tools that rely on the old REST API, you can still (optionally) build & deploy it alongside the "server" webapp via the "-Pdspace-rest" Maven flag. See REST API v6 (deprecated)(see page 506)
- **Solr must be installed separately** due to changes in the packaging of recent Solr releases. The indexes have been reconfigured and must be rebuilt. See below.
- **GeoIP location database must be installed separately** due to changes in Maxmind's terms and conditions. MaxMind has changed the terms and procedure[157] for obtaining and using its GeoLite2 location database. Consequently, DSpace no longer automatically downloads the database during installation or update, and the DSpace-specific database update tool has been removed. If you wish to (continue to) record client location data in SOLR Statistics(see page 338), you will need to make new arrangements. See below.
- **The Submission Form configuration has changed.** The "item-submission.xml" file has changed its structure, and the "input-forms.xml" has been replaced by a "submission-forms.xml". See Submission User Interface(see page 260)

157 https://blog.maxmind.com/2019/12/18/significant-changes-to-accessing-and-using-geolite2-databases/

- **The traditional, 3-step Workflow system has been removed in favor of the** Configurable Workflow System(see page 250)**.** For most users, you should see no effect or difference. The default setup for this Configurable Workflow System is identical to the traditional, 3-step workflow ("Approve/Reject", "Approve/Reject/Edit Metadata", "Edit Metadata")
- **The old BTE import framework in favor of** Live Import Framework(see page 274) (features of BTE have been ported to Live Import)
- **ElasticSearch Usage Statistics have been removed.** Please use SOLR Statistics(see page 338) or DSpace Google Analytics Statistics(see page 363).
- Configuration(see page 552) **has been upgraded to Apache Commons Configuration version 2.** For most users, you should see no effect or difference. *No DSpace configuration files were modified during this upgrade and no configurations or settings were renamed or changed.* However, if you locally modified or customized the `[dspace]/config/config-definition.xml` (DSpace's Apache Commons Configuration settings), you will need to ensure those modifications are compatible with Apache Commons Configuration version 2. See the Apache Commons Configuration's configuration definition file reference[158] for more details.
- **Handle server has been updated to v9.3.** Most users will see no effect or difference, however a minor change to the Handle Server configuration is necessary, see below.
- **Many backend prerequisites have been upgraded to avoid "end of life" versions.** Therefore, pay very close attention to the required prerequisites listed below.
- **A large number of old/obsolete configurations were removed.** "7.0 Configurations Removed" section in the Release Notes(see page 21).

3.2 Upgrading the Backend (Server API)

3.2.1 Backup your DSpace Backend

Before you start your upgrade, it is strongly recommended that you create a backup of your DSpace content. Backups are easy to recover from; a botched install/upgrade is very difficult if not impossible to recover from. The DSpace specific things to backup are: configs, source code modifications, database, and assetstore. On your server that runs DSpace, you might additionally consider checking on your cron/scheduled tasks, servlet container, and database.

Make a complete backup of your system, including:

- Database: Make a snapshot/dump of the database. For the PostgreSQL database use Postgres' pg_dump[159] command. For example:

```
pg_dump -U [database-user] -f [backup-file-location] [database-name]
```

- Assetstore: Backup the directory (`[dspace]/assetstore` by default, and any other assetstores configured in `[dspace]/config/spring/api/bitstore.xml`)
- Configuration: Backup the entire directory content of `[dspace]/config`.
- Customizations: If you have custom code, such as themes, modifications, or custom scripts, you will want to back them up to a safe location.
- Statistics data: what to back up depends on what you were using before: the options are the default SOLR Statistics(see page 338), or the legacy statistics. Legacy stats utilizes the dspace.log files, while SOLR Statistics stores data in `[dspace]/solr/statistics`. A simple copy of the logs or the Solr core directory tree

158 https://commons.apache.org/proper/commons-configuration/userguide/howto_combinedbuilder.html#Configuration_definition_file_reference
159 http://www.postgresql.org/docs/8.4/static/app-pgdump.html

should give you a point of recovery, should something go wrong in the update process. We can't stress this enough: your users depend on these statistics more than you realize. You need a backup.
- Authority data: stored in `[dspace]/solr/authority`. As with the statistics data, making a copy of the directory tree should enable recovery from errors.

3.2.2 Update Backend Prerequisite Software

DSpace 7.x requires the following versions of prerequisite software:

- **Updated:** Java 11 (Oracle or OpenJDK)
- **Updated:** Apache Maven 3.3.x or above
- **Updated:** Apache Ant 1.10.x or above
- Database
  - **Updated:** PostgreSQL 11 (with pgcrypto installed), OR
  - Oracle 10g or above
- **Updated:** Tomcat 9.x
- **New:** Solr 8.x. *See step #11 below.*
- **New:** (optional) IP to City Database for location-based Statistics (either MaxMind's GeoLite City Database[160] or DB-IP's City Lite Database[161]). *See step #13 below.*

Refer to the Backend Requirements section of "Installing DSpace" (see page 53) for more details around configuring and installing these prerequisites.

3.2.3 Upgrading the Backend Steps

1. **Ensure that your database is compatible:** Starting with DSpace 6.x, there are new database requirements for DSpace (refer to the Backend Requirements section of "Installing DSpace" (see page 53) for full details).
   a. *PostgreSQL databases*: PostgreSQL 9.4 or above is required and the "pgcrypto" extension must be installed.
      i. Notes on installing pgcrypto
         1. On most Linux operating systems (Ubuntu, Debian, RedHat), this extension is provided in the "postgresql-contrib" package in your package manager. So, ensure you've installed "postgresql-contrib".
         2. On Windows, this extension should be provided automatically by the installer (check your "[PostgreSQL]/share/extension" folder for files starting with "pgcrypto")
      ii. Enabling pgcrypto on your DSpace database. (Additional options/notes in the Installation Documentation(see page 53))

```
# Login to your "dspace" database as a superuser
psql --username=postgres dspace
# Enable the pgcrypto extension on this database
CREATE EXTENSION pgcrypto;
```

   b. *Oracle databases:* Oracle database have no additional requirements at this time.
2. **From your old version of DSpace, dump your `authority` and `statistics` Solr cores.** (Only necessary when upgrading from DSpace 6 or older & you want to keep both your authority records and/or SOLR Statistics(see page 338))

160 https://dev.maxmind.com/geoip/geoip2/geolite2/
161 https://db-ip.com/db/download/ip-to-city-lite

```
[dspace]/bin/dspace solr-export-statistics -i authority
[dspace]/bin/dspace solr-export-statistics -i statistics
```

The dumps will be written to the directory `[dspace]/solr-export`. *This may take a long time and require quite a lot of storage.* In particular, the statistics core is likely to be huge, perhaps double the size of the content of `solr/statistics/data`. You should ensure that you have sufficient free storage.

This is not the same as the disaster-recovery backup that was done above. These dumps will be reloaded into new, reconfigured cores later(see page 84).

If you are **sharding** your statistics data, you will need to dump each shard separately. The index names for prior years will be `statistics-YYYY` (for example: `statistics-2017 statistics-2018` etc.) The current year's statistics shard is named `statistics` and you should dump that one too.

3. **Download** the latest DSpace release[162] from the DSpace GitHub Repository. You can choose to either download the zip or tar.gz file provided by GitHub, or you can use "git" to checkout the appropriate tag (e.g. `dspace-7.0`) or branch.
    a. Unpack it using "unzip" or "gunzip". If you have an older version of DSpace installed on this same server, you may wish to unpack it to a different location than that release. This will ensure no files are accidentally overwritten during the unpacking process, and allow you to compare configs side by side.
    b. For ease of reference, we will refer to the location of this unzipped version of the DSpace release as *[dspace-source]* in the remainder of these instructions.
4. **Replace your old build.properties file with a local.cfg** *(ONLY REQUIRED if upgrading from DSpace 5 or previous):* As of DSpace 6.0, the `build.properties` configuration file has been replaced by an enhanced `local.cfg` configuration file. Therefore, any old `build.properties` file (or similar `[dspace-source]/*.properties` files) WILL BE IGNORED. Instead, you should create a new `local.cfg` file, based on the provided `[dspace-source]/dspace/config/local.cfg.EXAMPLE` and use it to specify all of your locally customized DSpace configurations. This new `local.cfg` can be used to override ANY setting in any other configuration file (`dspace.cfg` or `modules/*.cfg`). To override a default setting, simply copy the configuration into your `local.cfg` and change its value(s). For much more information on the features of local.cfg, see the Configuration Reference(see page 552) documentation and the local.cfg Configuration File(see page 560) section on that page.

```
cd [dspace-source]
cp dspace/config/local.cfg.EXAMPLE local.cfg

# Then edit the local.cfg, specifying (at a minimum) your basic DSpace configuration settings.
# Optionally, you may copy any settings from other *.cfg configuration files into your local.cfg to
override them.
# After building DSpace, this local.cfg will be copied to [dspace]/config/local.cfg, where it will
also be used at runtime.
```

5. **Build DSpace Backend.** Run the following commands to compile DSpace :

```
cd [dspace-source]
mvn -U clean package
```

The above command will re-compile the DSpace source code and build its "installer". You will find the result in `[dspace-source]/dspace/target/dspace-installer`

162 https://github.com/DSpace/DSpace/releases

> **Defaults to PostgreSQL settings**
>
> Without any extra arguments, the DSpace installation package is initialized for PostgreSQL. If you use Oracle instead, you should build the DSpace installation package as follows:
> `mvn -Ddb.name`[163]`=oracle -U clean package`

6. **Stop Tomcat (or servlet container).** Take down your servlet container.
    a. For Tomcat, use the `$CATALINA_HOME/shutdown.sh` script. (Many Unix-based installations will have a startup/shutdown script in the `/etc/init.d` or `/etc/rc.d` directories.)
7. **Update DSpace Installation.** Update the DSpace installation directory with the new code and libraries. Issue the following commands:

```
cd [dspace-source]/dspace/target/dspace-installer
ant update
```

8. **Update your DSpace Configurations.** Depending on the version of DSpace you are upgrading from, not all steps are required.
    a. **If you are upgrading from DSpace 5.x or below.** *If you are upgrading from DSpace 6.x you can skip these steps and move to the next bullet.*
        i. **Search/Browse requires Discovery:** As of DSpace 6, only Discovery(see page 386) (Apache Solr) is supported for search/browse. Support for Legacy Search (using Apache Lucene) and Legacy Browse (using database tables) has been removed, along with all their configurations.
        ii. **XPDF media filtering no longer exists:** XPDF media filtering, deprecated in DSpace 5, has been removed. If you used this, you will need to reconfigure using the remaining alternatives(see page 469) (e.g. PDF Text Extractor and/or ImageMagick PDF Thumbnail Generator)
    b. **If you are upgrading from DSpace 6.x or below.** *All administrators will need to perform these steps.*
        i. **Review your customized configurations (recommended to be in local.cfg):** As mentioned above, we recommend any local configuration changes be placed in a local.cfg Configuration File(see page 560). With any major upgrade some configurations may have changed. Therefore, it is recommended to review all configuration changes that exist in the `config` directory, and its subdirectories, concentrating on configurations your previously customized in your local.cfg. See also the Configuration Reference(see page 552).
        ii. **Remove obsolete configurations.** With the removal of the JSPUI and XMLUI, a large number of server-side (backend) configurations were made obsolete and were therefore removed between the 6.x and 7.0 release. A full list can be found in the Release Notes(see page 24).
        iii. **Migrate or recreate your Submission configuration.** As of DSpace 7, the submission configuration has changed. The format of the "item-submission.xml" file has been updated, and the older "input-forms.xml" has been replaced by a new "submission-forms.xml". You can choose to either start fresh with the new v7 configuration files, or you can use the steps below to migrate your old configurations into the new format. See the Submission User Interface(see page 260) for more information
            1. First, create a temporary folder to copy your old v6 configurations into

```
# Example of creating a [dspace]/config/temp folder for this migration
# You must replace [dspace] with the full path of your DSpace 7 installation.
cd [dspace]/config
mkdir temp
```

163 http://ddb.name/

2. Copy your old (v5 or v6) "item-submission.xml" and "input-forms.xml" into that temporary folder
3. Run the command-line migration script to migrate them to v7 configuration files

```
# This example uses [dspace] as a placeholder for all paths.
# Replace it with either the absolute or relative path of these files
[dspace]/bin/dspace submission-forms-migrate -s [dspace]/config/temp/item-
submission.xml -f [dspace]/config/temp/input-forms.xml
```

4. The result will be two files. These are valid v7 configurations based on your original submission configuration files.
    a. `[dspace]/config/item-submission.xml.migrated`
    b. `[dspace]/config/submission-forms.xml.migrated`
5. These "*.migrated" files have *no inline comments*, so you may want to edit them further before installing them (by removing the ".migrated" suffix). Alternatively, you may choose to copy sections of the *.migrated files into the default configurations in the `[dspace]/config/` folder, therefore retaining the inline comments in those default files.

iv. **City IP Database file for Solr Statistics has been renamed.** The old `[dspace]/config/GeoLiteCity.dat` file is no longer maintained by its provider. You can delete it. The new file is named `GeoLite2-City.mmdb` by default. If you have configured a different name and/or location for this file, you should check the setting of `usage-statistics.dbfile` in `[dspace]/config/modules/usage-statistics.cfg` (and perhaps move your custom setting to `local.cfg`).

v. **tm-extractors media filtering (WordFilter) no longer exists:** the PoiWordFilter plugin now fulfills this function. If you still have WordFilter configured, remove from `dspace.cfg` and/or `local.cfg` all lines referencing `org.dspace.app.mediafilter.WordFilter` and uncomment all lines referencing `org.dspace.app.mediafilter.PoiWordFilter`.

vi. **Re-configure Solr URLs:** change the value of `solr.server` to point at your new Solr external service. It will probably become something like `solr.server = https://${dspace.hostname}:8983/solr`. Also review the values of
    1. `discovery.search.server`
    2. `oai.solr.url`
    3. `solr.authority.server`
    4. `solr.statistics.server`

vii. **Sitemaps are now automatically generated/updated:** A new `sitemap.cron` setting exists in the dspace.cfg which controls when Sitemaps are generated. By default they are enabled to update once per day, for optimal SEO. See Search Engine Optimization(see page 485) docs for more detail
    1. Because of this change, if you had a system cron job which ran "`./dspace generate-sitemaps`", this system cron job can be removed in favor of the new `sitemap.cron` setting.

9. **Upgrade your database** (*optional, but recommended for major upgrades*). As of DSpace 5 (and above), the DSpace code will automatically upgrade your database (*from any prior version of DSpace*). By default, this database upgrade occurs automatically when you restart Tomcat (or your servlet container). However, if you have a large repository or are upgrading across multiple versions of DSpace at once, you may wish to manually perform the upgrade (as it could take some time, anywhere from 5-15 minutes for large sites).
    a. First, you can optionally verify whether DSpace correctly detects the version of your DSpace database. It is **very important** that the DSpace version is detected correctly before you attempt the migration:

```
[dspace]/bin/dspace database info
# Look for a line at the bottom that says something like:
# "Your database looks to be compatible with DSpace version ___"
```

b. In some rare scenarios, if your database's "sequences" are outdated, inconsistent or incorrect, a database migration error may occur (in your DSpace logs). While this is seemingly a rare occurrence, you may choose to run the "update-sequences" command PRIOR to upgrading your database. If your database sequences are inconsistent or incorrect, this "update-sequences" command will auto-correct them (otherwise, it will do nothing).

```
# General PostgreSQL example
psql -U [database-user] -f [dspace]/etc/postgres/update-sequences.sql [database-name]

# Example for a PostgreSQL database named "dspace", and a user account named "dspace"
# psql -U dspace -f [dspace]/etc/postgres/update-sequences.sql dspace
```

c. Then, you can upgrade your DSpace database to the latest version of DSpace. (NOTE: check the DSpace log, `[dspace]/log/dspace.log.[date]`, for any output from this command)

```
[dspace]/bin/dspace database migrate
```

If you are upgrading from DSpace 6 or earlier, there are database changes which were previously optional but now are mandatory (specifically Configurable Workflow(see page 250) database changes). Instead of (or after) the above command:

```
[dspace]/bin/dspace database migrate ignored
```

to apply these changes.

d. *If the database upgrade process fails or throws errors*, then you likely have manually customized your database structure (and/or backported later DSpace features to an older version of DSpace). In this scenario, you may need to do some manual migrations before the automatic migrations will succeed. The general process would be something like this:
   i. Revert back to your current DSpace database
   ii. Manually upgrade just your database **past** the failing migration. For example, if you are current using DSpace 1.5 and the "V1.6" migration is failing, you may need to first manually upgrade your database to 1.6 compatibility. This may involve either referencing the upgrade documentation for that older version of DSpace, or running the appropriate SQL script from under `[dspace-src]/dspace-api/src/main/resources/org/dspace/storage/rdbms/sqlmigration/`)
   iii. Then, re-run the migration process from that point forward (i.e. re-run `./dspace database migrate`)

e. More information on the "database" command can be found in Database Utilities[164] documentation.

164 https://wiki.lyrasis.org/display/DSDOC5x/Database+Utilities

> ⚠ **By default, your site will be automatically reindexed after a database upgrade**
>
> If any database migrations are run (even during minor release upgrades), then by default DSpace will automatically reindex all content in your site. This process is run automatically in order to ensure that any database-level changes are also immediately updated within the search/browse interfaces. See the notes below under "**Restart Tomcat (servlet container)**" for more information.
>
> However, you may choose to **skip automatic reindexing.** Some sites choose to run the reindex process manually in order to better control when/how it runs.
>
> To disable automatic reindexing, set
>
> `discovery.autoReindex = false` in `config/local.cfg` or `config/modules/discovery.cfg`.
>
> As you have disabled automatic reindexing, make sure to manually reindex your site by running `[dspace]/bin/dspace discovery -b` *(This must be run after restarting Tomcat)*
>
> *WARNING: It is not recommended to skip automatic reindexing, unless you will **manually reindex** at a later time, or have verified that a reindex is not necessary. Forgetting to reindex your site after an upgrade may result in unexpected errors or instabilties.*

> ⚠ **Sites with Oracle database backends (and Configurable Workflow enabled) may need to run a "repair" on your database.**
>
> In version 6.3, we fixed an Oracle migration issue related to Configurable (XML) Workflow(see page 250). See DS-3788[165].
>
> If you are upgrading an Oracle-based site to 6.3 from 6.0, 6.1 or 6.2 AND had Configurable Workflow already enabled, then you will need to manually "repair" your database to align it with the latest schema. *This does not affect PostgreSQL-based backends or any sites that are upgrading from 5.x or below.*
>
> Simply run the following to repair your Oracle database:
>
> ```
> [dspace]/bin/dspace database repair
> ```

10. **Deploy Server web application:** The DSpace backend consists of a single "server" webapp (in `[dspace]/webapps/server`). You need to deploy this webapp into your Servlet Container (e.g. Tomcat). Generally, there are two options (or techniques) which you could use...either configure Tomcat to find the DSpace "server" webapp, or copy the "server" webapp into Tomcat's own webapps folder. For more information & example commands, see the Installation Guide(see page 53)
    a. Optionally, you may also install the deprecated DSpace 6.x REST API web application ("rest" webapp). If you previously used the DSpace 6.x REST API(see page 506), for backwards compatibility the old, deprecated "rest" webapp is still available to install (in `[dspace]/webapps/rest`). It is NOT used by the DSpace UI/frontend. So, most users should skip this step.
11. **Install new Solr cores and rebuild your indexes.** (Only necessary if upgrading from 6.x or below)
    a. Copy the new, empty Solr cores to your new Solr instance.

```
cp -R [dspace]/solr/* [solr]/server/solr/configsets
chown -R solr:solr [solr]/server/solr/configsets
```

165 https://jira.duraspace.org/browse/DS-3788

b. Start Solr, or restart it if it is running, so that these new cores are loaded.

```
[solr]/bin/solr restart
```

c. Load `authority` and `statistics` from the dumps(see page 78) that you made earlier (not the disaster-recovery backup).

```
[dspace]/bin/dspace solr-import-statistics -i authority
[dspace]/bin/dspace solr-import-statistics -i statistics
```

This could take quite some time.

If you had sharded your statistics, you will need to load the dump of each shard separately. As when dumping, the index names will be `... statistics-2017 statistics-2018 statistics`.

d. For Statistics shards only, upgrade legacy DSpace Object Identifiers (pre-6.4 statistics) to UUID Identifiers.

```
[dspace]/bin/dspace solr-upgrade-statistics-6x -i statistics
```

Again If you had sharded your statistics, you will need to run this for each shard separately. See also SOLR Statistics Maintenance(see page 356)

e. Rebuild the `oai` and `search` cores.

```
[dspace]/bin/dspace oai import
[dspace]/bin/dspace index-discovery -b
```

If you have a great deal of content, this could take a long time.

12. **Update Handle Server Configuration.** (Only necessary if upgrading from 6.x or below) Because we've updated to Handle Server v9, if you are using the built-in Handle server (most installations do), you'll need to add the follow to the end of the *server_config* section of your `[dspace]/handle-server/config.dct` file (the only new line is the "enable_txn_queue" line)

```
"case_sensitive" = "no"
"storage_type" = "CUSTOM"
"storage_class" = "org.dspace.handle.HandlePlugin"
"enable_txn_queue" = "no"
```

a. Alternatively, you could re-run the `./dspace make-handle-config` script, which is in charge of updating this `config.dct` file.

13. **(Optional) Set up IP to City database for location-based statistics.** If you wish to (continue to) record the geographic origin of client activity, you will need to install (and regularly update) one of the following:
    - Either, a copy of MaxMind's GeoLite City database[166] (in MMDB format)
        - NOTE: Installing MaxMind GeoLite2 is *free*. However, you **must** sign up for a (free) MaxMind account in order to obtain a license key to use the GeoLite2 database.
        - You may download GeoLite2 directly from MaxMind, or many Linux distributions provide the `geoipupdate` tool directly via their package manager. You will still need to configure your license key prior to usage.

166 https://dev.maxmind.com/geoip/geoip2/geolite2/

        - Once the "GeoLite2-City.mmdb" database file is installed on your system, you will need to configure its location as the value of `usage-statistics.dbfile` in your `local.cfg` configuration file .
        - You can discard any old GeoLiteCity.dat database(s) found in the `config/` directory (if they exist).
        - See the "Managing the City Database File" section of SOLR Statistics(see page 338) for more information about using a City Database with DSpace.
    - Or, you can alternatively use/install DB-IP's City Lite database[167] (in MMDB format)
        - This database is also free to use, but does **not** require an account to download.
        - Once the "dbip-city-lite.mmdb" database file is installed on your system, you will need to configure its location as the value of `usage-statistics.dbfile` in your `local.cfg` configuration file .
        - See the "Managing the City Database File" section of SOLR Statistics(see page 338) for more information about using a City Database with DSpace.
14. **Restart Tomcat (servlet container).** Now restart your servlet container (Tomcat/Jetty/Resin) and test out the upgrade.
    a. **Upgrade of database:** If you didn't manually upgrade your database in the previous step, then your database will be automatically upgraded to the latest version. This may take some time (seconds to minutes), depending on the size of your repository, etc. Check the DSpace log (`[dspace]/log/dspace.log.[date]`) for information on its status.
    b. **Reindexing of all content for search/browse:** If your database was just upgraded (either manually or automatically), all the content in your DSpace will be automatically re-indexed for searching/browsing. As the process can take some time (minutes to hours, depending on the size of your repository), it is performed in the background; meanwhile, DSpace can be used as the index is gradually filled. *But, keep in mind that not all content will be visible until the indexing process is completed.* Again, check the DSpace log (`[dspace]/log/dspace.log.[date]`) for information on its status.
        i. *If you wish to skip automatic reindexing, please see the Note above under the "Upgrade your Database" step.*
15. **Check your cron / Task Scheduler jobs.** In recent versions of DSpace, some of the scripts names have changed.
    a. Check the Scheduled Tasks via Cron(see page 481) documentation for details. If you have been using the `dspace stats-util --optimize` tool, it is no longer recommended and you should stop.
    b. WINDOWS NOTE: If you are running the Handle Server on a Windows machine, a new `[dspace]/bin/start-handle-server.bat` script is available to more easily startup your Handle Server.
16. **Install the new User Interface (see below)**

3.3 Upgrading the Frontend (User Interface)

1. **Install the new User Interface per the** Installing DSpace(see page 53) **guide.** The JSPUI and XMLUI are no longer supported and cannot work with the DSpace 7 backend. You will need to install the new (Angular.io) User Interface.
    a. JSPUI or XMLUI based themes cannot be migrated. That said, since the new Angular UI also uses Bootstrap, you may be able to borrow some basic CSS from your old themes. But any HTML-level changes will need to be reimplemented in the new UI.

167 https://db-ip.com/db/download/ip-to-city-lite

3.4 Troubleshooting Upgrade Issues

3.4.1 "Ignored" Flyway Migrations

In very rare instances, a Flyway database migration will be "ignored." One known instance of this is documented in DS-3407. If you are upgrading from DSpace 5.x to a later version of DSpace, the migration put in place to address DS-2818 will be "ignored" because it is not necessary. There is a special command you can run which will un-flag this migration as "ignored."

```
dspace database migrate ignored
```

warning: dangerous command: BACK UP YOUR DATABASE!

The `dspace database migrate ignored` command can be dangerous: it will attempt to re-run *all* ignored migrations. In the case outlined above, this is safe. However, under other circumstances, re-running ignored migrations can lead to unpredictable results. To be absolutely safe, be sure you have a current backup of your database.

The presence of `ignored` migrations can indicate a problem in the database. It's best not to use this command unless instructed to.

3.4.2 Manually updating the Metadata Registries

The database migration (./dspace database migratte) should *automatically trigger* your metadata/file registries to be updated (based on the config files in `[dspace]/config/registries/`). However, if this update was NOT triggered, you can also manually run these registry updates (they will not harm existing registry contents) as follows:

```
[dspace]/bin/dspace registry-loader -metadata [dspace]/config/registries/dcterms-types.xml
[dspace]/bin/dspace registry-loader -metadata [dspace]/config/registries/dublin-core-types.xml
[dspace]/bin/dspace registry-loader -metadata [dspace]/config/registries/eperson-types.xml
[dspace]/bin/dspace registry-loader -metadata [dspace]/config/registries/local-types.xml
[dspace]/bin/dspace registry-loader -metadata [dspace]/config/registries/sword-metadata.xml
[dspace]/bin/dspace registry-loader -metadata [dspace]/config/registries/workflow-types.xml
```

4 Using DSpace

This page offers access to all aspects of the documentation relevant to using DSpace after it has been properly installed or upgraded. These pages assume that DSpace is functioning properly. Please refer to the section on System Administration(see page 410) if you are looking for information on diagnosing DSpace issues and measures you can take to restore your DSpace to a state in which it functions properly.

4.1 Authentication and Authorization

- Authentication Plugins(see page 87)
- Embargo(see page 113)
- Managing User Accounts(see page 121)
- Request a Copy(see page 126)

4.1.1 Authentication Plugins

- Stackable Authentication Method(s)(see page 87)
  - Authentication by Password(see page 89)
    - Enabling Authentication by Password(see page 89)
    - Configuring Authentication by Password(see page 89)
  - Shibboleth Authentication(see page 90)
    - Enabling Shibboleth Authentication(see page 90)
    - Configuring Shibboleth Authentication(see page 91)
      - Apache "mod_shib" Configuration (required)(see page 91)
      - Sample shibboleth2.xml Configuration(see page 94)
      - Sample attribute-map.xml Configuration (for samltest.id)(see page 95)
      - DSpace Shibboleth Configuration Options(see page 96)
  - LDAP Authentication(see page 101)
    - Introduction to LDAP specific terminology(see page 101)
    - Enabling LDAP Authentication(see page 102)
    - Configuring LDAP Authentication(see page 102)
    - Debugging LDAP connection and configuration(see page 108)
    - Enabling Hierarchical LDAP Authentication(see page 109)
    - Configuring Hierarchical LDAP Authentication(see page 110)
  - IP Authentication(see page 111)
    - Enabling IP Authentication(see page 111)
    - Configuring IP Authentication(see page 111)
  - X.509 Certificate Authentication(see page 112)
    - Enabling X.509 Certificate Authentication(see page 112)
    - Configuring X.509 Certificate Authentication(see page 112)
  - Example of a Custom Authentication Method(see page 113)

4.1.1.1 Stackable Authentication Method(s)

Since many institutions and organizations have existing authentication systems, DSpace has been designed to allow these to be easily integrated into an existing authentication infrastructure. It keeps a series, or "stack", of *authentication methods*, so each one can be tried in turn. This makes it easy to add new authentication methods or rearrange the order without changing any existing code. You can also share authentication code with other sites.

Configuration File:	**[dspace]/config/modules/authentication.cfg**
Property:	plugin.sequence.org.dspace.authenticate.AuthenticationMethod
Example Value:	plugin.sequence.org.dspace.authenticate.AuthenticationMethod = org.dspace.authenticate.PasswordAuthentication

The configuration property `plugin.sequence.org.dspace.authenticate.AuthenticationMethod` defines the authentication stack. It is a comma-separated list of class names. Each of these classes implements a different `authentication method`, or way of determining the identity of the user. They are invoked in the order specified until one succeeds.

Existing Authentication Methods include

- Authentication by Password(see page 89) (class: `org.dspace.authenticate.PasswordAuthentication`) (DEFAULT)
- Shibboleth Authentication(see page 90) (class: `org.dspace.authenticate.ShibAuthentication`)
- LDAP Authentication(see page 101) (class: `org.dspace.authenticate.LDAPAuthentication`)
- IP Address based Authentication(see page 111) (class: `org.dspace.authenticate.IPAuthentication`)
- X.509 Certificate Authentication(see page 112) (class: `org.dspace.authenticate.X509Authentication`)

An authentication method is a class that implements the interface `org.dspace.authenticate.AuthenticationMethod`. It `authenticates` a user by evaluating the *credentials* (e.g. username and password) he or she presents and checking that they are valid.

The basic authentication procedure in the DSpace Web UI is this:

1. A request is received from an end-user's browser that, if fulfilled, would lead to an action requiring authorization taking place.
2. If the end-user is already authenticated:
    - If the end-user is allowed to perform the action, the action proceeds
    - If the end-user is NOT allowed to perform the action, an authorization error is displayed.
    - If the end-user is NOT authenticated, i.e. is accessing DSpace anonymously:
3. The parameters etc. of the request are stored.
4. The Web UI's `startAuthentication` method is invoked.
5. First it tries all the authentication methods which do `implicit` authentication (i.e. they work with just the information already in the Web request, such as an X.509 client certificate). If one of these succeeds, it proceeds from Step 2 above.
6. If none of the implicit methods succeed, the UI responds by putting up a "login" page to collect credentials for one of the `explicit` authentication methods in the stack. The servlet processing that page then gives the proffered credentials to each authentication method in turn until one succeeds, at which point it retries the original operation from Step 2 above.
   Please see the source files `AuthenticationManager.java` and `AuthenticationMethod.java` for more details about this mechanism.

Authentication by Password

Enabling Authentication by Password

By default, this authentication method is enabled in DSpace.

However, to enable Authentication by Password, you must ensure the `org.dspace.authenticate.PasswordAuthentication` class is listed as one of the AuthenticationMethods in the following configuration:

Configuration File:	**`[dspace]/config/modules/authentication.cfg`**
Property:	`plugin.sequence.org.dspace.authenticate.AuthenticationMethod`
Example Value:	`plugin.sequence.org.dspace.authenticate.AuthenticationMethod = org.dspace.authenticate.PasswordAuthentication`

Configuring Authentication by Password

The default method `org.dspace.authenticate.PasswordAuthentication` has the following properties:

- Use of inbuilt e-mail address/password-based log-in. This is achieved by forwarding a request that is attempting an action requiring authorization to the password log-in servlet, `/password-login`. The password log-in servlet (`org.dspace.app.webui.servlet.PasswordServlet`) contains code that will resume the original request if authentication is successful, as per step 3. described above.
- Users can register themselves (i.e. add themselves as e-people without needing approval from the administrators), and can set their own passwords when they do this
- Users are not members of any special (dynamic) e-person groups
- You can restrict the domains from which new users are able to register. To enable this feature, uncomment the following line from dspace.cfg: `authentication.password.domain.valid = example.com` Example options might be '`@example.com`' to restrict registration to users with addresses ending in @example.com, or '`@example.com, .ac.uk`' to restrict registration to users with addresses ending in @example.com or with addresses in the .ac.uk domain.

A full list of all available Password Authentication Configurations:

Configuration File:	**`[dspace]/config/modules/authentication-password.cfg`**
Property:	`authentication-password.domain.valid`
Example Value:	`authentication-password.domain.value = @mit.edu, .ac.uk`

Informational Note:	This option allows you to limit self-registration to email addresses ending in a particular domain value. The above example would limit self-registration to individuals with "@mit.edu" email addresses and all ".ac.uk" email addresses.
Property:	`authentication-password.login.specialgroup`
Example Value:	`authentication-password.login.specialgroup = My DSpace Group`
Informational Note:	This option allows you to automatically add all password authenticated user sessions to a specific DSpace Group (the group must exist in DSpace) for the remainder of their logged in session.
Property:	`authentication-password.digestAlgorithm`
Example Value:	`authentication-password.digestAlgorithm = SHA-512`
Informational Note:	This option specifies the hashing algorithm to be used in converting plain-text passwords to more secure password digests. The example value is the default. You may select any digest algorithm available through java.security.MessageDigest on your system. At least MD2, MD5, SHA-1, SHA-256, SHA-384, and SHA-512 should be available, but you may have installed others. Most sites will not need to adjust this.

Shibboleth Authentication

Enabling Shibboleth Authentication

To enable Shibboleth Authentication, you must ensure the `org.dspace.authenticate.ShibAuthentication` class is listed as one of the AuthenticationMethods in the following configuration:

Configuration File:	**`[dspace]/config/modules/authentication.cfg`**
Property:	`plugin.sequence.org.dspace.authenticate.AuthenticationMethod`

Example Value:	`plugin.sequence.org.dspace.authenticate.AuthenticationMethod = org.dspace.authenticate.ShibAuthentication` (*NOTE: This setting may be repeated to support multiple AuthenticationMethods*)

Configuring Shibboleth Authentication

Shibboleth is a distributed authentication system for securely authenticating users and passing attributes about the user from one or more identity providers. In the Shibboleth terminology DSpace is a Service Provider which receives authentication information and then based upon that provides a service to the user. To use Shibboleth, DSpace *requires* that you use Apache installed with the mod_shib module acting as a proxy for all HTTP requests for your servlet container (typically Tomcat). DSpace will receive authentication information from the mod_shib module through HTTP headers.

Before DSpace will work with Shibboleth, you **must** have the following:

1. An Apache web server with the "mod_shib" module installed. As mentioned, this mod_shib module acts as a proxy for all HTTP requests for your servlet container (typically Tomcat). Any requests to DSpace that require authentication via Shibboleth should be redirected to 'shibd' (the shibboleth daemon) by this "mod_shib" module. Details on installing/configuring mod_shib in Apache are available at: https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPApacheConfig We also have a sample Apache + mod_shib configuration provided below.
2. An external Shibboleth IdP (Identity Provider). Using mod_shib, DSpace will only act as a Shibboleth SP (Service Provider). The actual Shibboleth Authentication & Identity information must be provided by an external IdP. If you are using Shibboleth at your institution already, then there already should be a Shibboleth IdP available. More information about Shibboleth IdPs versus SPs is available at: https://wiki.shibboleth.net/confluence/display/SHIB2/UnderstandingShibboleth

For more information on installing and configuring a Shibboleth Service Provider see: https://wiki.shibboleth.net/confluence/display/SHIB2/Installation

Note about Shibboleth Active vs Lazy Sessions:

When configuring your Shibboleth Service Provider there are two Shibboleth paradigms you may use: Active or Lazy Sessions. Active sessions is where the mod_shib module is configured to product an entire URL space. No one will be able to access that URL without first authenticating with Shibboleth. Using this method you will need to configure shibboleth to protect the URL: "/shibboleth-login". The alternative, Lazy Session does not protect any specific URL. Instead Apache will allow access to any URL, and when the application wants to it may initiate an authenticated session.

The Lazy Session method is preferable for most DSpace installations, as you usually want to provide public access to (most) DSpace content, while restricting access to only particular areas (e.g. administration UI/tools, private Items, etc.). When Active Sessions are enabled your *entire* DSpace site will be access restricted. In other words, when using Active Sessions, Shibboleth will require everyone to first authenticate before they can access any part of your repository (which essentially results in a "dark archive", as anonymous access will not be allowed).

Apache "mod_shib" Configuration (required)
As mentioned above, you must have Apache with the "mod_shib" module installed in order for DSpace to be able to act as a Shibboleth Service Provider (SP). The mod_shib module acts as a proxy for all HTTP requests for your servlet container (typically Tomcat). Any requests to DSpace that require authentication via Shibboleth should be redirected to 'shibd' (the shibboleth daemon) by this "mod_shib" module. Details on installing/configuring mod_shib in Apache are available at: https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPApacheConfig

General information about installing/configuring Shibboleth Service Providers (SPs) can be found at: https://wiki.shibboleth.net/confluence/display/SHIB2/Installation

A few extra notes/hints when configuring mod_shib & Apache:

- In Debian based environments, "mod_shib" tends to be in a package named something like "libapache2-mod-shib2"
- The Shibboleth setting "ShibUseHeaders" is no longer required to be set to "On", as DSpace will correctly utilize attributes instead of headers.
    - When "ShibUseHeaders" is set to "Off" (which is recommended in the mod_shib documentation[168]), proper configuration of Apache to pass attributes to Tomcat (via either mod_jk or mod_proxy) can be a bit tricky, SWITCH has some great documentation[169] on exactly what you need to do. We will eventually paraphrase/summarize this documentation here, but for now, the SWITCH page will have to do.
- When initially setting up Apache & mod_shib, https://samltest.id/ provides a great testing ground for your configurations. This site provides a sample/demo Shibboleth IdP (as well as a sample Shibboleth SP) which you can test against. It acts as a "sandbox" to get your configurations working properly, before you point DSpace at your production Shibboleth IdP.
- You also may wish to review the Shibboleth setup in our "dspace-shibboleth" Docker setup[170]which the development team uses for testing (and it uses https://samltest.id as the IdP). It may provide you with good examples/hints on getting everything setup. However, keep in mind this code has not been tested in Production scenarios.

Below, we have provided a sample Apache configuration. However, as every institution has their own specific Apache setup/configuration, it is highly likely that you will need to tweak this configuration in order to get it working properly. Again, see the official mod_shib documentation for much more detail about each of these settings: https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPApacheConfig These configurations are meant to be added to an Apache <VirtualHost> which acts as a proxy to your Tomcat (or other servlet container) running DSpace. More information on Apache VirtualHost settings can be found at: https://httpd.apache.org/docs/2.2/vhosts/

168 https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPApacheConfig#NativeSPApacheConfig-AuthConfigOptions
169 https://www.switch.ch/de/aai/support/serviceproviders/sp-access-rules.html#javaapplications
170 https://github.com/DSpace/DSpace/tree/main/dspace/src/main/docker/dspace-shibboleth

```
#### SAMPLE MOD_SHIB CONFIGURATION FOR APACHE2 (it may require local modifications based on your Apache
setup) ####
# While this sample VirtualHost is for HTTPS requests (recommended for Shibboleth, obviously),
# you may also need/want to create one for HTTP (*:80)
<VirtualHost *:443>
   ...
   # PLEASE NOTE: We have omitted many Apache settings (ServerName, LogLevel, SSLCertificateFile, etc)
   # which you may need/want to add to your VirtualHost

   # As long as Shibboleth module is installed, enable all Shibboleth/mod_shib related settings
   <IfModule mod_shib>
       # Shibboleth recommends turning on UseCanonicalName
       # See "Prepping Apache" in https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPApacheConfig
       UseCanonicalName On

       # Most DSpace instances will want to use Shibboleth "Lazy Session", which ensures that users
       # can access DSpace without first authenticating via Shibboleth.
       # This section turns on Shibboleth "Lazy Session". Also ensures that once they have authenticated
       # (by accessing /Shibboleth.sso/Login path), then their Shib session is kept alive
       <Location />
         AuthType shibboleth
         ShibRequireSession Off
         require shibboleth
         # If your "shibboleth2.xml" file specifies an <ApplicationOverride> setting for your
         # DSpace Service Provider, then you may need to tell Apache which "id" to redirect Shib requests
to.
         # Just uncomment this and change the value "my-dspace-id" to the associated @id attribute value.
         #ShibRequestSetting applicationId my-dspace-id
       </Location>

       # If a user attempts to access the DSpace shibboleth endpoint, force them to authenticate via Shib.
       <Location "/server/api/authn/shibboleth">
         Order deny,allow
         Allow from all
         AuthType shibboleth
         ShibRequireSession On
         # Please note that setting ShibUseHeaders to "On" is a potential security risk.
         # You may wish to set it to "Off". See the mod_shib docs for details about this setting:
         # https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPApacheConfig#NativeSPApacheConfig-
AuthConfigOptions
         # Here's a good guide to configuring Apache + Tomcat when this setting is "Off":
         # https://www.switch.ch/de/aai/support/serviceproviders/sp-access-rules.html#javaapplications
         ShibUseHeaders On
         Require shibboleth
       </Location>

       # If a user attempts to access the DSpace login endpoint, ensure Shibboleth is supported but other
auth methods can be too.
       <Location "/server/api/authn/login">
          Order deny,allow
          Allow from all
          AuthType shibboleth
          # For DSpace, this is required to be off otherwise the available auth methods will be not visible
          ShibRequireSession Off
          # Please note that setting ShibUseHeaders to "On" is a potential security risk.
```

```
          # You may wish to set it to "Off". See the mod_shib docs for details about this setting:
          # https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPApacheConfig#NativeSPApacheConfig-
AuthConfigOptions
          # Here's a good guide to configuring Apache + Tomcat when this setting is "Off":
          # https://www.switch.ch/de/aai/support/serviceproviders/sp-access-rules.html#javaapplications
          ShibUseHeaders On
       </Location>

       # Ensure /Shibboleth.sso path (in Apache) can be accessed
       # By default it may be inaccessible if your Apache security is tight.
       <Location "/Shibboleth.sso">
         Order deny,allow
         Allow from all
         # Also ensure Shibboleth/mod_shib responds to this path
         SetHandler shib
       </Location>

       # Finally, you may need to ensure requests to /Shibboleth.sso are NOT redirected
       # to Tomcat (as they need to be handled by mod_shib instead).
       # NOTE: THIS SETTING IS LIKELY ONLY NEEDED IF YOU ARE USING mod_proxy TO REDIRECT
       # ALL REQUESTS TO TOMCAT (e.g. ProxyPass /server ajp://localhost:8009/server)
       ProxyPass /Shibboleth.sso !
   </IfModule>

   ...


   # You will likely need Proxy settings to ensure Apache is proxying requests to Tomcat for the DSpace
REST API
   # The below is just an example of proxying for REST API only. It requires installing & enabling
"mod_proxy" and "mod_proxy_ajp"
   ## Proxy / Forwarding Settings ##
   <Proxy *>
      AddDefaultCharset Off
      Order allow,deny
      Allow from all
   </Proxy>

   # Proxy all requests to /server to Tomcat via AJP
   ProxyPass /server ajp://localhost:8009/server
   ProxyPassReverse /server ajp://localhost:8009/server

   # Optionally, also proxy Angular UI (if on same server). This requires "mod_proxy_http"
   #ProxyPass / http://localhost:4000/
   #ProxyPassReverse / http://localhost:4000/
</VirtualHost>
```

Sample shibboleth2.xml Configuration
In addition, here's a sample "ApplicationOverride" configuration for "shibboleth2.xml". This particular "ApplicationOverride" is configured to use the Test IdP provided by https://samltest.id/ and is just meant as an example. In order to enable it for testing purposes, you **must** specify `ShibRequestSetting applicationId samltest` in your Apach `mod_shib` configuration (see above). An additional, more detailed example is provided in

our "dspace-shibboleth" Docker configurations at https://github.com/DSpace/DSpace/blob/main/dspace/src/main/docker/dspace-shibboleth/shibboleth2.xml

```
        <!-- *** Sample Shibboleth Settings for  https://samltest.id/  ***     -->
        <!-- This provides a simple sample of how you could configure            -->
        <!-- shibboleth2.xml for DSpace sites.                                   -->
        <!-- TO ENABLE: You'd need to specify "applicationId" as "samltest" in   -->
        <!-- your mod_shib settings, e.g.                                        -->
        <!-- <Location />                                                        -->
        <!--     ...                                                             -->
        <!--     ShibRequestSetting applicationId samltest                       -->
        <!-- </Location>                                                         -->
        <ApplicationOverride id="samltest" entityID="http://[mydspace.edu]/shibboleth" REMOTE_USER="eppn
persistent-id targeted-id">

            <!-- We'll use a TEST IdP, hosted by the awesome https://samltest.id/ testing service. -->
            <!-- See also: https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPServiceSSO -->
            <!-- DSPACE 7 requires Shibboleth to use "SameSite=None" property for its Cookies -->
            <Sessions lifetime="28800" timeout="3600" checkAddress="false" relayState="ss:mem"
handlerSSL="true" cookieProps="; path=/; SameSite=None; secure; HttpOnly">
               <SSO entityID="https://samltest.id/saml/idp">
                 SAML2 SAML1
               </SSO>
            </Sessions>

            <!-- Loads and trusts a metadata file that describes the IdP and how to communicate with it. --
>
            <!-- By default, metadata is retrieved from the TEST IdP at https://samltest.id/ -->
            <!-- and is cached in a local file named "samltest-metadata.xml". -->
            <!-- See also: https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPMetadataProvider --
>
            <MetadataProvider type="XML" uri="https://samltest.id/saml/idp"
                          backingFilePath="samltest-metadata.xml" reloadInterval="180000"/>
        </ApplicationOverride>
```

Sample attribute-map.xml Configuration (for samltest.id)
In order to use the above example for https://samltest.id/, you may also need to modify your attribute-map.xml to support their attributes. Again, a more complete example is in our "dspace-shibboleth" Docker configurations at https://github.com/DSpace/DSpace/blob/main/dspace/src/main/docker/dspace-shibboleth/attribute-map.xml

```
<Attributes xmlns="urn:mace:shibboleth:2.0:attribute-map" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
    <!-- Custom Attributes specific to samltest.id -->
    <Attribute name="urn:oid:0.9.2342.19200300.100.1.1" id="uid"/>
    <Attribute name="urn:oid:0.9.2342.19200300.100.1.3" id="mail"/>
    <Attribute name="urn:oid:2.5.4.4" id="sn"/>
    <Attribute name="urn:oid:2.16.840.1.113730.3.1.241" id="displayName"/>
    <Attribute name="urn:oid:2.5.4.20" id="telephoneNumber"/>
    <Attribute name="urn:oid:2.5.4.42" id="givenName"/>
    <Attribute name="https://samltest.id/attributes/role" id="role"/>

    ...

</Attributes>
```

DSpace Shibboleth Configuration Options
Authentication Methods:

DSpace supports authentication using NetID, or email address. A user's NetID is a unique identifier from the IdP that identifies a particular user. The NetID can be of almost any form such as a unique integer, string, or with Shibboleth 2.0 you can use "targeted ids". You will need to coordinate with your shibboleth federation or identity provider. There are three ways to supply identity information to DSpace:

*1) NetID from Shibboleth Header (**best**)*

The NetID-based method is superior because users may change their email address with the identity provider. When this happens DSpace will not be able to associate their new address with their old account.

*2) Email address from Shibboleth Header (**okay**)*

In the case where a NetID header is not available or not found DSpace will fall back to identifying a user based-upon their email address.

*3) Tomcat's Remote User (**worst**)*

In the event that neither Shibboleth headers are found then as a last resort DSpace will look at Tomcat's remote user field. This is the least attractive option because Tomcat has no way to supply additional attributes about a user. Because of this the autoregister option is not supported if this method is used.

Identity Scheme Migration Strategies:

If you are currently using Email based authentication (either 1 or 2) and want to upgrade to NetID based authentication then there is an easy path. Simply enable shibboleth to pass the NetID attribute and set the netid-header below to the correct value. When a user attempts to log in to DSpace first DSpace will look for an EPerson with the passed NetID, however when this fails DSpace will fall back to email based authentication. Then DSpace will update the user's EPerson account record to set their NetID so all future authentications for this user will be based upon NetID. One thing to note is that DSpace will prevent an account from switching NetIDs. If an account already has a NetID set and then they try and authenticate with a different NetID the authentication will fail.

EPerson Metadata:

One of the primary benefits of using Shibboleth based authentication is receiving additional attributes about users such as their names, telephone numbers, and possibly their academic department or graduation semester if desired. DSpace treats the first and last name attributes differently because they (along with email address) are the three pieces of minimal information required to create a new user account. For both first and last name supply

direct mappings to the Shibboleth headers. In additional to the first and last name DSpace supports other metadata fields such as phone, or really anything you want to store on an eperson object. Beyond the phone field, which is accessible in the user's profile screen, none of these additional metadata fields will be used by DSpace out-of-the box. However if you develop any local modification you may access these attributes from the EPerson object. The Vireo ETD workflow system utilizes this to aid students when submitting an ETD.

Role-based Groups:

DSpace is able to place users into pre-defined groups based upon values received from Shibboleth. Using this option you can place all faculty members into a DSpace group when the correct affiliation's attribute is provided. When DSpace does this they are considered 'special groups', these are really groups but the user's membership within these groups is not recorded in the database. Each time a user authenticates they are automatically placed within the pre-defined DSpace group, so if the user loses their affiliation then the next time they login they will no longer be in the group.

Depending upon the shibboleth attributed use in the role-header it may be scoped. Scoped is shibboleth terminology for identifying where an attribute originated from. For example a students affiliation may be encoded as "student@tamu.edu". The part after the @ sign is the scope, and the preceding value is the value. You may use the whole value or only the value or scope. Using this you could generate a role for students and one institution different than students at another institution. Or if you turn on ignore-scope you could ignore the institution and place all students into one group.

The values extracted (a user may have multiple roles) will be used to look up which groups to place the user into. The groups are defined as "authentication-shibboleth.role.<role-name>" which is a comma separated list of DSpace groups.

Having issues getting Safari working?

In addition to the below settings, you may need to ensure your Shibboleth IdP is *trusted* by the DSpace backend by adding it to your `rest.cors.allowed-origins` configuration. *This is required for Safari web browsers to work with DSpace's Shibboleth plugin.*

For example, if your IdP is https://samltest.id/, then you need to append that URL to the comma-separated list of "allowed-origins" like:

```
rest.cors.allowed-origins = ${dspace.ui.url}, https://samltest.id
```

More information on this configuration can be found in the REST API(see page 502) documentation.

Configuration File:	**`[dspace]/config/modules/authentication-shibboleth.cfg`**
Property:	`authentication-shibboleth.lazysession`
Example Value:	`authentication-shibboleth.lazysession = true`
Informational Note:	Whether to use lazy sessions or active sessions. For more DSpace instances, you will likely want to use lazy sessions. Active sessions will force **every** user to authenticate via Shibboleth before they can access your DSpace (essentially resulting in a "dark archive").

Property:	`authentication-shibboleth.lazysession.loginurl`
Example Value:	authentication-shibboleth.`lazysession.loginurl = /Shibboleth.sso/Login`
Informational Note:	The url to start a shibboleth session (only for lazy sessions). Generally this setting will be "/Shibboleth.sso/Login"
Property:	`authentication-shibboleth.lazysession.secure`
Example Value:	`authentication-shibboleth.lazysession.secure = true`
Informational Note:	Force HTTPS when authenticating (only for lazy sessions). Generally this is recommended to be "true".
Property:	`authentication-shibboleth.netid-header`
Example Value:	`authentication-shibboleth.netid-header = SHIB-NETID`
Informational Note:	The HTTP header where shibboleth will supply a user's NetID. This HTTP header should be specified as an Attribute within your Shibboleth "attribute-map.xml" configuration file.
Property:	`authentication-shibboleth.email-header`
Example Value:	`authentication-shibboleth.email-header = SHIB-MAIL`
Informational Note:	The HTTP header where the shibboleth will supply a user's email address. This HTTP header should be specified as an Attribute within your Shibboleth "attribute-map.xml" configuration file.
Property:	`authentication-shibboleth.email-use-tomcat-remote-user`
Example Value:	`authentication-shibboleth.email-use-tomcat-remote-user = false`

Informational Note:	Used when a netid or email headers are not available should Shibboleth authentication fall back to using Tomcat's remote user feature? Generally this is not recommended. See the "Authentication Methods" section above.
Property:	`authentication-shibboleth.reconvert.attributes`
Example Value	`authentication-shibboleth.reconvert.attributes = false`
Informational Note:	Shibboleth attributes are by default UTF-8 encoded. Some servlet container automatically converts the attributes from ISO-8859-1 (latin-1) to UTF-8. As the attributes already were UTF-8 encoded it may be necessary to reconvert them. If you set this property true, DSpace converts all shibboleth attributes retrieved from the servlet container from UTF-8 to ISO-8859-1 and uses the result as if it were UTF-8. This procedure restores the shibboleth attributes if the servlet container wrongly converted them from ISO-8859-1 to UTF-8. Set this true, if you notice character encoding problems within shibboleth attributes.
Property:	`authentication-shibboleth.autoregister`
Example Value:	`authentication-shibboleth.autoregister = true`
Informational Note:	Should we allow new users to be registered automatically?
Property:	`authentication-shibboleth.sword.compatibility`
Example Value:	`authentication-shibboleth.sword.compatibility = false`
Informational Note:	SWORD compatibility will allow this authentication method to work when using SWORD. SWORD relies on username and password based authentication and is entirely incapable of supporting shibboleth. This option allows you to authenticate username and passwords for SWORD sessions with out adding another authentication method onto the stack. You will need to ensure that a user has a password. One way to do that is to create the user via the create-administrator command line command and then edit their permissions. WARNING: If you enable this option while ALSO having "PasswordAuthentication" enabled, then you should ensure that "PasswordAuthentication" is listed prior to "ShibAuthentication" in your authentication.cfg file. Otherwise, ShibAuthentication will be used to authenticate all of your users INSTEAD OF PasswordAuthentication.

Property:	`authentication-shibboleth.firstname-header`
Example Value:	`authentication-shibboleth.firstname-header = SHIB_GIVENNAME`
Informational Note:	The HTTP header where the shibboleth will supply a user's given name. This HTTP header should be specified as an Attribute within your Shibboleth "attribute-map.xml" configuration file.
Property:	`authentication-shibboleth.lastname-header`
Example Value:	`authentication-shibboleth.lastname-header = SHIB_SN`
Informational Note:	The HTTP header where the shibboleth will supply a user's surname. This HTTP header should be specified as an Attribute within your Shibboleth "attribute-map.xml" configuration file.
Property:	`authentication-shibboleth.eperson.metadata`
Example Value:	`authentication-shibboleth.eperson.metadata = \` ` SHIB-telephone => phone, \` ` SHIB-cn => cn`
Informational Note:	Additional user attributes mapping, multiple attributes may be stored for each user. The left side is the Shibboleth-based metadata Header and the right side is the eperson metadata field to map the attribute to.
Property:	`authentication-shibboleth.eperson.metadata.autocreate`
Example Value:	`authentication-shibboleth.eperson.metadata.autocreate = true`
Informational Note:	If the eperson metadata field is not found, should it be automatically created?
Property:	`authentication-shibboleth.role-header`

Example Value:	`authentication-shibboleth.role-header = SHIB_SCOPED_AFFILIATION`
Informational Note:	The shibboleth header to do role-based mappings (see section on roll based mapping section above)
Property:	`authentication-shibboleth.role-header.ignore-scope`
Example Value:	`authentication-shibboleth.role-header.ignore-scope = true`
Informational Note:	Weather to ignore the attribute's scope (everything after the @ sign for scoped attributes)
Property:	`authentication-shibboleth.role-header.ignore-value`
Example Value:	`authentication-shibboleth.role-header.ignore-value = false`
Informational Note:	Weather to ignore the attribute's value (everything before the @ sign for scoped attributes)
Property:	`authentication-shibboleth.role.[affiliation-attribute]`
Example Value:	`authentication-shibboleth.role.faculty = Faculty, Member` `authentication-shibboleth.role.staff = Staff, Member` `authentication-shibboleth.role.student = Students, Member`
Informational Note:	Mapping of affiliation values to DSpace groups. See the "Role-based Groups" section above for more info.

LDAP Authentication

Introduction to LDAP specific terminology

If you are unfamiliar with LDAP, the following introduction to some of its terminology might come in handy:

https://stackoverflow.com/questions/18756688/what-are-cn-ou-dc-in-an-ldap-search

Enabling LDAP Authentication

To enable LDAP Authentication, you must ensure the `org.dspace.authenticate.LDAPAuthentication` class is listed as one of the AuthenticationMethods in the following configuration:

Configuration File:	**`[dspace]/config/modules/authentication.cfg`**
Property:	`plugin.sequence.org.dspace.authenticate.AuthenticationMethod`
Example Value:	`plugin.sequence.org.dspace.authenticate.AuthenticationMethod = org.dspace.authenticate.LDAPAuthentication`

Configuring LDAP Authentication

If LDAP is enabled, then new users will be able to register by entering their username and password without being sent the registration token. If users do not have a username and password, then they can still register and login with just their email address the same way they do now.

If you want to give any special privileges to LDAP users, create a stackable authentication method to automatically put people who have a netid into a special group. You might also want to give certain email addresses special privileges. Refer to the Custom Authentication Code section(see page 113) below for more information about how to do this.

Ensure required commas are escaped in LDAP configuration

NOTE: As of DSpace 6, commas (,) are now a special character in the Configuration (see page 552)system. As some LDAP configuration may contain commas, you must be careful to escape any required commas by adding a backslash (\) before each comma, e.g. "\,". The configuration reference for authentication-ldap.cfg has been updated below with additional examples.

Here is an explanation of each of the different LDAP configuration parameters:

Configuration File:	**`[dspace]/config/modules/authentication-ldap.cfg`**
Property:	`authentication-ldap.enable`
Example Value:	`authentication-ldap.enable = false`

Informational Note:	This setting will enable or disable LDAP authentication in DSpace. With the setting off, users will be required to register and login with their email address. With this setting on, users will be able to login and register with their LDAP user ids and passwords.
Property:	`authentication-ldap.autoregister`
Example Value:	`authentication-ldap.autoregister = true`
Informational Note:	This will turn LDAP autoregistration on or off. With this on, a new EPerson object will be created for any user who successfully authenticates against the LDAP server when they first login. With this setting off, the user must first register to get an EPerson object by entering their ldap username and password and filling out the forms.
Property:	`authentication-ldap.provider_url`
Example Value:	`authentication-ldap.provider_url = ldap://ldap.myu.edu/o=myu.edu\,ou=mydept`
Informational Note:	This is the url to your institution's LDAP server. You may or may not need the /o=myu.edu part at the end. Your server may also require the ldaps:// protocol. (This field has no default value) **NOTE:** As of DSpace 6, commas (,) are now a special character in the Configuration (see page 552)system. Therefore, be careful to escape any required commas in this configuration by adding a backslash (\) before each comma, e.g. "\,"
Property:	`authentication-ldap.starttls`
Example Value:	`authentication-ldap.starttls = false`
Informational Note:	Should we issue StartTLS after establishing TCP connection in order to initiate an encrypted connection? Note: This (TLS) is different from LDAPS: • TLS is a tunnel for plain LDAP and is typically recognized on the same port (standard LDAP port: 389) • LDAPS is a separate protocol, deprecated in favor of the standard TLS method. (standard LDAPS port: 636)

Property:	`authentication-ldap.id_field`
Example Value:	`authentication-ldap.id_field = uid`
Explanation:	This is the unique identifier field in the LDAP directory where the username is stored. (This field has no default value)
Property:	`authentication-ldap.object_context`
Example Value:	`authentication-ldap.object_context = ou=people\,o=myu.edu`
Informational Note:	This is the LDAP object context to use when authenticating the user. By default, DSpace will use this value to create the user's DN in order to attempt to authenticate them. It is appended to the *id_field* and username. For example `uid=username\,ou=people\,o=myu.edu`. You will need to modify this to match your LDAP configuration. (This field has no default value) If your users do NOT all exist under a single "object_context" in LDAP, then you should ignore this setting and INSTEAD use the Hierarchical LDAP Authentication settings below(see page 110) (especially see "`search.user`" or "`search.anonymous`") **NOTE:** As of DSpace 6, commas (,) are now a special character in the Configuration (see page 552)system. Therefore, be careful to escape any required commas in this configuration by adding a backslash (\) before each comma, e.g. "\,"
Property:	`authentication-ldap.search_context`
Example Value:	`authentication-ldap.search_context = ou=people`

Informational Note:	This is the search context used when looking up a user's LDAP object to retrieve their data for autoregistering. With `autoregister=true`, when a user authenticates without an EPerson object we search the LDAP directory to get their name (`id_field`) and email address (`email_field`) so that we can create one for them. So after we have authenticated against *uid=username,ou=people,o=byu.edu* we now search in ou=people for filtering on [uid=username]. Often the `search_context` is the same as the `object_context` parameter. But again this depends on your LDAP server configuration. (This field has no default value, and it MUST be specified when either `search.anonymous=true` or `search.user` is specified) **NOTE:** As of DSpace 6, commas (,) are now a special character in the Configuration (see page 552)system. Therefore, be careful to escape any required commas in this configuration by adding a backslash (\) before each comma, e.g. "\,"
Property:	`authentication-ldap.email_field`
Example Value:	`authentication-ldap.email_field = mail`
Informational Note:	This is the LDAP object field where the user's email address is stored. "mail" is the most common for LDAP servers. (This field has no default value) If the "email_field" is unspecified, or the user has no email address in LDAP, his/her username (id_field value) will be saved as the email in DSpace (or appended to `netid_email_domain`, when specified)
Property:	`authentication-ldap.netid_email_domain`
Example Value:	`authentication-ldap.netid_email_domain = @example.com`[171]

171 http://example.com

Informational Note:	If your LDAP server does not hold an email address for a user (i.e. no `email_field`), you can use the following field to specify your email domain. This value is appended to the netid (`id_field`) in order to make an email address (which is then stored in the DSpace EPerson). For example, a netid of 'user' and `netid_email_domain` as `@example.com`[172] would set the email of the user to be user@example.com[173] *Please note:* this field will only be used if "`email_field`" is unspecified OR the user in question has no email address stored in LDAP. If both "`email_field`" and "`netid_email_domain`" are unspecified, then the "`id_field`" will be used as the email address.
Property:	`authentication-ldap.surname_field`
Example Value:	`authentication-ldap.surname_field = sn`
Informational Note:	This is the LDAP object field where the user's last name is stored. "`sn`" is the most common for LDAP servers. If the field is not found the field will be left blank in the new eperson object. (This field has no default value)
Property:	`authentication-ldap.givenname_field`
Example Value:	`authentication-ldap.givenname_field = givenName`
Informational Note:	This is the LDAP object field where the user's given names are stored. I'm not sure how common the givenName field is in different LDAP instances. If the field is not found the field will be left blank in the new eperson object. (This field has no default value)
Property:	`authentication-ldap.phone_field`
Example Value:	`authentication-ldap.phone_field = telephoneNumber`
Informational Note:	This is the field where the user's phone number is stored in the LDAP directory. If the field is not found the field will be left blank in the new eperson object. (This field has no default value)
Property:	`authentication-ldap.login.specialgroup`

172 http://example.com
173 mailto:user@example.com

Example Value:	`authentication-ldap.login.specialgroup = group-name`
Informational Note:	If specified, all user sessions successfully logged in via LDAP will automatically become members of this DSpace Group (for the remainder of their current, logged in session). This DSpace Group **must** already exist (it will not be automatically created). This is useful if you want a DSpace Group made up of all internal authenticated users. This DSpace Group can then be used to bestow special permissions on any users who have authenticated via LDAP (e.g. you could allow anyone authenticated via LDAP to view special, on campus only collections or similar)
Property:	`login.groupmap.*`
Example Value:	`authentication-ldap.login.groupmap.1 = ou=Students:ALL_STUDENTS` `authentication-ldap.login.groupmap.2 = ou=Employees:ALL_EMPLOYEES` `authentication-ldap.login.groupmap.3 = ou=Faculty:ALL_FACULTY`
Informational Note:	The left part of the value (before the ":") must correspond to a portion of a user's DN (unless "`login.group.attribute`" is specified..please see below). The right part of the value corresponds to the name of an existing DSpace group. For example, if the authenticated user's DN in LDAP is in the following form: `cn=jdoe,OU=Students,OU=Users,dc=example,dc=edu` that user would get assigned to the ALL_STUDENTS DSpace group for the remainder of their current session. However, if that same user later graduates and is employed by the university, their DN in LDAP may change to: `cn=jdoe,OU=Employees,OU=Users,dc=example,dc=edu` Upon logging into DSpace after that DN change, the authenticated user would now be assigned to the ALL_EMPLOYEES DSpace group for the remainder of their current session. *Note:* This option can be used independently from the login.specialgroup option, which will put all LDAP users into a single DSpace group. Both options may be used together.
Property:	`authentication-ldap.login.groupmap.attribute`

Example Value:	`authentication-ldap.login.groupmap.attribute = group`
Informational Note:	The value of the "`authentication-ldap.login.groupmap.attribute`" should specify the name of a single LDAP attribute. If this property is uncommented, it changes the meaning of the left part of "`authentication-ldap.login.groupmap.*`" (see above) as follows: • If the authenticated user has this LDAP attribute, look up the value of this LDAP attribute in the left part (before the ":") of the `authentication-ldap.login.groupmap.*` value • If that LDAP value is found in any "`authentication-ldap.login.groupmap.*`" field, assign this authenticated user to the DSpace Group specified by the right part (after the ":") of the `authentication-ldap.login.groupmap.*` value. For example: • `authentication-ldap.login.groupmap.attribute = group` • `authentication-ldap.login.groupmap.1 = mathematics:Mathematics_Group` The above would ensure that any authenticated users where their LDAP "group" attribute equals "mathematics" would be added to the DSpace Group named "Mathematics_Group" for the remainder of their current session. However, if that same user logged in later with a new LDAP "group" value of "computer science", he/she would no longer be a member of the "Mathematics_Group" in DSpace.

Debugging LDAP connection and configuration

As every LDAP is different, configuring your DSpace to communicate with your LDAP can sometimes be a challenge. We recommend using third-party LDAP tools to test your LDAP connection / username / password, and perform sample searches to better understand what information is being returned from your local LDAP. This will help ensure that LDAP configuration goes more smoothly.

One example of such an LDAP tool is the `ldapsearch`[174] commandline tool available in most Linux operating systems (e.g. in Debian / Ubuntu it's available in the "ldap-utils" package). Below are some example ldapsearch commands that can be used to determine (and/or debug) specific configurations in your `authentication-ldap.cfg`. In the below examples, we've used the names of specific DSpace configurations as placeholders (in square brackets).

174 https://linux.die.net/man/1/ldapsearch

```
# Basic anonymous connection (for VERBOSE, add -v)
ldapsearch -x -H [provider_url]

# Debug a connection error (add -d-1)
# If you are connecting to an LDAPS URL and see connection errors (e.g. "peer cert untrusted or revoked")
# then see below note about "SSL Connection Errors"
ldapsearch -x -H [provider_url] -d-1


# Attempt to connect to [provider_url] as [search.user] (will prompt for search.user's password)
# This doesn't actually perform a query, just ensures that authentication is working
# NOTE: "search.user" is USUALLY either the full user DN (e.g. "cn=dspaceadmin,ou=people,o=myu.edu")
# or "DOMAIN\USERNAME" (e.g. "MYU\DSpaceUser"). The latter is more likely with Windows Active Directory
ldapsearch -x -H [provider_url] -D [search.user] -W


# Attempt to list the first 100 users in a given [search_context], returning the "cn", "mail" and "sn"
 fields for each
ldapsearch -x -H [provider_url] -D [search.user] -W -b [search_context] -z 100 cn mail sn


# Attempt to find the first 100 users whose [id_field] starts with the letter "t", returning the
[id_field], "cn", "mail" and "sn" fields for each
ldapsearch -x -H [provider_url] -D [search.user] -W -b [search_context] -z 100 -s sub "([id_field]=t*)"
 [id_field] cn mail sn
```

SSL Connection Errors: If you are using ldapsearch with an LDAPS connection (secure connection), you may receive "peer cert untrusted or revoked" errors if the LDAP SSL certificate is self-signed. You can temporarily tell LDAP to accept any security certificate by setting `TLS_REQCERT allow` in your ldapsearch's `ldap.conf` file. *Be sure to remove this setting however after you are done testing!*

```
# FOR TESTING ONLY! This setting disables the check for a valid LDAP Server security certificate,
# which is considered a security issue for production LDAP setups. Setting this to "allow" tells
# the LDAP client to accept any security certificates that it cannot verify or validate.
TLS_REQCERT allow
```

More information on this SSL workaround can be found at:

- http://www.bind9.net/manual/openldap/2.3/tls.html
- http://muzso.hu/2012/03/29/how-to-configure-ssl-aka.-ldaps-for-libnss-ldap-auth-client-config-in-ubuntu

Enabling Hierarchical LDAP Authentication

> Please note, that DSpace doesn't contain the `LDAPHierarchicalAuthentication` class anymore. This functionality is now supported by `LDAPAuthentication`, which uses the same configuration options.

If your users are spread out across a hierarchical tree on your LDAP server, you may wish to have DSpace search for the user name in your tree. Here's how it works:

1. DSpace gets the user name from the login form

2. DSpace binds to LDAP as an administrative user with right to search in DNs (LDAP may be configured to allow anonymous users to search)
3. DSpace searches for the user name as within DNs (username is a part of full DN)
4. DSpace binds with the found full DN and password from login form
5. DSpace logs user in if LDAP reports successful authentication; refuses login otherwise

Configuring Hierarchical LDAP Authentication

Hierarchical LDAP Authentication shares all the above standard LDAP configurations(see page 102), but has some additional settings.

You can optionally specify the search scope. If anonymous access is not enabled on your LDAP server, you will need to specify the full DN and password of a user that is allowed to bind in order to search for the users.

Configuration File:	**`[dspace]/config/modules/authentication-ldap.cfg`**
Property:	`authentication-ldap.search_scope`
Example Value:	`authentication-ldap.search_scope = 2`
Informational Note:	This is the search scope value for the LDAP search during autoregistering (`autoregister=true`). This will depend on your LDAP server setup, and is only really necessary if your users are spread out across a hierarchical tree on your LDAP server. This value must be one of the following integers corresponding to the following values: `object scope : 0` `one level scope : 1` `subtree scope : 2` Please note that "`search_context`" in the LDAP configurations must also be specified.
Property:	`authentication-ldap.search.anonymous`
Example Value:	`authentication-ldap.search.anonymous = true`
Informational Note:	If true, DSpace will anonymously search LDAP (in the "`search_context`") for the DN of the user trying to login to DSpace. This setting is "false" by default. By default, DSpace will either use "`search.user`" to authenticate for the LDAP search (if search.user is specified), or will use the "`object_context`" value to create the user's DN.

Property:	`authentication-ldap.search.user` `authentication-ldap.search.password`
Example Value:	`authentication-ldap.search.user = cn=admin\,ou=people\,o=myu.edu` `authentication-ldap.search.password = password`
Informational Note:	The full DN and password of a user allowed to connect to the LDAP server and search (in the "`search_context`") for the DN of the user trying to login. By default, if unspecified, DSpace will either search LDAP anonymously for the user's DN (when `search.anonymous=true`), or will use the "`object_context`" value to create the user's DN. **NOTE:** As of DSpace 6, commas (,) are now a special character in the Configuration (see page 552)system. Therefore, be careful to escape any required commas in this configuration by adding a backslash (\) before each comma, e.g. "\,"

IP Authentication

Enabling IP Authentication

To enable IP Authentication, you must ensure the `org.dspace.authenticate.IPAuthentication` class is listed as one of the AuthenticationMethods in the following configuration:

Configuration File:	**`[dspace]/config/modules/authentication.cfg`**
Property:	`plugin.sequence.org.dspace.authenticate.AuthenticationMethod`
Example Value:	`plugin.sequence.org.dspace.authenticate.AuthenticationMethod = org.dspace.authenticate.IPAuthentication`

Configuring IP Authentication

Configuration File:	**`[dspace]/config/modules/authentication-ip.cfg`**

Once enabled, you are then able to map DSpace groups to IP addresses in `authentication-ip.cfg` by setting `ip.GROUPNAME = iprange[, iprange ...]`, e.g:

```
authentication-ip.MY_UNIVERSITY = 10.1.2.3, \                  # Full IP
13.5, \                      # Partial IP
11.3.4.5/24, \               # with CIDR
12.7.8.9/255.255.128.0, \    # with netmask
2001:18e8::32                # IPv6 too
```

Negative matches can be set by prepending the entry with a '-'. For example if you want to include all of a class B network except for users of a contained class c network, you could use: 111.222,-111.222.333.

Notes:

- If the Groupname contains blanks you must escape the spaces, e.g. "Department\ of\ Statistics"
- If your DSpace installation is hidden behind a web proxy, remember to set the `useProxies` configuration option within the 'Logging' section of `dspace.cfg` to use the IP address of the user rather than the IP address of the proxy server.

X.509 Certificate Authentication

Enabling X.509 Certificate Authentication

The X.509 authentication method uses an X.509 certificate sent by the client to establish his/her identity. It requires the client to have a personal Web certificate installed on their browser (or other client software) which is issued by a Certifying Authority (CA) recognized by the web server.

1. See the HTTPS installation instructions(see page 87) to configure your Web server. If you are using HTTPS with Tomcat, note that the `<Connector>` tag *must* include the attribute `clientAuth="true"` so the server requests a personal Web certificate from the client.
2. Add the `org.dspace.authenticate.X509Authentication` plugin `first` to the list of stackable authentication methods in the value of the configuration key `plugin.sequence.org.dspace.authenticate.AuthenticationMethod`

Configuration File:	**[dspace]/config/modules/authentication.cfg**
Property:	`plugin.sequence.org.dspace.authenticate.AuthenticationMethod`
Example Value:	`plugin.sequence.org.dspace.authenticate.AuthenticationMethod = org.dspace.authenticate.X509Authentication` `plugin.sequence.org.dspace.authenticate.AuthenticationMethod = org.dspace.authenticate.PasswordAuthentication`

Configuring X.509 Certificate Authentication

Configuration File:	**[dspace]/config/modules/authentication-x509.cfg**

1. You must also configure DSpace with the same CA certificates as the web server, so it can accept and interpret the clients' certificates. It can share the same keystore file as the web server, or a separate one, or a CA certificate in a file by itself. Configure it by *one*of these methods, either the Java keystore

```
authentication-x509.keystore.path =  path to Java keystore file
authentication-x509.keystore.password =  password to access the keystore
```

   ...or the separate CA certificate file (in PEM or DER format):

```
authentication-x509.ca.cert =  path to certificate file for CA whose client certs to accept.
```

2. Choose whether to enable auto-registration: If you want users who authenticate successfully to be automatically registered as new E-Persons if they are not already, set the `autoregister` configuration property to `true`. This lets you automatically accept all users with valid personal certificates. The default is `false`.

TODO: document the remaining authentication-x509.* properties

Example of a Custom Authentication Method

Also included in the source is an implementation of an authentication method used at MIT, *edu.mit.dspace.MITSpecialGroup*. This does not actually authenticate a user, it *only* adds the current user session to a special (dynamic) group called 'MIT Users' (which must be present in the system!). This allows us to create authorization policies for MIT users without having to manually maintain membership of the MIT users group.

By keeping this code in a separate method, we can customize the authentication process for MIT by simply adding it to the stack in the DSpace configuration. None of the code has to be touched.

You can create your own custom authentication method and add it to the stack. Use the most similar existing method as a model, e.g. `org.dspace.authenticate.PasswordAuthentication` for an "explicit" method (with credentials entered interactively) or `org.dspace.authenticate.X509Authentication` for an implicit method.

4.1.2 Embargo

4.1.2.1 What is an Embargo?

An embargo is a temporary access restriction placed on metadata or bitstreams (i.e. files). Its scope or duration may vary, but the fact that it eventually expires is what distinguishes it from other content restrictions. For example, it is not unusual for content destined for DSpace to come with permanent restrictions on use or access based on license-driven or other IP-based requirements that limit access to institutionally affiliated users. Restrictions such as these are imposed and managed using standard administrative tools in DSpace, typically by attaching specific access policies (aka "resource policies") to Items, Collections, Bitstreams, etc.

Embargo functionality was originally introduced as part of DSpace 1.6, enabling embargoes on the level of items that applied to all bitstreams included in the item. Since DSpace 3.0, this functionality has been extended to the Submission User Interface, enabling embargoes on the level of individual bitstreams.

4.1.2.2 DSpace Embargo Functionality

Embargoes can be applied per *item (including metadata)* and per *bitstream (i.e. file)*. The *item* level embargo will be the default for every *bitstream*, although it could be customized at *bitstream* level.

When an embargo is set on either an item level or a bitstream level, a new ResourcePolicy (i.e. access policy) is added to the corresponding Item or Bitstream. **This ResourcePolicy will automatically control the lifting of the embargo (when the embargo date passes).** An embargo lift date is generally stored as the "start date" of such a policy. Essentially, this means that the access rights defined in the policy do not get applied until *after that date passes* (and prior to that date, the access rights will default to Admin only).

The scheduled, manual "embargo-lifter" commands (used prior to DSpace 3) are no longer necessary and not recommended to run.

Managing Embargoes on existing Items

Administrators are able to change the lift date of any embargo by editing the authorization policy (ResourcePolicy) on the object. These authorization policies can be managed from the Edit Item screen by clicking on "Authorizations".

- To add an embargo, edit the appropriate policy and set a "start date". To add an full Item embargo (including metadata), edit the Item policy. To embargo individual bitstreams, edit the appropriate Bitstream policy.
- To remove an embargo, edit the appropriate policy, and clear out the "start date".
- To change an embargo, edit the appropriate policy, and change the "start date" to a new date.

Changes to the embargo should take effect immediately. However, as Administrators have full access to embargoed items, you may need to log out first. After logging out, you will be subject to the embargo.

4.1.2.3 Configuring and using Embargo in DSpace Submission User Interface

> Item-level embargo is not yet supported in DSpace 7.0 in the Submission user interface. In the Submission UI, only embargoes on specific bitstreams (files) is supported. However, you can add an item-level embargo in DSpace 7.0 using the "Manage Embargoes on existing Items" approach described above.
>
> Item-level embargo will be available in a future 7.x release. See DSpace Release 7.0 Status[175]

Starting in DSpace 7, embargo (and lease) settings are configurable via a Spring Bean configuration file `[dspace]/config/spring/api/access-conditions.xml`

For detailed information on configuring your Embargo options (and other related options like lease or restrict to a particular group of users), see the section on "*Configuring the File Upload Step*" of the Submission User Interface(see page 260).

Private/Public Item

It is also possible to adjust the Private/Public state of an item after it has been archived in the repository. This can be achieved from either the "Admin Search" (/admin/search), or from the "Status" tab under "Edit Item".

Private items are not retrievable through the DSpace search, browse or Discovery indexes.

Therefore, an "Admin Search" option is provided, which allows you to search across all items, including private or withdrawn items. You can also filter your results to display only private items.

Pre-3.0 Embargo Migration Routine

If you have just upgraded from a DSpace 1.x.x version, any embargoes that are currently "in effect" will need to be migrated into ResourcePolicies. Prior to 3.0, embargoes in DSpace were managed entirely in metadata fields (and required running a scheduled "embargo-lifter" command). However, DSpace now stores all embargo information directly on ResourcePolicies (i.e. "access policies"). These ResourcePolicies automatically "lift" an embargo after the embargo date passes.

In order to migrate old embargoes into ResourcePolicies, a migration routine has been developed. **Please note that this migration routine should only need to be run ONCE** (immediately after an upgrade from 1.x.x to a more recent version of DSpace). After that point, any newly defined embargoes will automatically be stored on ResourcePolicies.

To execute it, run the following command:

```
[dspace]/bin/dspace migrate-embargo -a
```

4.1.2.4 Technical Specifications

Introduction

The following sections illustrate the technical changes that have been made to the *back-end* to add the new *Advanced Embargo* functionality.

175 https://wiki.lyrasis.org/display/DSPACE/DSpace+Release+7.0+Status

ResourcePolicy

When an embargo is set at *item* level or *bitstream* level, a new *ResourcePolicy* will be added.

Three new attributes have been introduced in the *ResourcePolicy* class:

- *rpname*: resource policy name
- *rptype*: resource policy type
- *rpdescription*: resource policy description

While *rpname* and *rpdescription* are fields manageable by users, the *rptype* is managed by DSpace itself. It represents a type that a resource policy can assume, among the following:

- TYPE_SUBMISSION: all the policies added automatically during the submission process
- TYPE_WORKFLOW: all the policies added automatically during the workflow stage
- TYPE_CUSTOM: all the custom policies added by users
- TYPE_INHERITED: all the policies inherited from the enclosing object (for Item, a Collection; for Bitstream, an Item).

Here is an example of all information contained in a single policy record:

```
policy_id: 4847
resource_type_id: 2
resource_id: 89
action_id: 0
eperson_id:
epersongroup_id: 0
start_date: 2013-01-01
end_date:
rpname: Embargo Policy
rpdescription:  Embargoed through 2012
rptype: TYPE_CUSTOM
```

Item

To manage **Private/Public** state a new *boolean* attribute has been added to the Item:

- isDiscoverable

When an Item is private, the attribute will assume the value ***false***.

Item.inheritCollectionDefaultPolicies(Collection c)

This method has been adjusted to leave custom policies, added by the users, in place and add the default collection policies only if there are no custom policies.

AuthorizeService

Some methods have been changed on *AuthorizeService* to manage the new fields and some convenience methods have been introduced:

```
public static List<ResourcePolicy> findPoliciesByDSOAndType(Context c, DSpaceObject o, String type);
public static void removeAllPoliciesByDSOAndTypeNotEqualsTo(Context c, DSpaceObject o, String type);
public static boolean isAnIdenticalPolicyAlreadyInPlace(Context c, DSpaceObject o, ResourcePolicy rp);
public static ResourcePolicy createOrModifyPolicy(ResourcePolicy policy, Context context, String name, int
 idGroup, EPerson ePerson, Date embargoDate, int action, String reason, DSpaceObject dso);
```

Withdraw Item

The feature to withdraw an item from the repository has been modified to keep all the custom policies in place.

Reinstate Item

The feature to reinstate an item in the repository has been modified to preserve existing custom policies.

Pre-DSpace 3.0 Embargo Compatibility

The Pre-DSpace 3.0 embargo functionality (see below) has been modified to adjust the policies setter and lifter. These classes now also set the dates within the policy objects themselves in addition to setting the date in the item metadata.

4.1.2.5 Creating Embargoes via Metadata

Introduction

Prior to DSpace 3.0, all DSpace embargoes were stored as metadata. While embargoes are no longer stored permanently in metadata fields (they are now stored on ResourcePolicies, i.e. access policies), embargoes *can still be initialized via metadata fields.*

This ability to create/initialize embargoes via metadata is extremely powerful if you wish to submit embargoed content via electronic means (such as Importing Items via Simple Archive Format(see page 233), SWORDv1(see page 216), SWORDv2(see page 202), etc).

Setting Embargo terms via metadata

Functionally, the embargo system allows you to attach "terms" to an item before it is placed into the repository, which express how the embargo should be applied. What do we mean by "terms" here? They are really any expression that the system is capable of turning into (1) the time the embargo expires, and (2) a concrete set of access restrictions. Some examples:

"2020-09-12" - an absolute date (i.e. the date embargo will be lifted)
"6 months" - a time relative to when the item is accessioned
"forever" - an indefinite, or open-ended embargo
"local only until 2015" - both a time and an exception (public has no access until 2015, local users OK immediately)
"Nature Publishing Group standard" - look-up to a policy somewhere (typically 6 months)

These terms are interpreted by the embargo system to yield a specific date on which the embargo can be removed (or "lifted"), and a specific set of access policies. Obviously, some terms are easier to interpret than others (the absolute date really requires none at all), and the default embargo logic understands only the most basic terms (the first and third examples above). But as we will see below, the embargo system provides you with the ability to add your own interpreters to cope with any terms expressions you wish to have. This date that is the result of the interpretation is stored with the item. The embargo system detects when that date has passed, and removes the

embargo ("lifts it"), so the item bitstreams become available. Here is a more detailed life-cycle for an embargoed item:

Terms assignment

The first step in placing an embargo on an item is to attach (assign) "terms" to it. If these terms are missing, no embargo will be imposed. As we will see below, terms are carried in a configurable DSpace metadata field, so assigning terms just means assigning a value to a metadata field. This can be done in a web submission user interface form, in a SWORD deposit package, a batch import, etc. - anywhere metadata is passed to DSpace. The terms are not immediately acted upon, and may be revised, corrected, removed, etc, up until the next stage of the life-cycle. Thus a submitter could enter one value, and a collection editor replace it, and only the last value will be used. Since metadata fields are multivalued, theoretically there can be multiple terms values, but in the default implementation only one is recognized.

Terms interpretation/imposition

In DSpace terminology, when an Item has exited the last of any workflow steps (or if none have been defined for it), it is said to be "installed" into the repository. At this precise time, the interpretation of the terms occurs, and a computed "lift date" is assigned, and recorded as part of the ResourcePolicy (aka policy) of the Item. Once the lift date has been assigned to the ResourcePolicy, the metadata field which defined the embargo is **cleared**. From that point forward, all embargo information is controlled/defined by the ResourcePolicy.

It is important to understand that this interpretation happens only once, (just like the installation). Therefore, **updating/changing an embargo cannot be done via metadata fields.** Instead, all embargo updates must be made to the ResourcePolicies themselves (e.g. ResourcePolicies can be managed from the Admin UI in the Edit Item screens).

Also note that since these policy changes occur before installation, there is no time during which embargoed content is "exposed" (accessible by non-administrators). The terms interpretation and imposition together are called "setting" the embargo, and the component that performs them both is called the embargo "setter".

Embargo period

After an embargoed item has been installed, the policy restrictions remain in effect until the embargo date passes. Once the embargo date passes, the policy restrictions are automatically lifted. An embargo lift date is generally stored as the "start date" of a policy. Essentially, this means that the policy does not get applied until after that date passes (and prior to that date, the object defaults to Admin only access).

Administrators are able to change the lift date of the embargo by editing the policy (ResourcePolicy). These policies can be managed from the Edit Item screens.

Configuration of metadata fields

DSpace embargoes utilize standard metadata fields to hold both the "terms" and the "lift date". Which fields you use are configurable, and no specific metadata element is dedicated or pre-defined for use in embargo. Rather, you must specify exactly what field you want the embargo system to examine when it needs to find the terms or assign the lift date.

The properties that specify these assignments live in dspace.cfg:

```
# DC metadata field to hold the user-supplied embargo terms
embargo.field.terms = SCHEMA.ELEMENT.QUALIFIER

# DC metadata field to hold computed "lift date" of embargo
embargo.field.lift = SCHEMA.ELEMENT.QUALIFIER
```

You replace the placeholder values with real metadata field names. If you only need the "default" embargo behavior - which essentially accepts only absolute dates as "terms" - this is the only configuration required, except as noted below.

There is also a property for the special date of "forever":

```
# string in terms field to indicate indefinite embargo
embargo.terms.open = forever
```

which you may change to suit linguistic or other preference.

You are free to use existing metadata fields, or create new fields. If you choose the latter, you must understand that the embargo system does **not** create or configure these fields: i.e. you must follow all the standard documented procedures for actually creating them (i.e. adding them to the metadata registry, or to display templates, etc) - this does not happen automatically. Likewise, if you want the field for "terms" to appear in submission screens and workflows, you must follow the documented procedure for configurable submission (basically, this means adding the field to submission-forms.xml). The flexibility of metadata configuration makes if easy for you to restrict embargoes to specific collections, since configurable submission can be defined per collection.

Key recommendations:

1. Use a local metadata schema. Breaking compliance with the standard Dublin Core in the default metadata registry can create a problem for the portability of data to/from of your repository.
2. If using existing metadata fields, avoid any that are automatically managed by DSpace. For example, fields like "date.issued" or "date.accessioned" are normally automatically assigned, and thus must not be recruited for embargo use.
3. Do not place the field for "lift date" in submission screens. This can potentially confuse submitters because they may feel that they can directly assign values to it. As noted in the life-cycle above, this is erroneous: the lift date gets assigned by the embargo system based on the terms. Any pre-existing value will be over-written. But see next recommendation for an exception.
4. As the life-cycle discussion above makes clear, after the terms are applied, that field is no longer actionable in the embargo system. Conversely, the "lift date" field is not actionable **until** the application. Thus you may want to consider configuring both the "terms" and "lift date" to use the same metadata field. In this way, during workflow you would see only the terms, and after item installation, only the lift date. If you wish the metadata to retain the terms for any reason, use 2 distinct fields instead.

Operation

After the fields defined for terms and lift date have been assigned in dspace.cfg, and created and configured wherever they will be used, you can begin to embargo items simply by entering data (dates, if using the default setter) in the terms field. They will automatically be embargoed as they exit workflow, and that the computed lift date will be stored on the ResourcePolicy

Extending embargo functionality

The embargo system supplies a default "interpreter/imposition" class (the "Setter") .

Setter

The default setter recognizes only two expressions of terms: either a literal, non-relative date in the fixed format "yyyy-mm-dd" (known as ISO 8601), or a special string used for open-ended embargo (the default configured value for this is "forever", but this can be changed in dspace.cfg to "toujours", "unendlich", etc). It will perform a minimal sanity check that the date is not in the past. Similarly, the default setter will only remove all read policies as noted above, rather than applying more nuanced rules (e.g allow access to certain IP groups, deny the rest). Fortunately, the setter class itself is configurable and you can "plug in" any behavior you like, provided it is written in java and conforms to the setter interface. The dspace.cfg property:

```
# implementation of embargo setter plugin - replace with local implementation if applicable
plugin.single.org.dspace.embargo.EmbargoSetter = org.dspace.embargo.DefaultEmbargoSetter
```

controls which setter to use.

Lifter

⚠ DEPRECATED: The Lifter is no longer used in the DSpace API, and is not recommended to utilize. Embargo lift dates are now stored on ResourcePolicies and, as such, are "lifted" automatically when the embargo date passes. Manually running a "lifter" may bypass this automatic functionality and result in unexpected results.

The default lifter behavior as described above - essentially applying the collection policy rules to the item - might also not be sufficient for all purposes. It also can be replaced with another class:

```
implementation of embargo lifter plugin - - replace with local implementation if applicable
plugin.single.org.dspace.embargo.EmbargoLifter = org.dspace.embargo.DefaultEmbargoLifter
```

4.1.2.6 Pre-3.0 Embargo Lifter Commands

⚠ **DEPRECATED - Not recommended to use**

The old "embargo-lifter" command is no longer necessary to run. All Embargoes in DSpace are now stored on ResourcePolicies and are lifted automatically after the lift date passed. See Embargo(see page 113) documentation for more information.

Continuing to run the "embargo-lifter" is not recommended and this feature will be removed entirely in a future DSpace release.

If you have implemented the pre DSpace 3.0 Embargo(see page 113) feature, you will need to run it periodically to check for Items with expired embargoes and lift them.

Command used:	`[dspace]/bin/dspace embargo-lifter`

Java class:	org.dspace.embargo.EmbargoManager
Arguments short and (long) forms):	Description
`-c` or `--check`	ONLY check the state of embargoed Items, do NOT lift any embargoes
`-i` or `--identifier`	Process ONLY this handle identifier(s), which must be an Item. Can be repeated.
`-l` or `--lift`	Only lift embargoes, do NOT check the state of any embargoed items.
`-n` or `--dryrun`	Do no change anything in the data model, print message instead.
`-v` or `--verbose`	Print a line describing the action taken for each embargoed item found.
`-q` or `--quiet`	No output except upon error.
`-h` or `--help`	Display brief help screen.

You must run the Embargo Lifter task periodically to check for items with expired embargoes and lift them from being embargoed. For example, to check the status, at the CLI:

```
[dspace]/bin/dspace embargo-lifter -c
```

To lift the actual embargoes on those items that meet the time criteria, at the CLI:

```
[dspace]/bin/dspace embargo-lifter -l
```

4.1.3 Managing User Accounts

When a user registers an account for the purpose of subscribing to change notices, submitting content, or the like, DSpace creates an EPerson record in the database. Administrators can manipulate these records in several ways.

4.1.3.1 From the browser

- Login as an Administrator
- Sidemenu "Access Control" → "People"
- Browse or search for the account you wish to modify or delete.

To modify user permissions / group memberships:

- Login as an Administrator
- Sidemenu "Access Control" → "Groups"
- Edit the Group
- Search for the EPerson & add/remove them from that group.

4.1.3.2 From the command line

The `user` command

The `dspace user` command adds, lists, modifies, and deletes EPerson records.

To create a new user account:

```
[dspace]/bin/dspace user --add --email jquser@example.com -g John -s User --password hiddensecret
[dspace]/bin/dspace user --add --netid jquser --telephone 555-555-1234 --password hiddensecret
```

One of the options `--email` or `--netid` is required to name the record. The complete options are:

-a	--add	required
-m	--email	email address
-n	--netid	"netid" (a username in an external system such as a directory – see Authentication Methods for details)
-p	--password	a password for the account. Required.
-g	--givenname	First or given name
-s	--surname	Last or surname
-t	--telephone	Telephone number
-l	--language	Preferred language
-c	--requireCertificate	Certificate required? See X.509 Authentication(see page 87) for details.

To list accounts:

```
[dspace]/bin/dspace user --list
```

This simply lists some characteristics of each EPerson.

short	long	meaning
-L	--list	required

To modify an account:

```
[dspace]/bin/dspace user --modify -m george@example.com
```

short	long	meaning
-M	--modify	required
-m	--email	identify the account by email address
-n	--netid	identify the account by netid
-g	--givenname	First or given name
-s	--surname	Last or surname
-t	--telephone	telephone number
-l	--language	preferred language
-c	--requireCertificate	certificate required?
-C	--canLogIn	is the account enabled or disabled?
-i	--newEmail	set or change email address
-I	--newNetid	set or change netid

To delete an account:

```
[dspace]/bin/dspace user --delete -n martha
```

short	long	meaning
-d	--delete	required
-m	--email	identify the account by email address
-n	--netid	identify the account by netid

The Groomer

This tool inspects all user accounts for several conditions.

short	long	meaning
-a	--aging	find accounts not logged in since a given date
-u	--unsalted	find accounts not using salted password hashes
-b	--before	date cutoff for --aging
-d	--delete	delete disused accounts (used with --aging)

Find accounts with unsalted passwords

Earlier versions of DSpace used an "unsalted hash" method to protect user passwords. Recent versions use a salted hash. You can find accounts which have never been converted to salted hashing:

Discovering accounts with unsalted password hashes

```
[DSpace]/bin/dsrun org.dspace.eperson.Groomer -u
```

The output is a list of email addresses for matching accounts.

Find (and perhaps delete) disused accounts

You can list accounts which have not logged on since a given date:

Discovering disused accounts

```
[DSpace]/bin/dsrun org.dspace.eperson.Groomer -a -b 07/20/1969
```

The output is a tab-separated-value table of the EPerson ID, last login date, email address, netid, and full name for each matching account.

You can also have the tool delete matching accounts:

Deleting disused accounts

```
[DSpace]/bin/dsrun org.dspace.eperson.Groomer -a -b 07/20/1969 -d
```

4.1.3.3 Email Subscriptions

Introduction

Registered users can subscribe to collections in DSpace. After subscribing, users will receive a daily email containing the new and modified items in the collections they are subscribed to.

DSpace 7.0 does not yet support

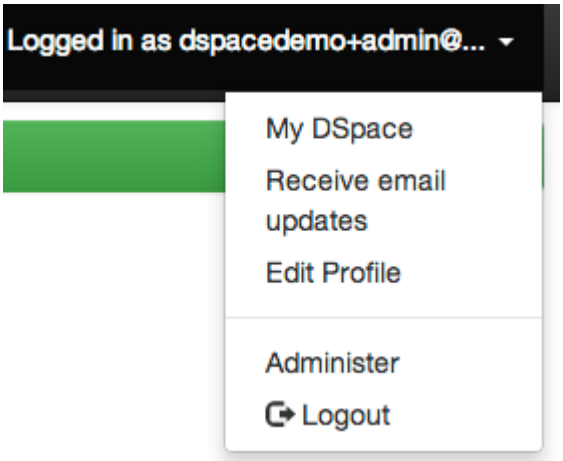
Email Subscriptions are not available in DSpace 7.0. They are scheduled to be restored in a later 7.x release (currently 7.2), see DSpace Release 7.0 Status[176]

Adding new subscriptions

Adding new subscriptions is only available to users who are logged in.

In the User interface, new subscriptions are added on the users Profile page.

In the JSP User Interface, a specific dialog "Receive Email Updates" is available from the dropdown in the top right corner.



176 https://wiki.lyrasis.org/display/DSPACE/DSpace+Release+7.0+Status

System configuration for sending out daily emails

To send out the subscription emails you need to invoke the **sub-daily** script from the DSpace command launcher. It is advised to setup this script as a scheduled task using cron(see page 125).

This script can be run with a parameter -t for testing purposes. When this parameter is passed, the log level is set to DEBUG to ensure that more diagnostic information will be added to the dspace logfile.

4.1.4 Request a Copy

4.1.4.1 Introduction

DSpace 7.0 does not yet support

Email Subscriptions are not available in DSpace 7.0. They are scheduled to be restored in a later 7.x release (currently 7.1), see DSpace Release 7.0 Status[177]

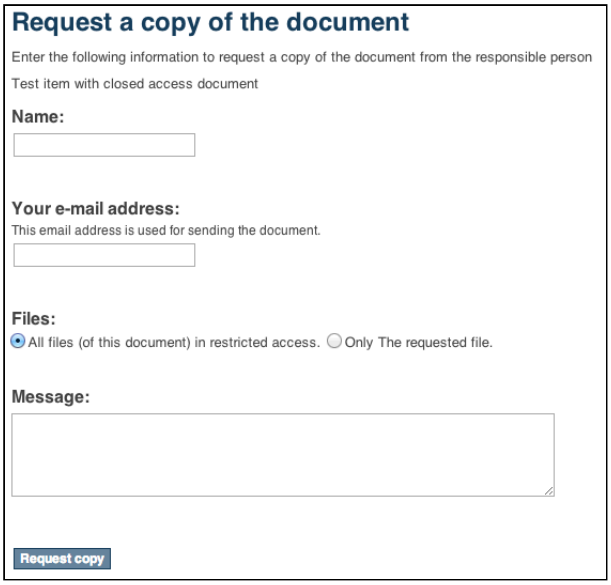
The request a copy functionality was added to DSpace as a measure to facilitate access in those cases when uploaded content can not be openly shared with the entire world immediately after submission into DSpace. It gives users an efficient way to request access to the original submitter of the item, who can approve this access with the click of a button. This practice complies with most applicable policies as the submitter interacts directly with the requester on a case by case basis.

4.1.4.2 Requesting a copy using the User Interface

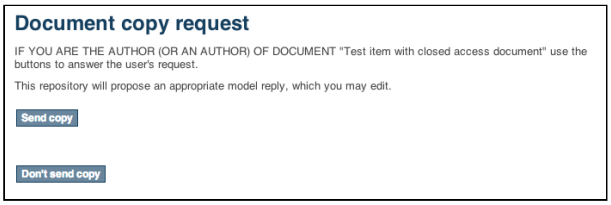
Users can request a copy by clicking the file thumbnail or the blue lock symbol displayed on files that are restricted to them.

177 https://wiki.lyrasis.org/display/DSPACE/DSpace+Release+7.0+Status

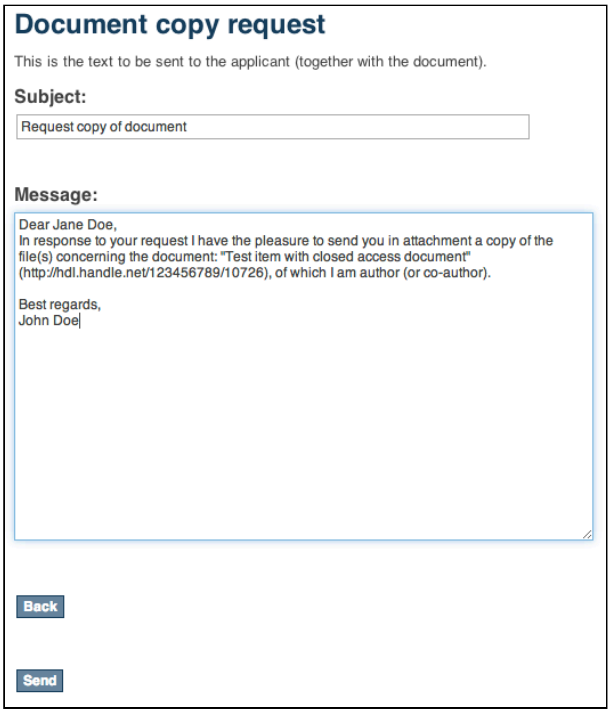
The request form asks the user for his or her name, email address and message where the reason for requesting access can be entered.



After clicking request copy at the bottom of this form, the original submitter of the item will receive an email containing the details of the request. The email also contains a link with a token that brings the original submitter to a page where he or she can either grant or reject access. If the original submitter can not evaluate the request, he or she can forward this email to the right person, who can use the link containing the token without having to log into DSpace.

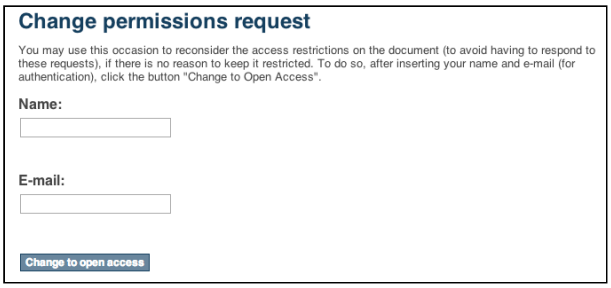


Each of these buttons registers the choice of the submitter, displaying the following form in which an additional reason for granting or rejecting the access can be added.



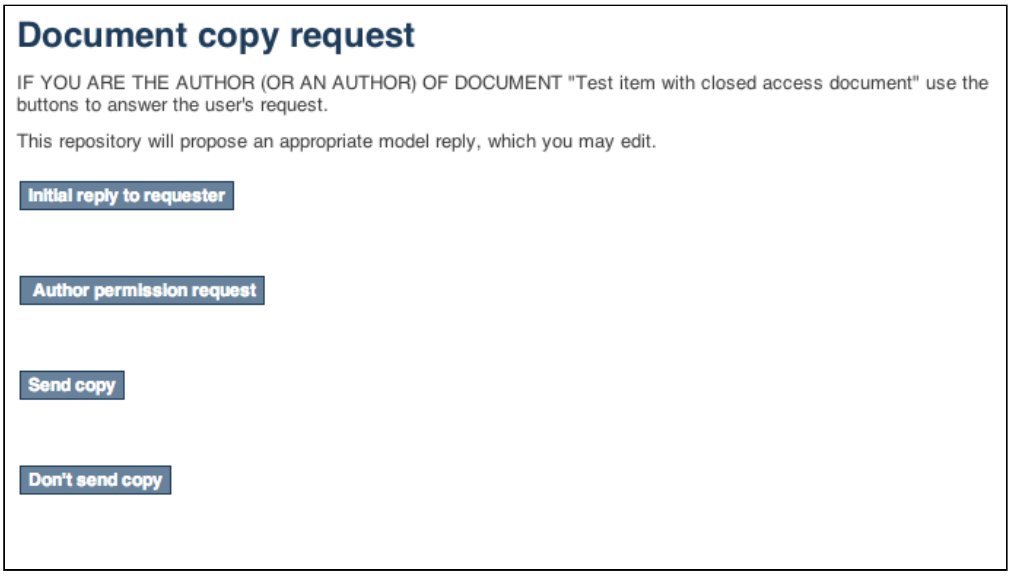
After hitting send, the contents of this form will be sent together with the associated files to the email address of the requester. In case the access is rejected, only the reason will be sent to the requester.

After responding positively to a request for copy, the person who approved is presented with an optional form to ask the repository administrator to alter the access rights of the item, allowing unrestricted open access to everyone.

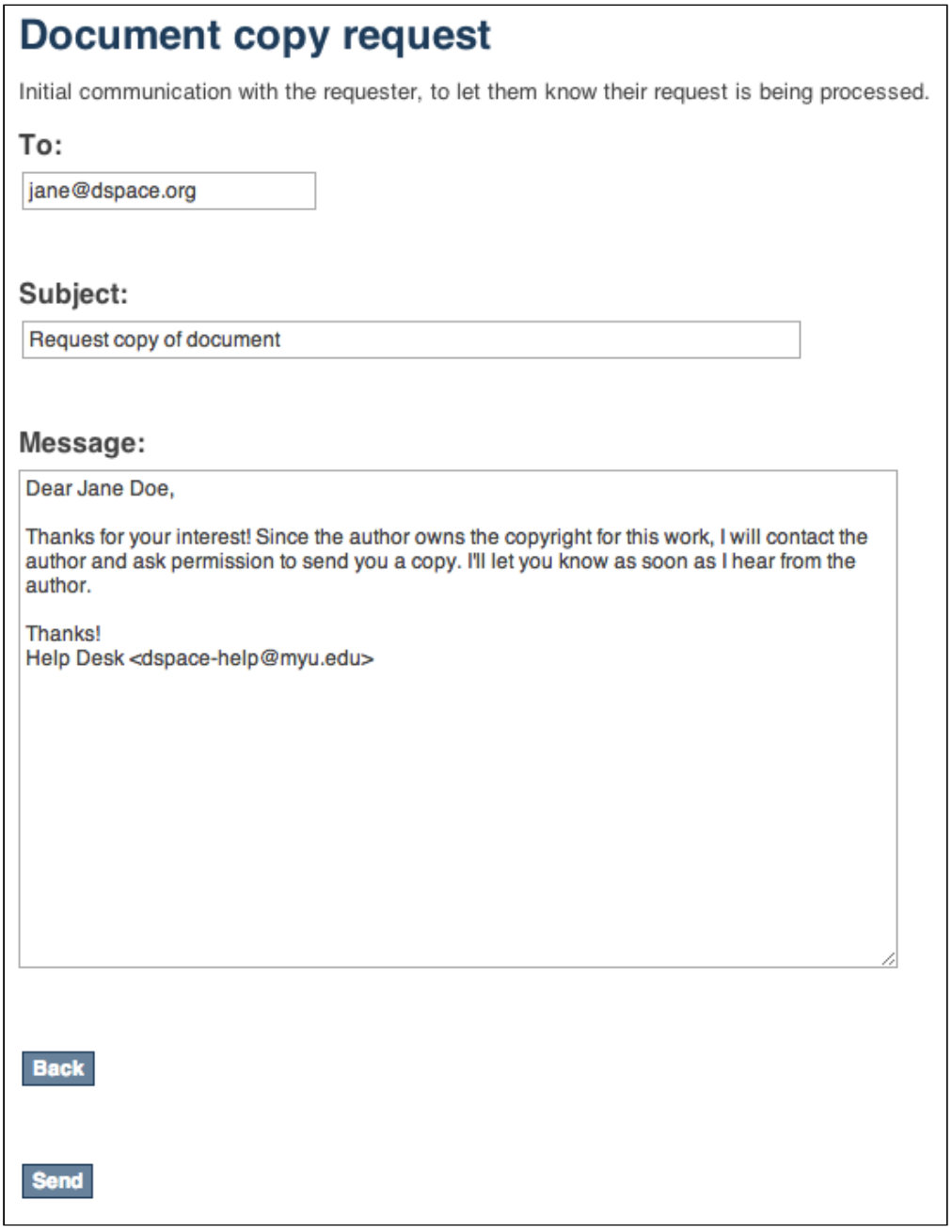


4.1.4.3 (Optional) Requesting a copy with Help Desk workflow

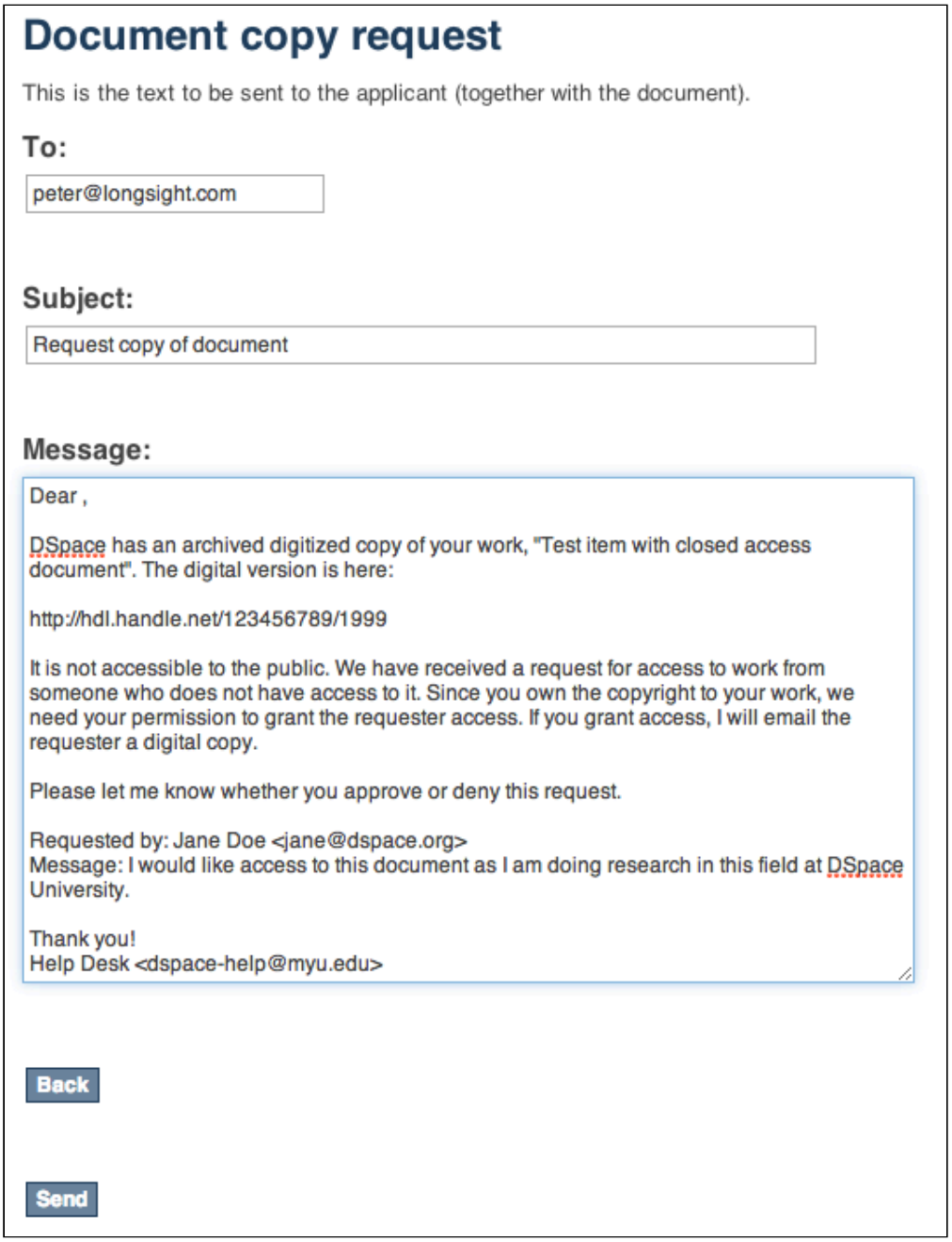
(Optional) Request Item with HelpDesk intermediary, is steered towards having your Repository Support staff act as a helpdesk that receives all incoming RequestItem requests, and then processes them. This adds the options of "Initial Reply to Requestor" to let the requestor know that their request is being worked on, and an option "Author Permission Request" which allows the helpdesk to email the author of the document, as not all documents are deposited by the author, or the author will need to be tracked down by a support staff, as DSpace might not have their current email address.



Initial Reply to Requester



Author permission request, includes information about the original request (requester name, requester email, requester's reason for requesting). The author/submitter's name and email address will be pre-populated in the form from the submitter, but the email address and author name are editable, as the submitter's of content to DSpace aren't always the author.



4.1.4.4 Email templates

Most of the email templates used by Request a Copy are treated just like other email templates in DSpace. The templates can be found in the /config/emails directory and can be altered just by changing the contents and restarting tomcat.

request_item.admin	template for the message that will be sent to the administrator of the repository, after the original submitter requests to have the permissions changed for this item.
request_item.author	template for the message that will be sent to the original submitter of an item with the request for copy.

The templates for emails that the requester receives, that could have been customized by the approver in the aforementioned dialog are not managed as separate email template files. These defaults are stored in the Messages.properties file under the keys

itemRequest.response.body.approve	Default message for informing the requester of the approval
itemRequest.response.body.reject	Default message for informing the requester of the rejection
itemRequest.response.body.contactAuthor	Default message for the helpdesk to contact the author
itemRequest.response.body.contactRequester	Default message for the helpdesk to contact the requester

4.1.4.5 Configuration parameters

Request a copy is enabled by default. Only two configuration parameters in dspace.cfg relate to Request a Copy:

Property:	`request.item.type`
Example Value	`request.item.type = all`
Informational Note	This parameter manages who can file a request for an item. The parameter is optional. When it is empty or commented out, request a copy is disabled across the entire repository. When set to **all**, any user can file a request for a copy. When set to **logged**, only registered users can file a request for copy.
Property:	`mail.helpdesk`
Example Value	`mail.helpdesk = foo@bar.com`

Informational Note	The email address assigned to this parameter will receive the emails both for granting or rejecting request a copy requests, as well as requests to change item policies. This parameter is optional. If it is empty or commented out, it will default to `mail.admin`. *WARNING:* This setting is only utilized if the `RequestItemHelpdeskStrategy` bean is enabled in `[dspace]/config/spring/api/requestitem.xml` (see below)
Property:	`request.item.helpdesk.override`
Example Value	`request.item.helpdesk.override = true`
Informational Note	Should all Request Copy emails go to the `mail.helpdesk` instead of the item submitter? Default is `false`, which sends Item Requests to the item submitter. *WARNING:* This setting is only utilized if the `RequestItemHelpdeskStrategy` bean is enabled in `[dspace]/config/spring/api/requestitem.xml` (see below)

4.1.4.6 Selecting Request a Copy strategy via Spring Configuration

The process that DSpace uses to determine who is the recipient of the Item Request is configurable in this Spring file: `[dspace]/config/spring/api/requestitem.xml`

By default the `RequestItemMetadataStrategy` is enabled, but falls back to the Item Submitter eperson's name and email. You can configure the `RequestItemMetadataStrategy` to load the author's name and email address if you set that information into an item metadata field. For example:

```
<bean class="org.dspace.app.requestitem.RequestItemMetadataStrategy"
 id="org.dspace.app.requestitem.RequestItemAuthorExtractor">
 <!--
 Uncomment these properties if you want lookup in metadata the email and the name of the author to contact
for request copy.
 If you don't configure that or if the requested item doesn't have these metadata the submitter data are
used as fail over

 <property name="emailMetadata" value="schema.element.qualifier" />
 <property name="fullNameMatadata" value="schema.element.qualifier" />

 -->
 </bean>
```

Another common request strategy is the use a single Helpdesk email address to receive all of these requests (see corresponding helpdesk configs in dspace.cfg above). If you wish to use the Helpdesk Strategy, you must first comment out the default `RequestItemMetadataStrategy`, bean and uncomment this bean:

```
<!-- HelpDesk to instead get RequestItem emails-->
<bean class="org.dspace.app.requestitem.RequestItemHelpdeskStrategy"
        id="org.dspace.app.requestitem.RequestItemAuthorExtractor"></bean>
```

4.2 Configurable Entities

4.2.1 Introduction

DSpace users have expressed the need for DSpace to be able to provide more support for different types of digital objects related to open access publications, such as authors/author profiles, data sets etc. Configurable Entities are designed to meet that need.

In DSpace, **an Entity is a special type of Item which often has Relationships to other Entities**. Breaking it down with more details...

- **Entity:** Every Entity is an Item.
  - This means they must belong to a Collection, just like a normal Item. (Community & Collection objects are unchanged and unaffected by Entities.)
  - Normal Items are still the "default" Item, and they are unchanged. So, not every Item is an Entity.
  - Because Entities are all Items, they are immediately usable in submission/workflow process, batch import/export, OAI-PMH, etc.
- **Entity (or Item) Type:** Entities all have a "dspace.entity.type" metadata field which defines their Entity/Item "type". For example, this type may be "Person", "Project", "Publication", "Journal", etc. It's highly visible within the User Interface as a label.
- **Relationships:** Based on that "type", an Entity may be related to other Entities via a Relationship. One Entity type may support several relationship types at once. Examples of relationship types include "isPersonOfProject" or "isPublicationOfAuthor". These relationship types are named based on the Entity "type" (as you can likely tell). Relationships also appear on Entities as metadata using the "relation" schema.
- **Virtual Metadata:** Entities of different types may also have customized visualizations in the User Interface. These visualizations may also dynamically pull in metadata from related Entities. For example, a Publication entity may be displayed in the User Interface with an author name dynamically pulled in from a related Person entity. The metadata "appears" as though it is part of the Entity you are viewing, but it is dynamically pulled via the Relationship.

Entities and their Relationships are also completely configurable. DSpace provides some sample models out of the box, which you can use directly or adapt as needed.

The Entity model also has similarities with the Portland Common Data Model (PCDM)[178], with an Entity roughly mapping to a "pcdm:Object" and existing Communities and Collections roughly mapping to a "pcdm:Collection". However, at this time DSpace Entities concentrate more on building a graph structure of relationships, instead of a tree structure.

4.2.2 Default Entity Models

DSpace currently comes with the following Entity models, both of which are defined in `[dspace]/config/entities/relationship-types.xml`. These Entity models are not used by default, but may be enabled as described below.

4.2.2.1 Research Entities

Research Entities include Person, OrgUnit, Project and Publication. They allow you to create author profiles (Person) in DSpace, and relate those people to their department(s) (OrgUnit), grant project(s) (Project) and works (Publication).

178 https://github.com/duraspace/pcdm/wiki

- Each publication can link to projects, people and org units
- Each person can link to projects, publications and org units
- Each project can link to publications, people and org units
- Each org units can link to projects, people and publications

4.2.2.2 Journals

Journal Entities include Journal, Journal Volume, Journal Issue and Publication (article). They allow you to represent a Journal hierarchy more easily within DSpace, starting at the overall Journal, consisting of multiple Volumes, and each Volume containing multiple Issues. Issues then link to all articles (Publication) which were a part of that journal issue.

NOTE: that this model includes the same "Publication" entity as the Research Entities model described above. This Entity overlap allows you to link an article (Publication) both to its author (Person) as well as the Journal Issue it appeared in.

4.2.3 Enabling Entities

By default, Entities are not used in DSpace. But, as described above several models are available out-of-the-box that may be optionally enabled.

⚠ Keep in mind, there are a few DSpace import/export features that do not yet support Entities in DSpace 7.0. These will be coming in future 7.x releases. See DSpace Release 7.0 Status[179] for prioritization information, etc.

- AIP Backup and Restore(see page 411) does not fully support entity types or relationships. In other words, Entities are only represented as normal Items in AIPs
- Importing and Exporting Items via Simple Archive Format[180] does not fully support entity types or relationships. In other words, Entities are only represented as normal Items in SAF. (Note: early work to bring this support is already begun in https://github.com/DSpace/DSpace/pull/3322)
- SWORDv1 Server(see page 216) and SWORDv2 Server(see page 202) does not yet support Entity or relationship creation.

179 https://wiki.lyrasis.org/display/DSPACE/DSpace+Release+7.0+Status
180 https://wiki.lyrasis.org/display/DSDOC6x/Importing+and+Exporting+Items+via+Simple+Archive+Format

4.2.3.1 1. Configure your entity model (optionally)

As described above, DSpace provides two default entity models defined in `[dspace]/config/entities/relationship-types.xml`. These models may be used as-is, or modified.

You can also design your own model from scratch (see "Designing your own model" section below). So, feel free to start by modifying `relationship-types.xml`, or creating your own model based on the `relationship-types.dtd`.

4.2.3.2 2. Import entity model into the database

In order to enable a defined entity model, it MUST be imported into the DSpace database This is achieved by using the "initialize-entities" script. The example below will import the "out-of-the-box" entity models into your DSpace installation

```
# The -f command requires a full path to an Entities model configuration file.
[dspace]/bin/dspace initialize-entities -f [dspace]/config/entities/relationship-types.xml
```

If an Entity (of same type name) already exists, it will be updated with any new relationships defined in relationship-types.xml

If an Entity (of same type name) doesn't exist, the new Entity type will be created along with its relationships defined in relationship-types.xml

Once imported into the Database, the overall structure is as follows:

- All valid Entity Types are stored in the "entity_type" database table.
- All Relationship type definitions are stored in the "relationship_type" database table
- All Relationships between entities get stored in the "relationship" table.
- Entities themselves are stored alongside Items in the 'item' table. Every Entity must have a "dspace.entity.type" metadata field whose value is a valid Entity Type (from the "entity_type" table).

Keep in mind, your currently enabled Entity model is defined in your database, and NOT in the "relationship-types.xml". Anytime you want to update your data model, you'd update/create a configuration (like relationship-types.xml) and re-run the "initialize-entities" command.

4.2.3.3 3. Configuration of community/collection list for Entity types

Because all Entities are Items, they MUST belong to a Collection. Therefore, the easiest way to create a different submission forms per Entity type (e.g. Person, Project, Journal, Publication, etc) is to ensure you create a Collection for each Entity Type (as each Collection can have a custom Submission Form).

1. Create at least one Collection for each Entity Type needing a custom Submission form. For example, a Collection for "Person" entities, and a separate one for "Publication" entities.
2. Edit the Collection, and create a Template Item (which is used for all Entities/Items submitted to that Collection) from the "Edit Metadata" tab
    a. In the Template Item, add a single metadata field "dspace.entity.type". Give it a value matching the Entity type (e.g. Publication, Person, Project, OrgUnit, Journal, JournalVolume, JournalIssue). This value IS CASE SENSITIVE and it MUST match the Entity type name defined in relationship-types.xml
        i. This single metadata field will ensure that every Item submitted to this collection is automatically assigned that Entity type. So, it ties this Collection to that type of Entity.

3. In the Edit Collection page, switch to the "Assign Roles" tab and create a "Submitters" group. Add any people who should be allowed to submit/create this new Entity type.
    a. If you only want Administrators to create this Entity type, you can skip this step. Administrators can submit to any Collection.
4. If you want to hide this Collection, you can choose to only make it visible to that same Submitters group (or Administrators). This does NOT hide the Entities from search or browse, but it will hide the Collection itself.
    a. In the Edit Collection page, switch to the "Authorizations" tab.
    b. Add a new Authorization of TYPE_CUSTOM, restricting "READ" to the Submitters group created above (or Administrators if there is no Submitters group). You can also add multiple READ policies as needed. WARNING: The Submitters group MUST have READ privileges to be able to submit/create new Entities.
    c. Remove the default READ policy giving Anonymous permissions.
    d. Assuming you want the Entities to still be publicly available, make sure the DEFAULT_ITEM_READ policy is set to "Anonymous"!

Obviously, how you organize your Entity Types into Collections is up to you. You can create a single Collection for all Entities of that type (e.g. an "Author Profiles" collection could be where all "Person" Entities are submitted/stored). Or, you could create many Collections for each Entity Type (e.g. each Department in your University may have it's own Community, and underneath have a "Staff Profiles" Collection where all "Person" Entities for that department are submitted/stored). A few example structures are shown below.

Example Structure based on the departments:

- Department of Architecture
- Building Technology Program
- Theses - Department of Architecture
- Department of Biology
- Theses - Biology
- People
- Projects

OR

- Department of Architecture
- Building Technology Program
- Theses - Department of Architecture
- People in Department of Architecture
- Projects in Department of Architecture
- Department of Biology
- Theses - Biology
- People in Department of Biology
- Projects in Department of Biology

Example Structure based on the publication type:

- Books
- Book Chapter
- Edited Volume
- Monograph
- Theses
- Bachelor Thesis

- Doctoral Thesis
- Habilitation Thesis
- Master Thesis

- People
- Projects

4.2.3.4 4. Configure Submission Forms for each Entity type

You should have already created Entity-specific Collections in the previous step. Now, we just need to map those Collections to Submission processes specific to each Entity.

On the backend, you will now need to modify the `[dspace]/config/item-submission.xml` to "map" this Collection (or Collections) to the submission process for this Entity type.

- DSpace comes with sample submission forms for each Entity type.
  - The sample <submission-process> is defined in `item-submission.xml` and named based on the Entity type (e.g. Publication, Person, Project, etc).
  - The metadata fields captured for each Entity are defined in a custom step in `submission-forms.xml`, and named in the format "[entityType]Step" (where the entity type is camelcased). For example: "publicationStep", "personStep", "projectStep".
- Optionally, modify those sample submission forms. See Submission User Interface(see page 260) for hints/tips on customizing the `item-submission.xml` or `submission-forms.xml` files
- Now, in `item-submission.xml`, map your Collection's handle (findable on the Collection homepage) to the submission form you want it to use. In the below example, we've mapped a single Collection to each of the out-of-the-box Entity types.

```
<name-map collection-handle="123456789/5" submission-name="Publication"/>
<name-map collection-handle="123456789/6" submission-name="Person"/>
<name-map collection-handle="123456789/7" submission-name="Project"/>
<name-map collection-handle="123456789/8" submission-name="OrgUnit"/>
<name-map collection-handle="123456789/28" submission-name="Journal"/>
<name-map collection-handle="123456789/29" submission-name="JournalVolume"/>
<name-map collection-handle="123456789/30" submission-name="JournalIssue"/>
```

Once your modifications to the submission process are complete, you will need to quickly reboot Tomcat (or your servlet container) to reload the current settings.

4.2.3.5 5. Configure Workflow for each Entity type (optionally)

The DSpace workflow can be used for reviewing all objects in the Object Model since these objects are all Items, and separate collections can be used. The workflow used for e.g. a Person Object can be configured to be identical to a publication, different from a publication, or use no workflow at all.

See Configurable Workflow(see page 250) for more information on configuring workflows per Collection.

4.2.3.6 6. Configure Virtual Metadata to display for related Entities (optionally)

"Virtual Metadata" is metadata that is dynamically determined (at the time of access) based on an Entity's relationship to other Entities. A basic example is displaying a Person Entity's name in the "dc.contributor.author" field of a related Publication Entity. That "dc.contributor.author" field doesn't actually exist on the Publication, but

is dynamically added as "virtual metadata" simply because the Publication is linked to the Person (via a relationship).

Virtual Metadata is configurable for all Entities and all relationships. DSpace comes with default settings for its default Entity model, and those can be found in `[dspace]/config/spring/api/virtual-metadata.xml`. In that Spring Bean configuration file, you'll find a map of each relationship type to a metadata field & its value. Here's a summary of how it works:

- The "org.dspace.content.virtual.VirtualMetadataPopulator" bean maps every Relationship type (from `relationship-types.xml`) to a <util:map> definition (of a given ID) also in the `virtual-metadata.xml`

```
<!-- For example, the isAuthorOfPublication relationship is linked to a map of ID
"isAuthorOfPublicationMap" -->
<entry key="isAuthorOfPublication" value-ref="isAuthorOfPublicationMap"/>
```

- That <util:map> defintion defines which DSpace metadata field will store the virtual metadata. It also links to the bean which will dynamically define the value of this metadata field.

```
<!-- In this example, isAuthorOfPublication will be displayed in the "dc.contributor.author" field
-->
<!-- The *value* of that field will be defined by the "publicationAuthor_author" bean -->
<util:map id="isAuthorOfPublicationMap">
    <entry key="dc.contributor.author" value-ref="publicationAuthor_author"/>
</util:map>
```

- A bean of that ID then defines the value of the field, based on the related Entity. In this example, these fields are pulled from the related Person entity and concatenated. If the Person has "person.familyName=Jones" and "person.givenName=Jane", then the value of "dc.contributor.author" on the related Publication will be dynamically set to "Jones, Jane.

```
<bean class="org.dspace.content.virtual.Concatenate" id="publicationAuthor_author">
   <property name="fields">
       <util:list>
           <value>person.familyName</value>
           <value>person.givenName</value>
           <value>organization.legalName</value>
       </util:list>
   </property>
   <property name="separator">
       <value>, </value>
   </property>
   <property name="useForPlace" value="true"/>
   <property name="populateWithNameVariant" value="true"/>
</bean>
```

If the default Virtual Metadata looks good to you, no changes are needed. If you make any changes, be sure to restart Tomcat to update the bean definitions.

4.2.4 Designing your own Entity model

When using a different entities model, the new model has to be configured an loaded into your repository

4.2.4.1 Thinking about the object model

First step: identify the entity types

- Which types of objects would you want to create items for: e.g. Person, Publication, JournalVolume
- Be careful not to confuse a type with a relationship. A Person is a type, an author is a relationship between the publication and the person

Second step: identify the relationship types

- Which relationship types would you want to create between the entity items from the previous step: e.g. isAuthorOfPublication, isEditorOfPublication, isProjectOfPublication, isOrgUnitOfPerson, isJournalIssueOfPublication
- Multiple relationships between the same 2 types can be created: isAuthorOfPublication, isEditorOfPublication
- Relationships are automatically bidirectional, so no need to worry about whether you want to display the authors in a publication or the publications of an author

Third step: visualize your model

- By creating a drawing of your model, you'll be able to quickly verify whether anything is missing

4.2.4.2 Configuring the object model

Configure the model in relationship-types.xml

- Similar to the default relationship-types.xml[181], configure a relationship type per connection between 2 entity types
- Include the 2 entity type names which are being connected.
- Determine a clear an unambiguous name for the relation in both directions
- Optionally: determine the cardinality (min/max occurrences) for the relationships
- Optionally: determine default behavior for copying metadata if the relationship is deleted

181 https://github.com/DSpace/DSpace/blob/main/dspace/config/entities/relationship-types.xml

4.2.4.3 Configuring the metadata fields

Determining the metadata fields to use

- Dublin Core works for publications, but not for a Person, JournalVolume, …
- There are many standards which can be easily configured: schema.org[182], eurocris, datacite, …
- Pick a schema which suits your needs

Configure the submission forms

- Add a form in submission-forms.xml[183] for each entity type, containing the relevant metadata fields
- See also Submission User Interface(see page 260) documentation.
- Configure which relationships to create[184]

4.2.4.4 Configuring the item display pages

- The metadata configuration is not specific to configurable entities.
- Similar to other customizations to the item display pages, configure in Angular which metadata fields to display and their label. A template per entity type can be created
- The relationship display is similar to the metadata configuration
- Similar to the metadata configuration: configure in Angular which relationship to display and their label

4.2.4.5 Configuring virtual metadata

- The isAuthorOfPublication relationship can be displayed for the Publication item as dc.contributor.author
- The isOrgUnitOfPerson relationship can be displayed for the Person item as organization.legalName
- This can be configured in virtual-metadata.xml[185]

4.2.4.6 Configuring discovery

- Configure the discovery facets, filters, sort options, …
- The facets for a Person can be job title, organization, project, …
- The filters for a Person can be person.familyName, person.givenName, …

4.2.4.7 Additional Technical Details

The original Entities design document is available in Google Docs at: https://docs.google.com/document/d/1wEmHirFzrY3qgGtRr2YBQwGOvH1IuTVGmxDIdnqvwxM/edit

We are working on pulling that information into this Wiki space as a final home, but currently some technical details exist only in that document.

A talk on Configurable Entities was also presented at DSpace 7 at OR2021[186]

182 http://schema.org
183 https://github.com/DSpace/DSpace/blob/master/dspace/config/submission-forms.xml
184 https://docs.google.com/document/d/1X0XsppZYOtPtbmq7yXwmu7FbMAfLxxOCONbw0_rl7jY/edit#heading=h.5bu9shx0j942
185 https://github.com/DSpace/DSpace/blob/master/dspace/config/spring/api/virtual-metadata.xml
186 https://wiki.lyrasis.org/display/DSPACE/DSpace+7+at+OR2021

4.3 Curation System

DSpace supports running curation tasks, which are described in this section. DSpace includes several useful tasks out-of-the-box, but the system also is designed to allow new tasks to be added between releases, both general purpose tasks that come from the community, and locally written(see page 539) and deployed tasks.

4.3.1 Tasks

The goal of the curation system ("CS") is to provide a simple, extensible way to manage routine content operations on a repository. These operations are known to CS as "tasks", and they can operate on any DSpaceObject (i.e. subclasses of DSpaceObject) - which means the entire Site, Communities, Collections, and Items - viz. core data model objects. Tasks may elect to work on only one type of DSpace object - typically an Item - and in this case they may simply ignore other data types (tasks have the ability to "skip" objects for any reason). The DSpace core distribution will provide a number of useful tasks, but the system is designed to encourage local extension - tasks can be written for any purpose, and placed in any java package. This gives DSpace sites the ability to customize the behavior of their repository without having to alter - and therefore manage synchronization with - the DSpace source code. What sorts of activities are appropriate for tasks?

Some examples:

- apply a virus scan to item bitstreams (this will be our example below)
- profile a collection based on format types - good for identifying format migrations
- ensure a given set of metadata fields are present in every item, or even that they have particular values
- call a network service to enhance/replace/normalize an item's metadata or content
- ensure all item bitstreams are readable and their checksums agree with the ingest values

Since tasks have access to, and can modify, DSpace content, performing tasks is considered an administrative function to be available only to knowledgeable collection editors, repository administrators, sysadmins, etc. No tasks are exposed in the public interfaces.

4.3.2 Activation

For CS to run a task, the code for the task must of course be included with other deployed code (to `[dspace]/lib`, WAR, etc) but it must also be declared and given a name. This is done via a configuration property in `[dspace]/config/modules/curate.cfg` as follows:

```
### Task Class implementations
plugin.named.org.dspace.curate.CurationTask = org.dspace.ctask.general.NoOpCurationTask = noop
plugin.named.org.dspace.curate.CurationTask = org.dspace.ctask.general.ProfileFormats = profileformats
plugin.named.org.dspace.curate.CurationTask = org.dspace.ctask.general.RequiredMetadata = requiredmetadata
plugin.named.org.dspace.curate.CurationTask = org.dspace.ctask.general.ClamScan = vscan
plugin.named.org.dspace.curate.CurationTask = org.dspace.ctask.general.MicrosoftTranslator = translate
plugin.named.org.dspace.curate.CurationTask = org.dspace.ctask.general.MetadataValueLinkChecker =
checklinks
```

For each activated task, a key-value pair is added. The key is the fully qualified class name and the value is the *taskname* used elsewhere to configure the use of the task, as will be seen below. Note that the curate.cfg configuration file, while in the config directory, is located under "modules". The intent is that tasks, as well as any configuration they require, will be optional "add-ons" to the basic system configuration. Adding or removing tasks has no impact on dspace.cfg.

For many tasks, this activation configuration is all that will be required to use it. But for others, the task needs specific configuration itself. A concrete example is described below, but note that these task-specific configuration property files also reside in `[dspace]/config/modules`

4.3.3 Task Invocation

Tasks are invoked using CS framework classes that manage a few details (to be described below), and this invocation can occur wherever needed, but CS offers great versatility "out of the box":

4.3.3.1 On the command line

A simple tool "CurationCli" provides access to CS via the command line. This tool bears the name "curate" in the DSpace launcher. For example, to perform a virus check on collection "4":

```
[dspace]/bin/dspace curate -t vscan -i 123456789/4
```

The complete list of options:

option	meaning
-t taskname	name of task to perform.
-T filename	name of file containing a list of tasknames to be performed.
-e epersonID	(email address or netid) will be the superuser if unspecified.
-i identifier	ID of object to curate. May be (1) a Handle, (2) a workflow ID, or (3) 'all' to operate on the whole repository.
-q queue	name of queue to process. -i and -q are mutually exclusive.
-l limit	maximum number of objects in Context cache. If absent, unlimited objects may be added.

option	meaning
-s scope	declare a scope for database transactions. Scope must be: (1) 'open' (default value), (2) 'curation' or (3) 'object'.
-v	emit verbose output
-r filename	emit reporting to the named file. '-r -' writes reporting to standard out. If not specified, report is discarded silently.
-p name=value	set a runtime task parameter `name` to the value `value`. May be repeated as needed. See "Task parameters" below.

As with other command-line tools, these invocations could be placed in a cron table and run on a fixed schedule, or run on demand by an administrator.

4.3.3.2 In the admin UI

In the UI, there are several ways to execute configured Curation Tasks:

1. **From the "Curate" tab/button that appears on each "Edit Community/Collection/Item" page:** this tab allows an Administrator, Community Administrator or Collection Administrator to run a Curation Task on that particular Community, Collection or Item. When running a task on a Community or Collection, that task will also execute on all its child objects, unless the Task itself states otherwise (e.g. running a task on a Collection will also run it across all Items within that Collection).
    - NOTE: Community Administrators and Collection Administrators can only run Curation Tasks on the Community or Collection which they administer, along with any child objects of that Community or Collection. For example, a Collection Administrator can run a task on that specific Collection, or on any of the Items within that Collection.
2. **From the Administrator's "Curation Tasks" page:** This option is only available to DSpace Administrators, and appears in the Administrative side-menu. This page allows an Administrator to run a Curation Task across a single object, or all objects within the entire DSpace site.
    - In order to run a task from this interface, you must enter in the handle for the DSpace object. To run a task site-wide, you can use the handle: `[your-handle-prefix]/0`

Each of the above pages exposes a drop-down list of configured tasks, with a button to 'perform' the task, or queue it for later operation (see section below). Not all activated tasks need appear in the Curate tab - you filter them by means of a configuration property. This property also permits you to assign to the task a more user-friendly name than the PluginManager *taskname*. The property resides in `[dspace]/config/modules/curate.cfg`:

```
curate.ui.tasknames = profileformats = Profile Bitstream Formats
curate.ui.tasknames = requiredmetadata = Check for Required Metadata
```

When a task is selected from the drop-down list and performed, the tab displays both a phrase interpreting the "status code" of the task execution, and the "result" message if any has been defined. When the task has been queued, an acknowledgement appears instead. You may configure the words used for status codes in curate.cfg (for clarity, language localization, etc):

```
curate.ui.statusmessages = -3 = Unknown Task
curate.ui.statusmessages = -2 = No Status Set
curate.ui.statusmessages = -1 = Error
curate.ui.statusmessages = 0 = Success
curate.ui.statusmessages = 1 = Fail
curate.ui.statusmessages = 2 = Skip
curate.ui.statusmessages = other = Invalid Status
```

Report output from tasks run in this way is collected by configuring a Reporter plugin. You must have exactly one Reporter configured. The default is to use the FileReporter, which writes a single report of the output of all tasks in the run over all of the selected objects, to a file in the reports directory (configured as report.dir). See `[DSpace]/config/modules/submission-configuration.cfg` for the value of `plugin.single.org`[187]`.dspace.curate.Reporter`. Other Reporter implementations are provided, or you may supply your own.

As the number of tasks configured for a system grows, a simple drop-down list of **all** tasks may become too cluttered or large. DSpace 1.8+ provides a way to address this issue, known as *task groups*. A task group is a simple collection of tasks that the Admin UI will display in a separate drop-down list. You may define as many or as few groups as you please. If no groups are defined, then all tasks that are listed in the *ui.taskname*s property will appear in a single drop-down list. If at least *one* group is defined, then the admin UI will display **two** drop-down lists. The first is the list of task groups, and the second is the list of task names associated with the selected group. A few key points to keep in mind when setting up task groups:

- a task can appear in more than one group if desired
- tasks that belong to no group are *invisible* to the admin UI (but of course available in other contexts of use)

The configuration of groups follows the same simple pattern as tasks, using properties in `[dspace]/config/modules/curate.cfg`. The group is assigned a simple logical name, but also a localizable name that appears in the UI. For example:

```
# ui.taskgroups contains the list of defined groups, together with a pretty name for UI display
curate.ui.taskgroups = replication = Backup and Restoration Tasks
curate.ui.taskgroups = integrity = Metadata Integrity Tasks
.....
# each group membership list is a separate property, whose value is comma-separated list of logical task
names
curate.ui.taskgroup.integrity = profileformats, requiredmetadata
....
```

4.3.3.3 In workflow

CS provides the ability to attach any number of tasks to standard DSpace workflows. Using a configuration file `[dspace]/config/workflow-curation.xml`, you can declaratively (without coding) wire tasks to any step in a workflow. An example:

187 http://plugin.single.org

```
<taskset-map>
    <mapping collection-handle="default" taskset="cautious" />
</taskset-map>
<tasksets>
    <taskset name="cautious">
        <flowstep name="step1">
            <task name="vscan">
                <workflow>reject</workflow>
                <notify on="fail">$flowgroup</notify>
                <notify on="fail">$colladmin</notify>
                <notify on="error">$siteadmin</notify>
            </task>
        </flowstep>
    </taskset>
</tasksets>
```

This markup would cause a virus scan to occur during step one of workflow for any collection, and automatically reject any submissions with infected files. It would further notify (via email) both the reviewers (step 1 group), and the collection administrators, if either of these are defined. If it could not perform the scan, the site administrator would be notified.

The notifications use the same procedures that other workflow notifications do - namely email. There is a new email template defined for curation task use: `[dspace]/config/emails/flowtask_notify`. This may be language-localized or otherwise modified like any other email template.

Tasks wired in this way are normally performed as soon as the workflow step is entered, and the outcome action (defined by the 'workflow' element) immediately follows. It is also possible to delay the performance of the task - which will ensure a responsive system - by queuing the task instead of directly performing it:

```
...
    <taskset name="cautious">
        <flowstep name="step1" queue="workflow">
...
```

This attribute (which must always follow the "name" attribute in the flowstep element), will cause all tasks associated with the step to be placed on the queue named "workflow" (or any queue you wish to use, of course), and further has the effect of **suspending** the workflow. When the queue is emptied (meaning all tasks in it performed), then the workflow is restarted. Each workflow step may be separately configured,

Like configurable submission, you can assign these task rules per collection, as well as having a default for any collection.

As with task invocation from the administrative UI, workflow tasks need to have a Reporter configured in `submission-configuration.cfg`.

4.3.3.4 In arbitrary user code

If these pre-defined ways are not sufficient, you can of course manage curation directly in your code. You would use the CS helper classes. For example:

```
Collection coll = (Collection)HandleManager.resolveToObject(context, "123456789/4");
Curator curator = new Curator();
curator.setReporter(System.out);
curator.addTask("vscan").curate(coll);
System.out.println("Result: " + curator.getResult("vscan"));
```

would do approximately what the command line invocation did. the method "curate" just performs all the tasks configured (you can add multiple tasks to a curator).

The above directs report output to standard out. Any class which implements Appendable may be set as the reporter class.

4.3.4 Asynchronous (Deferred) Operation

Because some tasks may consume a fair amount of time, it may not be desirable to run them in an interactive context. CS provides a simple API and means to defer task execution, by a queuing system. Thus, using the previous example:

```
Curator curator = new Curator();
curator.addTask("vscan").queue(context, "monthly", "123456789/4");
```

would place a request on a named queue "monthly" to virus scan the collection. To read (and process) the queue, we could for example:

```
[dspace]/bin/dspace curate -q monthly
```

use the command-line tool, but we could also read the queue programmatically. Any number of queues can be defined and used as needed.
In the administrative UI curation "widget", there is the ability to both perform a task, but also place it on a queue for later processing.

4.3.5 Task Output and Reporting

Few assumptions are made by CS about what the 'outcome' of a task may be (if any) - it. could e.g. produce a report to a temporary file, it could modify DSpace content silently, etc. But the CS runtime does provide a few pieces of information whenever a task is performed:

4.3.5.1 Status Code

This was mentioned above. This is returned to CS whenever a task is called. The complete list of values:

```
-3 NOTASK  - CS could not find the requested task
-2 UNSET   - task did not return a status code because it has not yet run
-1 ERROR   - task could not be performed
 0 SUCCESS - task performed successfully
 1 FAIL    - task performed, but failed
 2 SKIP    - task not performed due to object not being eligible
```

In the administrative UI, this code is translated into the word or phrase configured by the *ui.statusmessages* property (discussed above) for display.

4.3.5.2 Result String

The task may define a string indicating details of the outcome. This result is displayed, in the "curation widget" described above:

```
"Virus 12312 detected on Bitstream 4 of 1234567789/3"
```

CS does not interpret or assign result strings, the task does it. A task may not assign a result, but the "best practice" for tasks is to assign one whenever possible.

4.3.5.3 Reporting Stream

For very fine-grained information, a task may write to a *reporting* stream. This stream may be sent to a file or to standard out, when running a task from the command line. Tasks run from the administrative UI or a workflow use a configured Reporter class to collect report output. Your own code may collect the report using any implementation of the Appendable interface. Unlike the result string, there is no limit to the amount of data that may be pushed to this stream.

4.3.6 Task Properties

DSpace 1.8 introduces a new "idiom" for tasks that require configuration data. It is available to any task whose implementation extends `AbstractCurationTask`, but is completely optional. There are a number of problems that task properties are designed to solve, but to make the discussion concrete we will start with a particular one: the problem of hard-coded configuration file names. A task that relies on configuration data will typically encode a fixed reference to a configuration file name. For example, the virus scan task reads a file called "`clamav.cfg`", which lives in `[dspace]/config/modules`. It could look up its configuration properties in the ordinary way. But tasks are supposed to be written by anyone in the community and shared around (without prior coordination), so if another task uses the same configuration file name, there is a name **collision** here that can't be easily fixed, since the reference is hard-coded in each task. In this case, if we wanted to use both at a given site, we would have to alter the source of one of them - which introduces needless code localization and maintenance.

Task properties gives us a simple solution. Here is how it works: suppose that both colliding tasks instead use the task properties facility instead of ordinary configuration lookup. For example, each asks for the property `clamav.service.host`. At runtime, the curation system **resolves** this request to a set of configuration properties, and it uses the *name the task has been configured as* as the prefix of the properties. So, for example, if both were installed (in, say, `curate.cfg`) as:

```
org.dspace.ctask.general.ClamAv = vscan,
org.community.ctask.ConflictTask = virusscan,
....
```

then the task property `foo` will resolve to the property named `vscan.foo` when called from ClamAv task, but `virusscan.foo` when called from ConflictTask's code. Note that the "`vscan`" etc are locally assigned names, so we can always prevent the "collisions" mentioned, and we make the tasks much more portable, since we remove the "hard-coding" of config names.

Another use of task properties is to support multiple task profiles. Suppose we have a task that we want to operate in one of two modes. A good example would be a mediafilter task that produces a thumbnail. We can either create one if it doesn't exist, or run with "-force" which will create one regardless. Suppose this behavior was controlled by a property in a config file. If we configured the task as "thumbnail", then we would have in (perhaps) `[dspace]/config/modules/thumbnail.cfg`:

```
...other properties...
thumbnail.thumbnail.maxheight = 80
thumbnail.thumbnail.maxwidth = 80
thumbnail.forceupdate=false
```

The thumbnail generating task code would then resolve "forcedupdate" to see whether filtering should be forced.

But an obvious use-case would be to want to run force mode **and** non-force mode from the admin UI on different occasions. To do this, one would have to stop Tomcat, change the property value in the config file, and restart, etc However, we can use task properties to elegantly rescue us here. All we need to do is go into the config/modules directory, and create a new file perhaps called: `thumbnail.force.cfg`. In this file, we put the properties:

```
thumbnail.force.thumbnail.maxheight = 80
thumbnail.force.thumbnail.maxwidth = 80
thumbnail.force.forceupdate=true
```

Then we add a new task (really just a new name, no new code) in curate.cfg:

```
org.dspace.ctask.general.ThumbnailTask = thumbnail
org.dspace.ctask.general.ThumbnailTask = thumbnail.force
```

Consider what happens: when we perform the task "`thumbnail`" (using taskProperties), it uses the `thumbnail.*` properties and operates in "non-force" profile (since the value is false), but when we run the task "`thumbnail.force`" the curation system uses the `thumbnail.force.*` properties. Notice that we did all this via local configuration - we have not had to touch the source code at all to obtain as many "profiles" as we would like.

See Task Properties in Curation Tasks(see page 539) for details of how properties are resolved in task code.

4.3.7 Task Parameters

New in DSpace 7, you can pass parameters to a task at invocation time. These runtime parameters will be presented to the task as if they were task properties (see above) and, if present, will override the value of identically-named properties. Example:

Task parameters

```
bin/dspace curate -t reticulate -i 123456789/36 -p foreground=red -p background=green
```

4.3.8 Scripted Tasks

> The procedure to set up curation tasks in Jython is described on a separate page: Curation tasks in Jython(see page 543)

DSpace 1.8 includes limited (and somewhat experimental) support for deploying and running tasks written in languages other than Java. Since version 6, Java has provided a standard way (API) to invoke so-called scripting or dynamic language code that runs on the java virtual machine (JVM). Scripted tasks are those written in a language accessible from this API. The exact number of supported languages will vary over time, and the degree of maturity of each language, or suitability of the language for curation tasks will also vary significantly. However, preliminary work indicates that Ruby (using the JRuby runtime) and Groovy may prove viable task languages.

Support for scripted tasks does **not** include any DSpace pre-installation of the scripting language itself - this must be done according to the instructions provided by the language maintainers, and typically only requires a few additional jars on the DSpace classpath. Once one or more languages have been installed into the DSpace deployment, task support is fairly straightforward. One new property must be defined in `[dspace]/config/modules/curate.cfg`:

```
curate.script.dir = ${dspace.dir}/scripts
```

This merely defines the directory location (usually relative to the deployment base) where task script files should be kept. This directory will contain a "catalog" of scripted tasks named `task.catalog` that contains information needed to run scripted tasks. Each task has a 'descriptor' property with value syntax:

`<engine>|<relFilePath>|<implClassCtor>`

An example property for a link checking task written in Ruby might be:

```
linkchecker = ruby|rubytask.rb|LinkChecker.new
```

This descriptor means that a "`ruby`" script engine will be created, a script file named "`rubytask.rb`" in the directory `<script.dir>` will be loaded and the resolver will expect an evaluation of "`LinkChecker.new`" will provide a correct implementation object. Note that the task must be configured in all other ways just like java tasks (in `ui.tasknames, ui.taskgroups`, etc).

Script files may embed their descriptors to facilitate deployment. To accomplish this, a script must include the descriptor string with syntax:
`$td=<descriptor>` somewhere on a comment line. For example:

```
# My descriptor $td=ruby|rubytask.rb|LinkChecker.new
```

For reasons of portability, the <relFilePath> component may be omitted in this context. Thus, "`$td=ruby||LinkChecker.new`" will be expanded to a descriptor with the name of the embedding file.

4.3.9 Bundled Tasks

DSpace bundles a small number of tasks of general applicability. Those that do not require configuration (or have usable default values) are activated by default to demonstrate the use of the curation system. They may be

deactivated by means of configuration, if desired, without affecting system integrity. Those that require configuration may be enabled (activated) by means editing DSpace configuration files. Each task is briefly described in this section.

All bundled tasks are in the package `org.dspace.ctask.general`. So, for example, to activate the no-operation task, which is implemented in the class `NoOpCurationTask`, one would configure:

```
plugin.named.org.dspace.curate.CurationTask = org.dspace.ctask.general.NoOpCurationTask = noop
```

4.3.9.1 Bitstream Format Profiler Task

The task with the taskname 'formatprofiler' (in the admin UI it is labeled "Profile Bitstream Formats") examines all the bitstreams in an item and produces a table ("profile") which is assigned to the result string. It is activated by default, and is configured to display in the administrative UI. The result string has the layout:

```
10 (K) Portable Network Graphics
5  (S) Plain Text
```

where the left column is the count of bitstreams of the named format and the letter in parentheses is an abbreviation of the repository-assigned support level for that format:

```
U  Unsupported
K  Known
S  Supported
```

The profiler will operate on any DSpace object. If the object is an item, then only that item's bitstreams are profiled; if a collection, all the bitstreams of all the items; if a community, all the items of all the collections of the community.

4.3.9.2 Link Checker Tasks

Two link checker tasks, BasicLinkChecker and MetadataValueLinkChecker, can be used to check for broken or unresolvable links appearing in item metadata.

This task is intended as a prototype / example for developers and administrators who are new to the curation system.

These tasks are not configurable.

Basic Link Checker

BasicLinkChecker iterates over all metadata fields ending in "uri" (eg. dc.relation.uri, dc.identifier.uri, dc.source.uri ...), attempts a GET to the value of the field, and checks for a 200 OK response.
Results are reported in a simple "one row per link" format.

Metadata Value Link Checker

MetadataValueLinkChecker parses all metadata fields for valid HTTP URLs, attempts a GET to those URLs, and checks for a 200 OK response.
Results are reported in a simple "one row per link" format.

4.3.9.3 MetadataWebService Task

DSpace item metadata can contain any number of identifiers or other field values that participate in networked information systems. For example, an item may include a DOI which is a controlled identifier in the DOI registry. Many web services exist to leverage these values, by using them as 'keys' to retrieve other useful data. In the DOI case for example, CrossRef provides many services that given a DOI will return author lists, citations, etc. The MetadataWebService task enables the use of such services, and allows you to obtain and (optionally) add to DSpace metadata the results of any web service call to any service provider. You simply need to describe what service you want to call, and what to do with the results. Using the task code (`[taskcode]`), you can create as many distinct tasks as you have services you want to call.

Each task description lives in a configuration file in 'config/modules' (or in your local.cfg), and is a simple properties file, like all other DSpace configuration files (see Configuration Reference(see page 552)). All of the settings associated with a given task should be prepended with the task name (as assigned in `config/modules/curate.cfg`). For example, if the task name is `issn2pubname` in curate.cfg, then all settings should start with "`issn2pubname.`" Your settings can either be set in your `local.cfg` , or in a new configuration file which is included (`include = path/to/new/file.cfg`) into either your local.cfg or the dspace.cfg. See the Configuration Reference(see page 552) for examples of including configuration files, or modifying your `local.cfg`

There are a few required properties you must configure for any service, and for certain services, a few additional ones. An example will illustrate best.

ISSN to Publisher Name

Suppose items (holding journal articles) include 'dc.identifier.issn' when available. We might also want to catalog the publisher name (in 'dc.publisher'). The cataloger could look up the name given the ISSN in various sources, but this 'research' is tedious, costly and error-prone. There are many good quality, free web services that can furnish this information. So we will configure a MetadataWebService task to call a service, and then automatically assign the publisher name to the item metadata. As noted above, all that is needed is a description of the service, and what to do with the results. Create a new file in 'config/modules' called 'issn2pubname.cfg' (or whatever is mnemonically useful to you). The first property in this file describes the service in a 'template'. The template is just the URL to call the web service, with parameters to substitute values in. Here we will use the 'Sherpa/Romeo' service:

```
[taskcode].template=http://www.sherpa.ac.uk/romeo/api29.php?issn={dc.identifier.issn}
```

When the task runs, it will replace '{dc.identifier.issn}' with the value of that field in the item, If the field has multiple values, the first one will be used. As a web service, the call to the above URL will return an XML document containing information (including the publisher name) about that ISSN. We need to describe what to do with this response document, i.e. what elements we want to extract, and what to do with the extracted content. This description is encoded in a property called the 'datamap'. Using the example service above we might have:

```
[taskcode].datamap=//publisher/name=>dc.publisher,//romeocolor
```

Each separate instruction is separated by a comma, so there are 2 instructions in this map. The first instruction essentially says: find the XML element 'publisher name' and assign the value or values of this element to the 'dc.publisher' field of the item. The second instruction says: find the XML element 'romeocolor', but do not add it to the DSpace item metadata - simply add it to the task result string (so that it can be seen by the person running the task). You can have as many instructions as you like in a datamap, which means that you can retrieve multiple values from a single web service call. A little more formally, each instruction consists of one to three parts. The first (mandatory) part identifies the desired data in the response document. The syntax (here '//publisher/name') is an XPath 1.0 expression, which is the standard language for navigating XML trees. If the value is to be assigned to the DSpace item metadata, then 2 other parts are needed. The first is the 'mapping symbol' (here '=>'), which is used to determine how the assignment should be made. There are 3 possible mapping symbols, shown here with their meanings:

```
'->' mapping will add to any existing value(s) in the item field
'=>' mapping will replace any existing value(s) in the item field
'~>' mapping will add *only if* item field has no existing value(s)
```

The third part (here 'dc.publisher') is simply the name of the metadata field to be updated. These two mandatory properties (template and datamap) are sufficient to describe a large number of web services. All that is required to enable this task is to edit '`config/modules/curate.cfg`' (or your `local.cfg`), and add '`issn2pubname`' to the list of tasks:

```
plugin.named.org.dspace.curate.CurationTask = org.dspace.ctask.general.MetadataWebService = issn2pubname
plugin.named.org.dspace.curate.CurationTask = org.dspace.ctask.general.MetadataWebService = doi2crossref
```

If you wish the task to be available in the Admin UI, see the Invocation from the Admin UI(see page 0) documentation (above) about how to configure it. The remaining sections describe some more specialized needs using the MetadataWebService task.

HTTP Headers

For some web services, protocol and other information is expressed not in the service URL, but in HTTP headers. Examples might be HTTP basic auth tokens, or requests for a particular media type response. In these cases, simply add a property to the configuration file (our example was 'issn2pubname.cfg') containing all headers you wish to transmit to the service:

```
[taskcode].headers=Accept: application/xml||Cache-Control: no-cache
```

You can specify any number of headers, just separate them with a 'double-pipe' ('`||`'). Ensure that any commas in values are escaped (with backslash comma, i.e. '`\,`').

Transformations

One potential problem with the simple parameter substitutions performed by the task is that the service might expect a different format or expression of a value than the way it is stored in the item metadata. For example, a DOI service might expect a bare prefix/suffix notation ('10.000/12345'), whereas the DSpace metadata field might have a URI representation ('http://dx.doi.org/10.000/12345').(see page 153) In these cases one can declare a 'transformation' of a value in the template. For example:

```
[taskcode].template=http://www.crossref.org/openurl/?id={doi:dc.relation.isversionof}&format=unixref
```

The 'doi:' prepended to the metadata field name declares that the value of the 'dc.relation.isversionof' field should be *transformed* before the substitution into the template using a transformation named 'doi'. The transformation is itself defined in the same configuration file as follows:

```
[taskcode].transform.doi=match 10. trunc 60
```

This would be read as: exclude the value string up to the occurrence of '10.', then truncate any characters after length 60. You may define as many transformations as you want in any task, although generally 1 or 2 will suffice. They keywords 'match', 'trunc', etc are names of 'functions' to be applied (in the order entered). The currently available functions are:

```
'cut' <number> = remove number leading characters
'trunc' <number> = remove trailing characters after number length
'match' <pattern> = start match at pattern
'text' <characters> = append literal characters (enclose in ' ' when whitespace needed)
```

When the task is run, if the transformation results in an invalid state (e.g. cutting more characters than there are in the value), the un-transformed value will be used and the condition will be logged. Transformations may also be applied to values returned from the web service. That is, one can apply the transformation to a value before assigning it to a metadata field. In this case, the declaration occurs in the datamap property, not the template:

```
[taskcode].datamap=//publisher/name=>shorten:dc.publisher,//romeocolor
```

Here the task will apply the 'shorten' transformation (which must be defined in the same config file) before assigning the value to 'dc.publisher'.

Result String Programatic Use

Normally a task result string appears in a window in the admin UI after it has been invoked. The MedataWebService task will concatenate all the values declared in the 'datamap' property and place them in the result string using the format: 'name:value name:value' for as many values as declared. In the example above we would get a string like 'publisher: Nature romeocolor: green'. This format is fine for simple display purposes, but can be tricky if the values contain spaces. You can override the space separator using an optional property 'separator' (put in the config file, with all other properties). If you use:

```
[taskcode].separator=||
```

for example, it becomes easy to parse the result string and preserve spaces in the values. This use of the result string can be very powerful, since you are essentially creating a map of returned values, which can then be used to populate a user interface, or any other way you wish to exploit the data (drive a workflow, etc).

Limits and Use

A few limitations should be noted. First, since the response parsing utilizes XPath, the service can only operate on XML, (not JSON) response documents. Most web services can provide either, so this should not be a major obstacle. The MetadataWebService can be used in many ways: showing an admin a value in the result string in a UI, run in a

batch to update a set of items, etc. One excellent configuration is to wire these tasks into submission workflow, so that 'automatic cataloging' of many fields can be performed on ingest.

4.3.9.4 MicrosoftTranslator Task

Microsoft Translator uses the Microsoft Translate API to translate metadata values from one source language into one or more target languages.

This task cab be configured to process particular fields, and use a default language if no authoritative language for an item can be found. Bing API v2 key is needed.

MicrosoftTranslator extends the more generic AbstractTranslator. This now seems wasteful, but a GoogleTranslator had also been written to extend AbstractTranslator. Unfortunately, Google has announced they are decommissioning free Translate API service, so this task hasn't been included in DSpace's general set of curation tasks.

Translated fields are added in addition to any existing fields, with the target language code in the 'language' column. This means that running a task multiple times over one item with the same configuration could result in duplicate metadata.

This task is intended as a prototype / example for developers and administrators who are new to the curation system.

Configure Microsoft Translator

An example configuration file can be found in `[dspace]/config/modules/translator.cfg`.

```
#---------------------------------------------------------------#
#----------TRANSLATOR CURATION TASK CONFIGURATIONS-------------#
#---------------------------------------------------------------#
# Configuration properties used solely by MicrosoftTranslator   #
# Curation Task (uses Microsoft Translation API v2)             #
#---------------------------------------------------------------#
## Translation field settings
##
## Authoritative language field
## This will be read to determine the original language an item was submitted in
## Default: dc.language
translator.field.language = dc.language

## Metadata fields you wish to have translated
translator.field.targets = dc.description.abstract, dc.title, dc.type

## Translation language settings
##
## If the language field configured in translate.field.language is not present
## in the record, set translate.language.default to a default source language
## or leave blank to use autodetection
translator.language.default = en

## Target languages for translation
translator.language.targets = de, fr

## Translation API settings
##
## Your Bing API v2 key and/or Google "Simple API Access" Key
## (note to Google users: your v1 API key will not work with Translate v2,
## you will need to visit https://code.google.com/apis/console and activate
## a Simple API Access key)
##
## You do not need to enter a key for both services.
translator.api.key.microsoft = YOUR_MICROSOFT_API_KEY_GOES_HERE
translator.api.key.google = YOUR_GOOGLE_API_KEY_GOES_HERE
```

4.3.9.5 NoOp Task

This task does absolutely nothing. It is intended as a starting point for developers and administrators wishing to learn more about the curation system.

4.3.9.6 Required Metadata Task

The "`requiredmetadata`" task examines item metadata and determines whether fields that the web submission (`input-forms.xml`) marks as required are present. It sets the result string to indicate either that all required fields are present, or constructs a list of metadata elements that are required but missing. When the task is performed on an item, it will display the result for that item. When performed on a collection or community, the task be performed on each item, and will display the *last* item result. If all items in the community or collection have all required fields, that will be the last in the collection. If the task fails for any item (i.e. the item lacks all required fields), the process is halted. This way the results for the 'failed' items are not lost.

4.3.9.7 Virus Scan Task

The "vscan" task performs a virus scan on the bitstreams of items using the ClamAV software product.

Clam AntiVirus is an open source (GPL) anti-virus toolkit for UNIX. A port for Windows is also available. The virus scanning curation task interacts with the ClamAV virus scanning service to scan the bitstreams contained in items, reporting on infection(s). Like other curation tasks, it can be run against a container or item, in the GUI or from the command line. It should be installed according to the documentation at http://www.clamav.net[188]. It should not be installed in the dspace installation directory. You may install it on the same machine as your dspace installation, or on another machine which has been configured properly.

Setup the service from the ClamAV documentation.

This plugin requires a ClamAV daemon installed and configured for TCP sockets. Instructions for installing ClamAV (http://[189]www.clamav.net/doc/latest/[190]**clamdoc**[191].pdf[192])

NOTICE: The following directions assume there is a properly installed and configured clamav daemon. Refer to links above for more information about ClamAV.
The Clam anti-virus database must be updated regularly to maintain the most current level of anti-virus protection. Please refer to the ClamAV documentation for instructions about maintaining the anti-virus database.

DSpace Configuration

In [dspace]/config/modules/curate.cfg, activate the task:

- Add the plugin to the list of curation tasks.

```
### Task Class implementations
plugin.named.org.dspace.curate.CurationTask = org.dspace.ctask.general.NoOpCurationTask = noop
plugin.named.org.dspace.curate.CurationTask = org.dspace.ctask.general.ProfileFormats = profileformats
plugin.named.org.dspace.curate.CurationTask = org.dspace.ctask.general.RequiredMetadata = requiredmetadata
# This is the ClamAV scanner plugin
plugin.named.org.dspace.curate.CurationTask = org.dspace.ctask.general.ClamScan = vscan
plugin.named.org.dspace.curate.CurationTask = org.dspace.ctask.general.MicrosoftTranslator = translate
plugin.named.org.dspace.curate.CurationTask = org.dspace.ctask.general.MetadataValueLinkChecker =
checklinks
```

- Optionally, add the vscan friendly name to the configuration to enable it in the administrative it in the administrative user interface.

```
curate.ui.tasknames = profileformats = Profile Bitstream Formats
curate.ui.tasknames = requiredmetadata = Check for Required Metadata
curate.ui.tasknames = checklinks = Check Links in Metadata
# Enable ClamAV from UI
curate.ui.tasknames = vscan = Virus Scan
```

188 http://www.clamav.net/
189 http://www.clamav.net/doc/latest/clamdoc.pdf
190 http://www.clamav.net/doc/latest/clamdoc.pdf
191 http://www.clamav.net/doc/latest/clamdoc.pdf
192 http://www.clamav.net/doc/latest/clamdoc.pdf

- In `[dspace]/config/modules`, edit configuration file clamav.cfg:

```
clamav.service.host = 127.0.0.1
# Change if not running on the same host as your DSpace installation.
clamav.service.port = 3310
# Change if not using standard ClamAV port
clamav.socket.timeout = 120
# Change if longer timeout needed
clamav.scan.failfast = false
# Change only if items have large numbers of bitstreams
```

- Finally, if desired virus scanning can be enabled as part of the submission process upload file step. In `[dspace]/config/modules`, edit configuration file `submission-curation.cfg`:

```
submission-curation.virus-scan = true
```

Task Operation from the Administrative user interface

Curation tasks can be run against container and item dspace objects by e-persons with administrative privileges. A curation tab will appear in the administrative ui after logging into DSpace:

1. Click on the curation tab.
2. Select the option configured in ui.tasknames above.
3. Select Perform.

Task Operation from the Item Submission user interface

If desired virus scanning can be enabled as part of the submission process upload file step. In `[dspace]/config/modules`, edit configuration file `submission-curation.cfg`:

```
submission-curation.virus-scan = true
```

Task Operation from the curation command line client

To output the results to the console:

```
[dspace]/bin/dspace curate -t vscan -i <handle of container or item dso> -r -
```

Or capture the results in a file:

```
[dspace]/bin/dspace curate -t vscan -i <handle of container or item dso> -r - > /<path...>/<name>
```

Table 1 – Virus Scan Results Table

GUI (Interactive Mode)	**FailFast**	**Expectation**
Container	T	Stop on 1st Infected Bitstream
Container	F	Stop on 1st Infected Item
Item	T	Stop on 1st Infected Bitstream
Item	F	Scan all bitstreams
Command Line		
Container	T	Report on 1st infected bitstream within an item/ Scan all contained Items
Container	F	Report on all infected bitstreams/Scan all contained Items
Item	T	Report on 1st infected bitstream
Item	F	Report on all infected bitstreams

4.4 Exporting Content and Metadata

General top level page to group all DSpace facilities for exporting content and metadata.

- Linked (Open) Data(see page 160)
- SWORDv1 Client(see page 176)
- Exchanging Content Between Repositories(see page 178)
- OAI(see page 179)
- OpenAIRE4 Guidelines Compliancy(see page 199)

4.4.1 Linked (Open) Data

4.4.1.1 Introduction

Exchanging repository contents

Most sites on the Internet are oriented towards human consumption. While HTML may be a good format for presenting information to humans, it is not a good format to export data in a way easy for a computer to work with. Like most software for building repositories, DSpace supports OAI-PMH(see page 179) as an interface to expose the stored metadata. While OAI-PMH is well known in the field of repositories, it is rarely known elsewhere (e.g. Google retired its support for OAI-PMH in 2008[193]). The Semantic Web is a generic approach to publish data on the Internet together with information about its semantics. Its application is not limited to repositories or libraries and it has a growing user base. RDF[194] and SPARQL[195] are W3C-released standards for publishing structured data on the web in a machine-readable way. The data stored in repositories is particularly suited for use in the Semantic Web, as the metadata are already available. It doesn't have to be generated or entered manually for publication as Linked Data. For most repositories, at least for Open Access repositories, it is quite important to share their stored content. Linked Data is a rather big chance for repositories to present their content in a way that can easily be accessed, interlinked and (re)used.

Terminology

We don't want to give a full introduction into the Semantic Web and its technologies here as this can be easily found in many places on the web. Nevertheless, we want to give a short glossary of the terms used most often in this context to make the following documentation more readable.

Semantic Web	The term "Semantic Web" refers to the part of the Internet containing Linked Data. Just like the World Wide Web, the Semantic Web is also woven together by links among the data.

193 http://googlewebmastercentral.blogspot.de/2008/04/retiring-support-for-oai-pmh-in.html
194 http://www.w3.org/standards/techs/rdf#w3c_all
195 http://www.w3.org/TR/sparql11-query/

Linked Data **Linked Open Data**	Data in RDF, following the Linked Data Principles[196] are called Linked Data. The Linked Data Principles describe the expected behavior of data publishers who shall ensure that the published data are easy to find, easy to retrieve, can be linked easily and link to other data as well. Linked Open Data is Linked Data published under an open license. There is no technical difference between Linked Data and Linked Open Data (often abbreviated as LOD). It is only a question of the license used to publish it.
RDF **RDF/XML** **Turtle** **N-Triples** **N3-Notation**	RDF is an acronym for Resource Description Framework, a metadata model. Don't think of RDF as a format, as it is a model. Nevertheless, there are different formats to serialize data following RDF. RDF/XML, Turtle, N-Triples and N3-Notation are probably the most well-known formats to serialize data in RDF. While RDF/XML uses XML, Turtle, N-Triples and N3-Notation don't and they are easier for humans to read and write. When we use RDF in DSpace configuration files, we currently prefer Turtle (but the code should be able to deal with any serialization).
Triple Store	A triple store is a database to natively store data following the RDF model. Just as you have to provide a relational database for DSpace, you have to provide a Triple Store for DSpace if you want to use the LOD support.
SPARQL	The SPARQL Protocol and RDF Query Language is a family of protocols to query triple stores. Since version 1.1, SPARQL can be used to manipulate triple stores as well, to store, delete or update data in triple stores. DSpace uses SPARQL 1.1 Graph Store HTTP Protocol and SPARQL 1.1 Query Language to communicate with the Triple Store. The SPARQL 1.1 Query Language is often referred to simply as SPARQL, so expect the SPARQL 1.1 Query Language if no other specific protocol out of the SPARQL family is explicitly specified.
SPARQL endpoint	A SPARQL endpoint is a SPARQL interface of a triple store. Since SPARQL 1.1, a SPARQL endpoint can be either read-only, allowing only to query the stored data; or readable and writable, allowing to modify the stored data as well. When talking about a SPARQL endpoint without specifying which SPARQL protocol is used, an endpoint supporting SPARQL 1.1 Query Language is meant.

4.4.1.2 Linked (Open) Data Support within DSpace

Starting with DSpace 5.0, DSpace provides support for publishing stored contents in form of Linked (Open) Data.

Architecture / Concept

To publish content stored in DSpace as Linked (Open) Data, the data have to be converted into RDF. The conversion into RDF has to be configurable as different DSpace instances may use different metadata schemata, different persistent identifiers (DOI, Handle, ...) and so on. Depending on the content to convert, configuration and other parameters, conversion may be time-intensive and impact performance. Content of repositories is much more often read then created, deleted or changed because the main goal of repositories is to safely store their contents.

196 http://www.w3.org/DesignIssues/LinkedData.html

For this reason, the content stored within DSpace is converted and stored in a triple store immediately after it is created or updated. The triple store serves as a cache and provides a SPARQL endpoint to make the converted data accessible using SPARQL. The conversion is triggered automatically by the DSpace event system and can be started manually using the command line interface – both cases are documented below. There is no need to backup the triple store, as all data stored in the triple store can be recreated from the contents stored elsewhere in DSpace (in the assetstore(s) and the database). Beside the SPARQL endpoint, the data should be published as RDF serialization as well. With dspace-rdf DSpace offers a module that loads converted data from the triple store and provides it as an RDF serialization. It currently supports RDF/XML, Turtle and N-Triples.

Repositories use Persistent Identifiers to make content citable and to address content. Following the Linked Data Principles, DSpace uses a Persistent Identifier in the form of HTTP(S) URIs, converting a Handle to http://hdl.handle.net/<handle> and a DOI to http://dx.doi.org/<doi>. Altogether, DSpace Linked Data support spans all three Layers: the storage layer with a triple store, the business logic with classes to convert stored contents into RDF, and the application layer with a module to publish RDF serializations. Just like DSpace allows you to choose Oracle or Postgresql as the relational database, you may choose between different triple stores. The only requirements are that the triple store must support SPARQL 1.1 Query Language and SPARQL 1.1 Graph Store HTTP Protocol which DSpace uses to store, update, delete and load converted data in/out of the triple store and uses the triple store to provide the data over a SPARQL endpoint.

Store public data only in the triple store!

The triple store should contain only data that are public, because the DSpace access restrictions won't affect the SPARQL endpoint. For this reason, DSpace converts only archived, discoverable (non-private) Items, Collections and Communities which are readable for anonymous users. Please consider this while configuring and/or extending DSpace Linked Data support.

The org.dspace.rdf.conversion[197] package contains the classes used to convert the repository content to RDF. The conversion itself is done by plugins. The org.dspace.rdf.conversion.ConverterPlugin[198] interface is really simple, so take a look at it you if can program in Java and want to extend the conversion. The only thing important is that plugins must only create RDF that can be made publicly available, as the triple store provides it using a sparql endpoint for which the DSpace access restrictions do not apply. Plugins converting metadata should check whether a specific metadata field needs to be protected or not (see org.dspace.app.util.MetadataExposure[199] on how to check that). The MetadataConverterPlugin[200] is heavily configurable (see below) and is used to convert the metadata of Items. The StaticDSOConverterPlugin[201] can be used to add static RDF Triples (see below). The SimpleDSORelationsConverterPlugin[202] creates links between items and collections, collections and communities, subcommunitites and their parents, and between top-level communities and the information representing the repository itself.

As different repositories uses different persistent identifiers to address their content, different algorithms to create URIs used within the converted data can be implemented. Currently HTTP(S) URIs of the repository (called local URIs), Handles and DOIs can be used. See the configuration part of this document for further information. If you want to add another algorithm, take a look at the org.dspace.rdf.storage.URIGenerator[203] interface.

197 https://github.com/DSpace/DSpace/tree/master/dspace-api/src/main/java/org/dspace/rdf/conversion
198 https://github.com/DSpace/DSpace/blob/master/dspace-api/src/main/java/org/dspace/rdf/conversion/ConverterPlugin.java
199 https://github.com/DSpace/DSpace/blob/master/dspace-api/src/main/java/org/dspace/app/util/MetadataExposure.java
200 https://github.com/DSpace/DSpace/blob/master/dspace-api/src/main/java/org/dspace/rdf/conversion/MetadataConverterPlugin.java
201 https://wiki.duraspace.org/dspace-api/src/main/java/org/dspace/rdf/conversion/StaticDSOConverterPlugin.java
202 https://wiki.duraspace.org/dspace-api/src/main/java/org/dspace/rdf/conversion/SimpleDSORelationsConverterPlugin.java
203 https://github.com/DSpace/DSpace/blob/master/dspace-api/src/main/java/org/dspace/rdf/storage/URIGenerator.java

Install a Triple Store

In addition to a normal DSpace installation you have to install a triple store. You can use any triple store that supports SPARQL 1.1 Query Language and SPARQL 1.1 Graph Store HTTP Protocol. If you do not have one yet, you can use Apache Fuseki. Download Fuseki from its official download page[204] and unpack the downloaded archive. The archive contains several scripts to start Fuseki. Use the start script appropriate to the OS of your choice with the options '--localhost --config=<dspace-install>/config/modules/rdf/fuseki-assembler.ttl'. Instead of changing to the directory into which you unpacked Fuseki, you may set the variable FUSEKI_HOME. If you're using Linux and bash, you unpacked Fuseki to /usr/local/jena-fuseki-1.0.1 and you installed DSpace to [dspace-install], this would look like this:

```
export FUSEKI_HOME=/usr/local/jena-fuseki-1.0.1 ; $FUSKI_HOME/fuseki-server --localhost --config [dspace-install]/config/modules/rdf/fuseki-assembler.ttl
```

Fuseki's archive contains a script to start Fuseki automatically at startup as well.

Make Fuseki connect to localhost only, by using the argument --localhost when launching if you use the configuration provided with DSpace! The configuration contains a writeable SPARQL endpoint that allows any connection to change/delete the content of your triple store.

Use Apache mod proxy, mod rewrite or any other appropriate web server/proxy to make localhost:3030/dspace/sparql readable from the internet. Use the address under which it is accessible as the address of your public sparql endpoint (see the property public.sparql.endpoint in the configuration reference(see page 165) below.).

The configuration provided within DSpace makes it store the files for the triple store under [dspace-install]/triplestore. Using this configuration, Fuseki provides three SPARQL endpoints: two read-only endpoints and one that can be used to change the data of the triple store. **You should not use this configuration if you let Fuseki connect to the internet directly** as it would make it possible for anyone to delete, change or add information to the triple store. The option --localhost tells Fuseki to listen only on the loopback device. You can use Apache mod_proxy or any other web or proxy server to make the read-only SPARQL endpoint accessible from the internet. With the configuration described, Fueski listens to the port 3030 using HTTP. Using the address *http://localhost:3030/* you can connect to the Fuseki Web UI. *http://localhost:3030/dspace/data* addresses a writeable SPARQL 1.1 HTTP Graph Store Protocol endpoint, and *http://localhost:3030/dspace/get* a read-only one. Under *http://localhost:3030/dspace/sparql* a read-only SPARQL 1.1 Query Language endpoint can be found. **The first one of these endpoints must be not accessible by the internet**, while the last one should be accessible publicly.

Default configuration and what you should change

First, you'll want to ensure the Linked Data endpoint is enabled/configured. In your `local.cfg`, add `rdf.enabled = true`. You can optionally change it's path by setting `rdf.path` (it defaults to "rdf" which means the Linked Data endpoint is available at `[dspace.server.url]/rdf/` (where `dspace.server.url` is also specified in your `local.cfg`)

204 http://jena.apache.org/documentation/serving_data/index.html#download-fuseki

In the file `[dspace]/config/dspace.cfg` you should look for the property `event.dispatcher.default.consumers` and add `rdf` there. Adding rdf there makes DSpace update the triple store automatically as the publicly available content of the repository changes.

As the Linked Data support of DSpace is highly configurable this section gives a short list of things you probably want to configure before using it. Below you can find more information on what is possible to configure.

In the file `[dspace]/config/modules/rdf.cfg` you want to configure the address of the public sparql endpoint and the address of the writable endpoint DSpace use to connect to the triple store (the properties `rdf.public.sparql.endpoint`, `rdf.storage.graphstore.endpoint`). In the same file you want to configure the URL that addresses the dspace-rdf module which is depending on where you deployed it (property `rdf.contextPath`) and switch content negotiation on (set property `rdf.contentNegotiation.enable = true`).

In the file `[dspace]/config/modules/rdf/constant-data-general.ttl` you should change the links to the Web UI of the repository and the public readable SPARQL endpoint. The URL of the public SPARQL endpoint should point to a URL that is proxied by a webserver to the Triple Store. See the section Install a Triple Store(see page 164) above for further information.

In the file `[dspace]/config/modules/rdf/constant-data-site.ttl` you may add any triples that should be added to the description of the repository itself.

If you want to change the way the metadata fields are converted, take a look into the file `[dspace]/config/modules/rdf/metadata-rdf-mapping.ttl`. This is also the place to add information on how to map metadata fields that you added to DSpace. There is already a quite acceptable default configuration for the metadata fields which DSpace supports out of the box. If you want to use some specific prefixes in RDF serializations that support prefixes, you have to edit `[dspace]onfig/modules/rdf/metadata-prefixes.ttl`.

Configuration Reference

There are several configuration files to configure DSpace's LOD support. The main configuration file can be found under `[dspace-source]/dspace/config/modules/rdf.cfg`. Within DSpace we use Spring to define which classes to load. For DSpace's LOD support this is done within `[dspace-source]/dspace/config/spring/api/rdf.xml`. All other configuration files are positioned in the directory `[dspace-source]/dspace/config/modules/rdf/`. Configurations in `rdf.cfg` can be modified directly, or overridden via your `local.cfg` config file (see Configuration Reference(see page 552)). You'll have to configure where to find and how to connect to the triple store. You may configure how to generate URIs to be used within the generated Linked Data and how to convert the contents stored in DSpace into RDF. We will guide you through the configuration file by file.

[dspace-source]/dspace/config/modules/rdf.cfg

Property:	rdf.enabled
Example Value:	rdf.enabled = true

Informational Note:	Defines whether the RDF endpoint is enabled or disabled (disabled by default). If enabled, the RDF endpoint is available at ${dspace.server.url}/${rdf.path}. Changing this value requires rebooting your servlet container (e.g. Tomcat)
Property:	rdf.path
Example Value:	rdf.path = rdf
Informational Note:	Defines the path of the RDF endpoint, if enabled. For example, a value of "rdf" (the default) means the RDF interface/endpoint is available at ${dspace.server.url}/rdf (e.g. if "dspace.server.url = http://localhost:8080/server", then it'd be available at "http://localhost:8080/server/rdf". Changing this value requires rebooting your servlet container (e.g. Tomcat)
Property:	rdf.contentNegotiation.enable
Example Value:	rdf.contentNegotiation.enable = true
Informational Note:	Defines whether content negotiation should be activated. Set this true, if you use Linked Data support.
Property:	rdf.contextPath
Example Value:	rdf.contextPath = ${dspace.baseUrl}/rdf

Informational Note:	The content negotiation needs to know where to refer if anyone asks for RDF serializations of content stored within DSpace. This property sets the URL where the dspace-rdf module can be reached on the Internet (depending on how you deployed it).
Property:	rdf.public.sparql.endpoint
Example Value:	rdf.public.sparql.endpoint = http://${dspace.baseUrl}/sparql
Informational Note:	Address of the read-only public SPARQL endpoint supporting SPARQL 1.1 Query Language.
Property:	rdf.storage.graphstore.endpoint
Example Value:	rdf.storage.graphstore.endpoint = http://localhost:3030/dspace/data
Informational Note:	Address of a writable SPARQL 1.1 Graph Store HTTP Protocol endpoint. This address is used to create, update and delete converted data in the triple store. If you use Fuseki with the configuration provided as part of DSpace 5, you can leave this as it is. If you use another Triple Store or configure Fuseki on your own, change this property to point to a writeable SPARQL endpoint supporting the SPARQL 1.1 Graph Store HTTP Protocol.
Property:	rdf.storage.graphstore.authentication
Example Value:	rdf.storage.graphstore.authentication = no

Informational Note:	Defines whether to use HTTP Basic authentication to connect to the writable SPARQL 1.1 Graph Store HTTP Protocol endpoint.
Properties:	rdf.storage.graphstore.login rdf.storage.graphstore.password
Example Values:	rdf.storage.graphstore.login = dspace rdf.storage.graphstore.password =ecapsd
Informational Note:	Credentials for the HTTP Basic authentication if it is necessary to connect to the writable SPARQL 1.1 Graph Store HTTP Protocol endpoint.
Property:	rdf.storage.sparql.endpoint
Example Value:	rdf.storage.sparql.endpoint = http://localhost:3030/dspace/sparql
Informational Note:	Besides a writable SPARQL 1.1 Graph Store HTTP Protocol endpoint, DSpace needs a SPARQL 1.1 Query Language endpoint, which can be read-only. This property allows you to set an address to be used to connect to such a SPARQL endpoint. If you leave this property empty the property ${rdf.public.sparql.endpoint} will be used instead.
Properties:	rdf.storage.sparql.authentication rdf.storage.sparql.login rdf.storage.sparql.password
Example Values:	rdf.storage.sparql.authentication = yes rdf.storage.sparql.login = dspace rdf.storage.sparql.password = ecapsd

Informational Note:	As for the SPARQL 1.1 Graph Store HTTP Protocol you can configure DSpace to use HTTP Basic authentication to authenticate against the (read-only) SPARQL 1.1 Query Language endpoint.
Property:	rdf.converter.DSOtypes
Example Value:	rdf.converter.DSOtypes = SITE, COMMUNITY, COLLECTION, ITEM
Informational Note:	Define which kind of DSpaceObjects should be converted. Bundles and Bitstreams will be converted as part of the Item they belong to. Don't add EPersons here unless you really know what you are doing. All converted data is stored in the triple store that provides a publicly readable SPARQL endpoint. So all data converted into RDF is exposed publicly. Every DSO type you add here must have an HTTP URI to be referenced in the generated RDF, which is another reason not to add EPersons here currently.
The following properties configure the StaticDSOConverterPlugin.	
Properties:	rdf.constant.data.GENERAL rdf.constant.data.COLLECTION rdf.constant.data.COMMUNITY rdf.constant.data.ITEM rdf.constant.data.SITE
Example Values:	rdf.constant.data.GENERAL = ${dspace.dir}/config/modules/rdf/constant-data-general.ttl rdf.constant.data.COLLECTION = ${dspace.dir}/config/modules/rdf/constant-data-collection.ttl rdf.constant.data.COMMUNITY = ${dspace.dir}/config/modules/rdf/constant-data-community.ttl rdf.constant.data.ITEM = ${dspace.dir}/config/modules/rdf/constant-data-item.ttl rdf.constant.data.SITE = ${dspace.dir}/config/modules/rdf/constant-data-site.ttl
Informational Note:	These properties define files to read static data from. These data should be in RDF, and by default Turtle is used as serialization. The data in the file referenced by the property ${rdf.constant.data.GENERAL} will be included in every Entity that is converted to RDF. E.g. it can be used to point to the address of the public readable SPARQL endpoint or may contain the name of the institution running DSpace. The other properties define files that will be included if a DSpace Object of the specified type (collection, community, item or site) is converted. This makes it possible to add static content to every Item, every Collection, ...
The following properties configure the MetadataConverterPlugin.	

Property:	rdf.metadata.mappings
Example Value:	rdf.metadata.mappings = ${dspace.dir}/config/modules/rdf/metadata-rdf-mapping.ttl
Informational Note:	Defines the file that contains the mappings for the MetadataConverterPlugin. See below the description of the configuration file [dspace-source]/dspace/config/modules/rdf/metadata-rdf-mapping.ttl.
Property:	rdf.metadata.schema
Example Value:	rdf.metadata.schema = file://${dspace.dir}/config/modules/rdf/metadata-rdf-schema.ttl
Informational Note:	Configures the URL used to load the RDF Schema of the DSpace Metadata RDF mapping Vocabulary. Using a file:// URI makes it possible to convert DSpace content without having an internet connection. The version of the schema has to be the right one for the used code. In DSpace 5.0 we use the version 0.2.0. This Schema can be found here as well: http://digital-repositories.org/ontologies/dspace-metadata-mapping/0.2.0. The newest version of the Schema can be found here: http://digital-repositories.org/ontologies/dspace-metadata-mapping/[205].
Property:	rdf.metadata.prefixes
Example Value:	rdf.metadata.prefixes = ${dspace.dir}/config/modules/rdf/metadata-prefixes.ttl
Informational Note:	If you want to use prefixes in RDF serializations that support prefixes, you can define these prefixes in the file referenced by this property.

205 http://digital-repositories.org/ontologies/dspace-metadata-mapping/0.2.0

The following properties configure the SimpleDSORelationsConverterPlugin	
Property:	rdf.simplerelations.prefixes
Example Value:	rdf.simplerelations.prefixes = ${dspace.dir}/config/modules/rdf/simple-relations-prefixes.ttl
Informational Note:	If you want to use prefixes in RDF serializations that support prefixes, you can define these prefixes in the file referenced by this property.
Property:	rdf.simplerelations.site2community
Example Value:	rdf.simplerelations.site2community = http://purl.org/dc/terms/hasPart, http://digital-repositories.org/ontologies/dspace/0.1.0#hasCommunity
Informational Note:	Defines the predicates used to link from the data representing the whole repository to the top level communities. Defining multiple predicates separated by commas will result in multiple triples.
Property:	rdf.simplerelations.community2site
Example Value:	rdf.simplerelations.community2site = http://purl.org/dc/terms/isPartOf, http://digital-repositories.org/ontologies/dspace/0.1.0#isPartOfRepository

Informational Note:	Defines the predicates used to link from the top level communities to the data representing the whole repository. Defining multiple predicates separated by commas will result in multiple triples.
Property:	rdf.simplerelations.community2subcommunity
Example Value:	rdf.simplerelations.community2subcommunity = http://purl.org/dc/terms/hasPart, http://digital-repositories.org/ontologies/dspace/0.1.0#hasSubcommunity
Informational Note:	Defines the predicates used to link from communities to their subcommunities. Defining multiple predicates separated by commas will result in multiple triples.
Property:	rdf.simplerelations.subcommunity2community
Example Value:	rdf.simplerelations.subcommunity2community = http://purl.org/dc/terms/isPartOf, http://digital-repositories.org/ontologies/dspace/0.1.0#isSubcommunityOf
Informational Note:	Defines the predicates used to link from subcommunities to the communities they belong to. Defining multiple predicates separated by commas will result in multiple triples.
Property:	rdf.simplerelations.community2collection
Example Value:	rdf.simplerelations.community2collection = http://purl.org/dc/terms/hasPart, http://digital-repositories.org/ontologies/dspace/0.1.0#hasCollection

Informational Note:	Defines the predicates used to link from communities to their collections. Defining multiple predicates separated by commas will result in multiple triples.
Property:	rdf.simplerelations.collection2community
Example Value:	rdf.simplerelations.collection2community = http://purl.org/dc/terms/isPartOf, http://digital-repositories.org/ontologies/dspace/0.1.0#isPartOfCommunity
Informational Note:	Defines the predicates used to link from collections to the communities they belong to. Defining multiple predicates separated by commas will result in multiple triples.
Property:	rdf.simplerelations.collection2item
Example Value:	rdf.simplerelations.collection2item = http://purl.org/dc/terms/hasPart[206], http://digital-repositories.org/ontologies/dspace/0.1.0#hasItem
Informational Note:	Defines the predicates used to link from collections to their items. Defining multiple predicates separated by commas will result in multiple triples.
Property:	rdf.simplerelations.item2collection

206 http://purl.org/dc/terms/hasPart,%5C%22%20data-mce-href=

Example Value:	rdf.simplerelations.item2collection = http://purl.org/dc/terms/isPartOf[207], http://digital-repositories.org/ontologies/dspace/0.1.0#isPartOfCollection
Informational Note:	Defines the predicates used to link from items to the collections they belong to. Defining multiple predicates separated by commas will result in multiple triples.
Property:	rdf.simplerelations.item2bitstream
Example Value:	rdf.simplerelations.item2bitstream = http://purl.org/dc/terms/hasPart[208], http://digital-repositories.org/ontologies/dspace/0.1.0#hasBitstream
Informational Note:	Defines the predicates used to link from item to their bitstreams. Defining multiple predicates separated by commas will result in multiple triples.

[dspace-source]/dspace/config/modules/rdf/constant-data-*.ttl

As described in the documentation of the configuration file [dspace-source]/dspace/config/modules/rdf.cfg, the constant-data-*.ttl files can be used to add static RDF to the converted data. The data are written in Turtle, but if you change the file suffix (and the path to find the files in rdf.cfg) you can use any other RDF serialization you like to. You can use this, for example, to add a link to the public readable SPARQL endpoint, add a link to the repository homepage, or add a triple to every community or collection defining it as an entity of a specific type like a bibo:collection. The content of the file [dspace-source]/dspace/config/modules/rdf/constant-data-general.ttl will be added to every DSpaceObject that is converted. The content of the file [dspace-source]/dspace/config/modules/rdf/constant-data-community.ttl to every community, the content of the file [dspace-source]/dspace/config/modules/rdf/constant-data-collection.ttl to every collection and the content of the file [dspace-source]/dspace/config/modules/rdf/constant-data-item.ttl to every Item. You can use the file [dspace-source]/dspace/config/modules/rdf/constant-data-site.ttl to specify data representing the whole repository.

[dspace-source]/dspace/config/modules/rdf/metadata-rdf-mapping.ttl

This file should contain several metadata mappings. A metadata mapping defines how to map a specific metadata field within DSpace to a triple that will be added to the converted data. The MetadataConverterPlugin uses these metadata mappings to convert the metadata of a item into RDF. For every metadata field and value it looks if any of

207 http://purl.org/dc/terms/isPartOf,
208 http://purl.org/dc/terms/hasPart,%5C%22%20data-mce-href=

the specified mappings matches. If one does, the plugin creates the specified triple and adds it to the converted data. In the file you'll find a lot of examples on how to define such a mapping.

For every mapping a metadata field name has to be specified, e.g. dc.title, dc.identifier.uri. In addition you can specify a condition that is matched against the field's value. The condition is specified as a regular expression (using the syntax of the java class java.util.regex.Pattern). If a condition is defined, the mapping will be used only on fields those values which are matched by the regex defined as condition.

The triple to create by a mapping is specified using reified RDF statements. The DSpace Metadata RDF Mapping Vocabulary[209] defines some placeholders that can be used. The most important placeholder is dm:DSpaceObjectIRI which is replaced by the URI used to identify the entity being converted to RDF. That means if a specific Item is converted the URI used to address this Item in RDF will be used instead of dm:DSpaceObjectIRI. There are three placeholders that allow reuse of the value of a meta data field. dm:DSpaceValue will be replace by the value as it is. dm:LiteralGenerator allows one to specify a regex and replacement string for it (see the syntax of the java classes java.util.regex.Pattern and java.util.regex.Matcher) and creates a Literal out of the field value using the regex and the replacement string. dm:ResourceGenerator does the same as dm:LiteralGenerator but it generates a HTTP(S) URI that is used in place. So you can use the resource generator to generate URIs containing modified field values (e.g. to link to classifications). If you know regular expressions and turtle, the syntax should be quite self explanatory.

[dspace-source]/dspace/config/modules/rdf/fuseki-assembler.ttl

This is a configuration for the triple store Fuseki of the Apache Jena project. You can find more information on the configuration it provides in the section Install a Triple Store(see page 164) above.

[dspace-source]/dspace/config/spring/api/rdf.xml

This file defines which classes are loaded by DSpace to provide the RDF functionality. There are two things you might want to change: the class that is responsible to generate the URIs to be used within the converted data, and the list of Plugins used during conversion. To change the class responsible for the URIs, change the following line:

```
<property name="generator" ref="org.dspace.rdf.storage.LocalURIGenerator"/>
```

This line defines how URIs should be generated, to be used within the converted data. The LocalURIGenerator generates URIs using the ${dspace.url} property. The HandleURIGenerator uses handles in form of HTTP URLs. It uses the property ${handle.canonical.prefix} to convert handles into HTTPS URLs. The class org.dspace.rdf.storage.DOIURIGenerator uses DOIs in the form of HTTP URLs if possible, or local URIs if there are no DOIs. It uses the DOI resolver "http://dx.doi.org" to convert DOIs into HTTP URLs. The class org.dspace.rdf.storage.DOIHandleGenerator does the same but uses Handles as fallback if no DOI exists. The fallbacks are necessary as DOIs are currently used for Items only and not for Communities or Collections.

All plugins that are instantiated within the configuration file will automatically be used during the conversion. Per default the list looks like the following:

209 http://digital-repositories.org/ontologies/dspace-metadata-mapping/

```
<!-- configure all plugins the converter should use. If you don't want to
         use a plugin, remove it here. -->
    <bean id="org.dspace.rdf.conversion.SimpleDSORelationsConverterPlugin" class="org.dspace.rdf.conversion
.SimpleDSORelationsConverterPlugin"/>
    <bean id="org.dspace.rdf.conversion.MetadataConverterPlugin" class="org.dspace.rdf.conversion.MetadataC
onverterPlugin"/>
    <bean id="org.dspace.rdf.conversion.StaticDSOConverterPlugin" class="org.dspace.rdf.conversion.StaticDS
OConverterPlugin"/>
```

You can remove plugins if you don't want them. If you develop a new conversion plugin, you want to add its class to this list.

Maintenance

As described above (see page 164)you should add `rdf` to the property `event.dispatcher.default.consumers` and in dspace.cfg. This configures DSpace to automatically update the triple store every time the publicly available content of the repository is changed. Nevertheless there is a command line tool that gives you the possibility to update the content of the triple store. As the triple store is used as a cache only, you can delete its content and reindex it every time you think it is necessary of helpful. The command line tool can be started by the following command which will show its online help:

```
[dspace-install]/bin/dspace rdfizer --help
```

The online help should give you all necessary information. There are commands to delete one specific entity; to delete all information stored in the triple store; to convert one item, one collection or community (including all subcommunities, collections and items) or to convert the complete content of your repository. If you start using the Linked Open Data support on a repository that already contains content, you should run `[dspace-install]/bin/dspace rdfizer --convert-all` once.

Every time content of DSpace is converted or Linked Data is requested, DSpace will try to connect to the triple store. So ensure that it is running (as you do with e.g. your sevlet container or relational database).

4.4.2 SWORDv1 Client

The embedded SWORD Client allows a user (currently restricted to an administrator) to copy an item to a SWORD server. This allows your DSpace installation to deposit items into another SWORD-compliant repository (including another DSpace install).

DSpace 7.0 does not yet support

The SWORDv1 Client is not available in DSpace 7.0. It may be restored in a later 7.x release, see DSpace Release 7.0 Status[210]

- Enabling the SWORD Client(see page 177)
- Configuring the SWORD Client(see page 177)

210 https://wiki.lyrasis.org/display/DSPACE/DSpace+Release+7.0+Status

4.4.2.1 Enabling the SWORD Client

The SWORDv1 Client is not available in DSpace 7.0. It may be restored in a later 7.x release, see DSpace Release 7.0 Status[211]

4.4.2.2 Configuring the SWORD Client

All the relevant configuration can be found in `sword-client.cfg`. These may be overridden in your `local.cfg` config (see Configuration Reference(see page 552)).

211 https://wiki.lyrasis.org/display/DSPACE/DSpace+Release+7.0+Status

Configuration File:	**[dspace]/config/modules/sword-client.cfg**
Property:	sword-client.targets
Example value:	sword-client.targets = http://localhost:8080/sword/servicedocument, \ http://client.swordapp.org/client/servicedocument, \ http://dspace.swordapp.org/sword/servicedocument, \ http://sword.eprints.org/sword-app/servicedocument, \ http://sword.intralibrary.com/IntraLibrary-Deposit/service, \ http://fedora.swordapp.org/sword-fedora/servicedocument
Informational note:	List of remote Sword servers. Used to build the drop-down list of selectable SWORD targets.
Property:	sword-client.file-types
Example value:	sword-client.file-types = application/zip
Informational note:	List of file types from which the user can select. If a type is not supported by the remote server it will not appear in the drop-down list.
Property:	sword-client.package-formats
Example value:	sword-client.package-formats = http://purl.org/net/sword-types/METSDSpaceSIP
Informational note:	List of package formats from which the user can select. If a format is not supported by the remote server it will not appear in the drop-down list.

4.4.3 Exchanging Content Between Repositories

- Transferring Items using OAI-ORE/OAI-PMH Harvester(see page 179)

4.4.3.1 Transferring Content via Export and Import

To migrate content from one DSpace to another, you can export content from the Source DSpace and import it into the Destination DSpace.

Transferring Communities, Collections, or Items using Packages

You may transfer any DSpace content (Communities, Collections or Items) from one DSpace to another by utilizing the AIP Backup and Restore(see page 411) tool. This tool allows you to export content into a series of Archival Information Packages (AIPs). These AIPs can be used to restore content (from a backup) or move/migrate content to another DSpace installation.

For more information see AIP Backup and Restore(see page 411).

4.4.3.2 Transferring Items using Simple Archive Format

Where items are to be moved between DSpace instances (for example from a test DSpace into a production DSpace) the Item Exporter and Item Importer(see page 233) can be used.

First, you should export the DSpace Item(s) into the Simple Archive Format, as detailed at: Importing and Exporting Items via Simple Archive Format(see page 233). Be sure to use the --migrate option, which removes fields that would be duplicated on import. Then import the resulting files into the other instance.

4.4.3.3 Transferring Items using OAI-ORE/OAI-PMH Harvester

> OAI Harvesting is not available in DSpace 7.0. It is scheduled to be restored in a later 7.x release (currently 7.1), see DSpace Release 7.0 Status[212]

You may also choose to enable the OAI-ORE Harvester. This OAI-ORE Harvester allows one DSpace installation to harvest Items (via OAI-ORE) from another DSpace Installation (or any other system supporting OAI-ORE). Items are harvested from a remote DSpace Collection into a local DSpace Collection. Harvesting can also be scheduled to run automatically (or by demand).

See OAI - Harvesting from another DSpace(see page 181)

4.4.4 OAI

4.4.4.1 OAI Interfaces

-
  - OAI-PMH Server(see page 180)
    - OAI-PMH Server Activation(see page 180)
    - OAI-PMH Server Maintenance(see page 180)
  - OAI-PMH / OAI-ORE Harvester (Client)(see page 181)
    - Harvesting from another DSpace(see page 181)

212 https://wiki.lyrasis.org/display/DSPACE/DSpace+Release+7.0+Status

- OAI-PMH / OAI-ORE Harvester Configuration(see page 181)

OAI-PMH Server

In the following sections and subpages, you will learn how to configure OAI-PMH server and activate additional OAI-PMH crosswalks. The user is also referred to OAI-PMH Data Provider(see page 654) for greater depth details of the program.

The OAI-PMH Interface may be used by other systems to harvest metadata records from your DSpace.

OAI-PMH Server Activation

DSpace's OAI-PMH server is enabled by default. However, you can choose to enable/disable it in your local.cfg using these configurations:

```
# Enable (true) or disable (false) OAI-PMH server
oai.enabled = true

# When enabled, OAI-PMH server is available at this path
oai.path = oai
```

If you modify either of these configuration, you must restart your Servlet Container (usually Tomcat).

- You can test that it is working by sending a request to: `[dspace.server.url]/[oai.path]/request?verb=Identify` (e.g. http://localhost:8080/server/oai/request?verb=Identify)
- The response should look similar to the response from the DSpace Demo Server: http://demo.dspace.org/oai/request?verb=Identify

If you're using a recent browser, you should see a HTML page describing your repository. What you're getting from the server is in fact an XML file with a link to an XSLT stylesheet that renders this HTML in your browser (client-side). Any browser that cannot interpret XSLT will display pure XML. The default stylesheet is located in `[dspace-source]/dspace-oai/src/main/resources/static/style.xsl` and can be changed by configuring the `stylesheet` attribute of the `Configuration` element in `[dspace]/config/crosswalks/oai/xoai.xml`.

Relevant Links

- OAI 2.0 Server(see page 189) - basic information needed to configure and use the OAI Server in DSpace
- OAI-PMH Data Provider 2.0 (Internals)(see page 186) - information on how it's implemented
- http://www.openarchives.org/pmh/ - information on the OAI-PMH protocol and its usage (not DSpace-specific)

OAI-PMH Server Maintenance

After activating the OAI-PMH server, you need to also ensure its index is updated on a regular basis. Currently, this doesn't happen automatically within DSpace. Instead, you must schedule the `[dspace.dir]/bin/dspace oai import` commandline tool to run on a regular basis (usually at least nightly, but you could schedule it more frequently).

Here's an example cron that can be used to schedule an OAI-PMH reindex on a nightly basis (for a full list of recommended DSpace cron tasks see Scheduled Tasks via Cron[213]):

213 https://wiki.lyrasis.org/display/DSDOC5x/Scheduled+Tasks+via+Cron

```
# Update the OAI-PMH index with the newest content (and re-optimize that index) at midnight every day
# NOTE: ONLY NECESSARY IF YOU ARE RUNNING OAI-PMH
# (This ensures new content is available via OAI-PMH and ensures the OAI-PMH index is optimized for better
performance)
0 0 * * * [dspace.dir]/bin/dspace oai import -o > /dev/null
```

More information about the `dspace oai` commandline tool can be found in the OAI Manager(see page 191) documentation.

OAI-PMH / OAI-ORE Harvester (Client)

This section describes the parameters used in configuring the OAI-ORE / OAI-ORE harvester. This harvester can be used to harvest content (bitstreams and metadata) into DSpace from an external OAI-PMH or OAI-ORE server.

> **DSpace 7.0 does not yet support**
>
> OAI Harvesting is not available in DSpace 7.0. It is scheduled to be restored in a later 7.x release (currently 7.1), see DSpace Release 7.0 Status[214]

Harvesting from another DSpace

If you are harvesting content (bitstreams and metadata) **from** an external DSpace installation via OAI-PMH & OAI-ORE, you first should verify that the external DSpace installation allows for OAI-ORE harvesting.

If the external DSpace is running v6.x or below, it must be running both the OAI-PMH interface and the XMLUI interface to support harvesting content from it via OAI-ORE.

If the external DSpace is running v7.x or above, it just needs to be running the OAI-PMH interface.

You can verify that OAI-ORE harvesting option is enabled by following these steps:

1. First, check to see if the external DSpace reports that it will support harvesting ORE via the OAI-PMH interface. Send the following request to the DSpace's OAI-PMH interface: `http://[full-URL-to-OAI-PMH]/request?verb=ListRecords&metadataPrefix=ore`
    - The response should be an XML document containing ORE, similar to the response from the DSpace Demo Server: http://demo.dspace.org/oai/request?verb=ListRecords&metadataPrefix=ore
2. For 6.x or below, you can verify that the XMLUI interface supports OAI-ORE (it should, as long as it's a current version of DSpace). First, find a valid Item Handle. Then, send the following request to the DSpace's XMLUI interface: `http://[full-URL-to-XMLUI]/metadata/handle/[item-handle]/ore.xml`
    - The response should be an OAI-ORE (XML) document which describes that specific Item. It should look similar to the response from the DSpace Demo Server: http://demo.dspace.org/xmlui/metadata/handle/10673/3/ore.xml

OAI-PMH / OAI-ORE Harvester Configuration

There are many possible configuration options for the OAI harvester. Most of these are contained in the `[dspace]/config/modules/oai.cfg` file (unless otherwise noted below). They may be updated there or overridden in your `local.cfg` config file (see Configuration Reference(see page 552)).

Configuration File:	**`[dspace]/config/modules/oai.cfg`**

214 https://wiki.lyrasis.org/display/DSPACE/DSpace+Release+7.0+Status

Property:	`oai.harvester.eperson`
Example Value:	`oai.harvester.eperson = admin@myu.edu`
Informational Note:	The EPerson under whose authorization automatic harvesting will be performed. This field does not have a default value and must be specified in order to use the harvest scheduling system. This will most likely be the DSpace admin account created during installation.
Property:	`oai.url`
Example Value:	`oai.url = ${dspace.server.url}/${oai.path}`
Informational Note:	The base url of the OAI-PMH disseminator webapp (i.e. do not include the / request on the end). This is necessary in order to mint URIs for ORE Resource Maps. The default value of `${dspace.baseUrl}/oai` will work for a typical installation, but should be changed if appropriate. Please note that `dspace.baseUrl` is defined in your `dspace.cfg` configuration file.
Property:	`oai.ore.authoritative.source`
Example Value:	`oai.ore.authoritative.source = oai`
Informational Note:	The webapp responsible for minting the URIs for ORE Resource Maps. If using oai, the `oai.url` config value must be set. • When set to 'oai', all URIs in ORE Resource Maps will be relative to the OAI-PMH URL (configured by `oai.url` above) • The URIs generated for ORE ReMs follow the following convention for either setting: http://\[base-URL\]/metadata/handle/\[item-handle\]/ore.xml
Property:	`oai.harvester.autoStart`
Example Value:	`oai.harvester.autoStart = false`
Informational Note:	Determines whether the harvest scheduler process starts up automatically when DSpace webapp is redeployed.

Property:	`oai.harvester.metadataformats.PluginName`
Example Value:	`oai.harvester.metadataformats.PluginName = \` `http://www.openarchives.org/OAI/2.0/oai_dc/, Simple Dublin Core`
Informational Note:	This field can be repeated and serves as a link between the metadata formats supported by the local repository and those supported by the remote OAI-PMH provider. It follows the form `oai.harvester.metadataformats.PluginName = NamespaceURI,Optional Display Name`. The pluginName designates the metadata schemas that the harvester "knows" the local DSpace repository can support. Consequently, the PluginName must correspond to a previously declared ingestion crosswalk. The namespace value is used during negotiation with the remote OAI-PMH provider, matching it against a list returned by the ListMetadataFormats request, and resolving it to whatever metadataPrefix the remote provider has assigned to that namespace. Finally, the optional display name is the string that will be displayed to the user when setting up a collection for harvesting. If omitted, the PluginName:NamespaceURI combo will be displayed instead.
Property:	`oai.harvester.oreSerializationFormat.OREPrefix`
Example Value:	`oai.harvester.oreSerializationFormat.OREPrefix = \` `http://www.w3.org/2005/Atom`
Informational Note:	This field works in much the same way as `oai.harvester.metadataformats.PluginName`. The `OREPrefix` must correspond to a declared ingestion crosswalk, while the Namespace must be supported by the target OAI-PMH provider when harvesting content.
Property:	`oai.harvester.timePadding`
Example Value:	`oai.harvester.timePadding = 120`
Informational Note:	Amount of time subtracted from the from argument of the PMH request to account for the time taken to negotiate a connection. Measured in seconds. Default value is 120.

Property:	`oai.harvester.harvestFrequency`
Example Value:	`oai.harvester.harvestFrequency = 720`
Informational Note:	How frequently the harvest scheduler checks the remote provider for updates. Should always be longer than *timePadding* . Measured in minutes. Default value is 720.
Property:	`oai.harvester.minHeartbeat`
Example Value:	`oai.harvester.minHeartbeat = 30`
Informational Note:	The heartbeat is the frequency at which the harvest scheduler queries the local database to determine if any collections are due for a harvest cycle (based on the *harvestFrequency*) value. The scheduler is optimized to then sleep until the next collection is actually ready to be harvested. The *minHeartbeat* and *maxHeartbeat* are the lower and upper bounds on this timeframe. Measured in seconds. Default value is 30.
Property:	`oai.harvester.maxHeartbeat`
Example Value:	`oai.harvester.maxHeartbeat = 3600`
Informational Note:	The heartbeat is the frequency at which the harvest scheduler queries the local database to determine if any collections are due for a harvest cycle (based on the *harvestFrequency*) value. The scheduler is optimized to then sleep until the next collection is actually ready to be harvested. The *minHeartbeat* and *maxHeartbeat* are the lower and upper bounds on this timeframe. Measured in seconds. Default value is 3600 (1 hour).
Property:	`oai.harvester.maxThreads`
Example Value:	`oai.harvester.maxThreads = 3`
Informational Note:	How many harvest process threads the scheduler can spool up at once. Default value is 3.
Property:	`oai.harvester.threadTimeout`

Example Value:	`oai.harvester.threadTimeout = 24`
Informational Note:	How much time passes before a harvest thread is terminated. The termination process waits for the current item to complete ingest and saves progress made up to that point. Measured in hours. Default value is 24.
Property:	`oai.harvester.unknownField`
Example Value:	`oai.harvester.unkownField = fail | add | ignore`
Informational Note:	You have three (3) choices. When a harvest process completes for a single item and it has been passed through ingestion crosswalks for ORE and its chosen descriptive metadata format, it might end up with DIM values that have not been defined in the local repository. This setting determines what should be done in the case where those DIM values belong to an already declared schema. *Fail* will terminate the harvesting task and generate an error. **Ignore** will quietly omit the unknown fields. **Add** will add the missing field to the local repository's metadata registry. Default value: **fail**.
Property:	`oai.harvester.unknownSchema`
Example Value:	`oai.harvester.unknownSchema = fail | add | ignore`
Informational Note:	When a harvest process completes for a single item and it has been passed through ingestion crosswalks for ORE and its chosen descriptive metadata format, it might end up with DIM values that have not been defined in the local repository. This setting determines what should be done in the case where those DIM values belong to an unknown schema. **Fail** will terminate the harvesting task and generate an error. **Ignore** will quietly omit the unknown fields. **Add** will add the missing schema to the local repository's metadata registry, using the schema name as the prefix and "unknown" as the namespace. Default value: **fail**.
Property:	`oai.harvester.acceptedHandleServer`
Example Value:	`oai.harvester.acceptedHandleServer = \` `hdl.handle.net, handle.test.edu`

Informational Note:	A harvest process will attempt to scan the metadata of the incoming items (identifier.uri field, to be exact) to see if it looks like a handle. If so, it matches the pattern against the values of this parameter. If there is a match the new item is assigned the handle from the metadata value instead of minting a new one. Default value: *hdl.handle.net* .
Property:	`oai.harvester.rejectedHandlePrefix`
Example Value:	`oai.harvester.rejectedHandlePrefix = 123456789, myeduHandle`
Informational Note:	Pattern to reject as an invalid handle prefix (known test string, for example) when attempting to find the handle of harvested items. If there is a match with this config parameter, a new handle will be minted instead. Default value: *123456789* .

4.4.4.2 OAI-PMH Data Provider 2.0 (Internals)

OAI-PMH Data Provider 2.0 (Internals)

The DSpace platform supports the Open Archives Initiative Protocol for Metadata Harvesting[215] (OAI-PMH) version 2.0 as a data provider. This is accomplished using the XOAI[216] OAI-PMH Java Toolkit.

The DSpace build process builds a single backend webapp, which optionally includes an OAI-PMH endpoint (when oai.enabled=true) In a typical configuration, this endpoint is deployed at `${dspace.server.url}/oai` (configured by "oai.path"), containing request, driver and openaire contexts, for example:

```
http://dspace.myu.edu/server/oai/request?verb=Identify
```

The "base URL" of this DSpace deployment would be:

```
http://dspace.myu.edu/server/oai/request
```

215 http://www.openarchives.org/
216 https://github.com/lyncode/xoai

But one could also provide the Driver or OpenAIRE contexts:

```
http://dspace.myu.edu/server/oai/driver
http://dspace.myu.edu/server/oai/openaire
```

It is this URL that should be registered with www.openarchives.org[217].

DSpace provides implementations of the XOAI data sources interfaces.

Sets

OAI-PMH allows repositories to expose an hierarchy of sets in which records may be placed. A record can be in zero or more sets.

DSpace exposes collections and communities as sets.

Each community and collection has a corresponding OAI set, discoverable by harvesters via the ListSets verb. The setSpec is based on the community/collection handle, with the "/" converted to underscore to form a legal setSpec. The setSpec is prefixed by "com_" or "col_" for communities and collections, respectively (this is a change in set names in DSpace 3.0 / OAI 2.0). For example:

```
col_1721.1_1234
```

Naturally enough, the community/collection name is also the name of the corresponding set.

Unique Identifier

Every item in OAI-PMH data repository must have an unique identifier, which must conform to the URI syntax. As of DSpace 1.2, Handles are not used; this is because in OAI-PMH, the OAI identifier identifies the *metadata record* associated with the *resource*. The *resource* is the DSpace item, whose *resource identifier* is the Handle. In practical terms, using the Handle for the OAI identifier may cause problems in the future if DSpace instances share items with the same Handles; the OAI metadata record identifiers should be different as the different DSpace instances would need to be harvested separately and may have different metadata for the item.

The OAI identifiers that DSpace uses are of the form:

```
oai:PREFIX:handle
```

For example:

```
oai:dspace.myu.edu:123456789/345
```

If you wish to use a different scheme, this can easily be changed by editing the value of identifier.prefix at [dspace]/config/modules/oai.cfg file.

217 http://www.openarchives.org/

Access control

OAI provides no authentication/authorisation details, although these could be implemented using standard HTTP methods. It is assumed that all access will be anonymous for the time being.

A question is, "is all metadata public?" Presently the answer to this is yes; all metadata is exposed via OAI-PMH, even if the item has restricted access policies. The reasoning behind this is that people who do actually have permission to read a restricted item should still be able to use OAI-based services to discover the content. But, exposed data could be changed by changing the XSLT defined at [dspace]/config/crosswalks/oai/metadataFormats.

Modification Date (OAI Date Stamp)

OAI-PMH harvesters need to know when a record has been created, changed or deleted. DSpace keeps track of a "last modified" date for each item in the system, and this date is used for the OAI-PMH date stamp. This means that any changes to the metadata (e.g. admins correcting a field, or a withdrawal) will be exposed to harvesters.

"About" Information

As part of each record given out to a harvester, there is an optional, repeatable "about" section which can be filled out in any (XML-schema conformant) way. Common uses are for provenance and rights information, and there are schemas in use by OAI communities for this. Presently DSpace does not provide any of this information, but XOAI core library allows its definition. This requires to dive into code and perform some changes.

Deletions

As DSpace supports two forms of deletions (withdrawals or permanent expunging), this has an impact on how OAI-PMH exposes delitions. During a permanent deletion (expunge), DSpace no longer retains any information about the deleted object. Therefore, permanent deletions "disappear" from OAI-PMH, as DSpace no longer has any information about the object. This is considered a "transient" approach to deletion based on OAI-PMH definitions[218].

When an item is withdrawn in DSpace, the item still exists but it hidden from public view. Withdrawn items will report a "<header status="deleted">" in OAI-PMH when a GetRecord request is made for a withdrawn item (however, they are NOT shown in an OAI-PMH "ListRecords" request by default). Keep in mind that the OAI-PMH index does NOT update automatically, so withdrawn items will not show this "deleted" status until "./dspace oai import" is next run.

Once an item has been withdrawn, OAI-PMH harvests of the date range in which the withdrawal occurred will find the "deleted" record header. Harvests of a date range prior to the withdrawal will *not* find the record, despite the fact that the record did exist at that time. As an example of this, consider an item that was created on 2002-05-02 and withdrawn on 2002-10-06. A request to harvest the month 2002-10 will yield the "record deleted" header. However, a harvest of the month 2002-05 will not yield the original record.

Flow Control (Resumption Tokens)

An OAI data provider can prevent any performance impact caused by harvesting by forcing a harvester to receive data in time-separated chunks. If the data provider receives a request for a lot of data, it can send part of the data with a resumption token. The harvester can then return later with the resumption token and continue.

DSpace supports resumption tokens for "ListRecords", "ListIdentifiers" and "ListSets" OAI-PMH requests.

---

218 https://www.openarchives.org/OAI/openarchivesprotocol.html#deletion

Each OAI-PMH ListRecords request will return at most 100 records (by default) but it could be configured in the `[dspace]/config/crosswalks/oai/xoai.xml` file.

When a resumption token is issued, the optional *completeListSize* and *cursor* attributes are included. OAI 2.0 resumption tokens are persistent, so *expirationDate* of the resumption token is undefined, they do not expire.

Resumption tokens contain all the state information required to continue a request and it is encoded in Base64.

4.4.4.3 OAI 2.0 Server

Introduction

Open Archives Initiative Protocol for Metadata Harvesting[219] is a low-barrier mechanism for repository interoperability. Data Providers are repositories that expose structured metadata via OAI-PMH. Service Providers then make OAI-PMH service requests to harvest that metadata. OAI-PMH is a set of six verbs or services that are invoked within HTTP.

What is OAI 2.0?

OAI 2.0 is a Java implementation of an OAI-PMH data provider interface (originally developed by Lyncode) that uses XOAI, an OAI-PMH Java Library[220].

Why OAI 2.0?

Projects like OpenAIRE[221] have specific metadata requirements (to the published content through the OAI-PMH interface). As the OAI-PMH protocol doesn't establish any frame to these specifics, OAI 2.0 can, in a simple way, have

219 http://www.openarchives.org/pmh/
220 https://github.com/DSpace/xoai
221 http://www.openaire.eu/

more than one instance of an OAI interface (feature provided by the XOAI core library) so one could define an interface for each project. That is the main purpose, although, OAI 2.0 allows much more than that.

Concepts (XOAI Core Library)

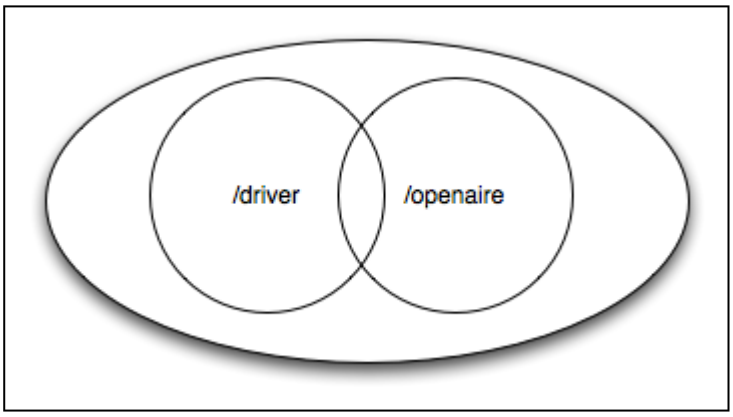
To understand how XOAI works, one must understand the concept of Filter, Transformer and Context. With a Filter it is possible to select information from the data source. A Transformer allows one to make some changes in the metadata before showing it in the OAI interface. XOAI also adds a new concept to the OAI-PMH basic specification, the concept of context. A context is identified in the URL:

```
http://www.example.com/oai/<context>
```

Contexts could be seen as virtual distinct OAI interfaces, so with this one could have things like:

- http://www.example.com/oai/request
- http://www.example.com/oai/driver
- http://www.example.com/oai/openaire

With this ingredients it is possible to build a robust solution that fulfills all requirements of *Driver*, *OpenAIRE* and also other project-specific requirements. As shown in Figure 1, with contexts one could select a subset of all available items in the data source. So when entering the *OpenAIRE* context, all OAI-PMH request will be restricted to that subset of items.



At this stage, contexts could be seen as sets (also defined in the basic OAI-PMH protocol). The magic of XOAI happens when one need specific metadata format to be shown in each context. Metadata requirements by *Driver* slightly differs from the *OpenAIRE* ones. So for each context one must define its specific transformer. So, contexts could be seen as an extension to the concept of sets.

To implement an OAI interface from the XOAI core library, one just need to implement the datasource interface.

OAI 2.0

OAI 2.0 is deployed as a part of the DSpace server (backend) webapp. OAI 2.0 has a configurable data source, by default it will not query the DSpace SQL database at the time of the OAI-PMH request. Instead, it keeps the required metadata in its Solr index (currently in a separate "oai" Solr core) and serves it from there. It's also possible to set OAI 2.0 to only use the database for querying purposes if necessary, but this decreases performance significantly. Furthermore, it caches the requests, so doing the same query repeatedly is very fast. In addition to that it also compiles DSpace items to make uncached responses much faster.

Details about OAI 2.0 internals can be found here[222].

> The OAI 2.0 Server only uses Solr for its indexing. The previous capability to use Database indexing has been removed.

Indexing OAI content

OAI 2.0 uses Solr for all indexing of content.

The Solr index can be updated at your convenience, depending on how fresh you need the information to be. Typically, the administrator sets up a nightly cron job to update the Solr index from the SQL database.

OAI Manager
OAI manager is a utility that allows one to do certain administrative operations with OAI. You can call it from the command line using the dspace launcher:

Syntax

[dspace]/bin/dspace oai <action> [parameters]

Actions

- import *Imports DSpace items into OAI Solr index (also cleans OAI cache)*
- clean-cache *Cleans the OAI cache*

Parameters

- -o *Optimize index after indexing*
- -c *Clears the Solr index before indexing (it will import all items again)*
- -v *Verbose output*
- -h *Shows an help text*

Scheduled Tasks
In order to refresh the OAI Solr index, it is required to run the `[dspace]/bin/dspace oai import` command periodically. You can add the following task to your crontab:

```
0 3 * * * [dspace]/bin/dspace oai import
```

Note that `[dspace]` should be replaced by the correct value, that is, the value defined in `dspace.cfg` parameter `dspace.dir`.

Client-side stylesheet

The OAI-PMH response is an XML file. While OAI-PMH is primarily used by harvesting tools and usually not directly by humans, sometimes it can be useful to look at the OAI-PMH requests directly - usually when setting it up for the first time or to verify any changes you make. For these cases, XOAI provides an XSLT stylesheet to transform the response XML to a nice looking, human-readable and interactive HTML. The stylesheet is linked from the XML response and the transformation takes place in the user's browser (this requires a recent browser, older browsers will only display the XML directly). Most automated tools are interested only in the XML file itself and will not perform the transformation. If you want, you can change which stylesheet will be used by placing it into the `[dspace]/webapps/oai/static` directory (or into the `[dspace-src]/dspace-xoai/dspace-xoai-webapp/src/main/webapp/static` after which you have to rebuild DSpace), modifying the "stylesheet" attribute of the

222 https://wiki.duraspace.org/pages/viewpage.action?pageId=32478690

"Configuration" element in `[dspace]/config/crosswalks/oai/xoai.xml` and restarting your servlet container.

Metadata Formats

By default OAI 2.0 provides 12 metadata formats within the /request context:

1. OAI_DC
2. DIDL
3. DIM
4. ETDMS
5. METS
6. MODS
7. OAI-ORE
8. QDC
9. RDF
10. MARC
11. UKETD_DC
12. XOAI

At /driver context it provdes:

1. OAI_DC
2. DIDL
3. METS

And at /openaire context it provides:

1. OAI_DC
2. METS

Encoding problems

There are two main potential sources of encoding problems:

a) The servlet connector port has to use the correct encoding. E.g. for Tomcat, this would be `<Connector port="8080" ... URIEncoding="UTF-8" />`, where the port attribute specifies port of the connector that DSpace is configured to access Solr on (this is usually 8080, 80 or in case of AJP 8009).

b) System locale of the dspace command line script that is used to do the oai import. Make sure the user account launching the script (usually from cron) has the correct locale set (e.g. en_US.UTF-8). Also make sure the locale is actually present on your system.

Configuration

Basic Configuration

Configuration File:	**`[dspace]/config/modules/oai.cfg`**
Property:	oai.enabled
Example Value:	`oai.enabled = true (default)`

Configuration File:	**`[dspace]/config/modules/oai.cfg`**
Information Note:	Allows you to enable or disable the OAI module/endpoint
Property:	oai.path
Example Value:	`oai.path = oai (default)`
Information Note:	Allows you to specify the path where the OAI module will be deployed. This path is relative to the dspace.server.url. So, for example, if "dspace.server.url=http://localhost:8080/server", then by default the OAI module is available at http://localhost:8080/server/oai/
Property:	`oai.storage`
Example Value:	`oai.storage = solr`
Information Note:	This allows to choose the OAI data source between solr and database. ONLY "solr" is supported at this time.
Property:	`oai.solr.url`
Example Value:	`oai.solr.url = ${solr.server}/oai`
Informational Note:	Solr Server location
Property:	`oai.identifier.prefix`
Example Value:	`oai.identifier.prefix = ${dspace.hostname}`
Informational Note:	OAI persistent identifier prefix. Format - oai:PREFIX:HANDLE
Property:	`oai.config.dir`
Example Value:	`oai.config.dir = ${dspace.dir}/config/crosswalks/oai`

Configuration File:	**`[dspace]/config/modules/oai.cfg`**
Informational Note:	Configuration directory, used by XOAI (core library). Contains xoai.xml, metadata format XSLTs and transformer XSLTs.
Property:	`oai.cache.enabled`
Example Value:	`oai.cache.enabled = true`
Informational Note:	Whether to enable the OAI cache. Default is true (for better performance).
Property:	`oai.cache.dir`
Example Value:	`oai.cache.dir = ${dspace.dir}/var/oai`
Informational Note:	Directory to store runtime generated files (for caching purposes).

Advanced Configuration

OAI 2.0 allows you to configure following advanced options:

- Contexts
- Transformers
- Metadata Formats
- Filters
- Sets

It's an XML file commonly located at: **[dspace]/config/crosswalks/oai/xoai.xml**

General options
These options influence the OAI interface globally. "per page" means per request, next page (if there is one) can be requested using resumptionToken provided in current page.

- identation [boolean] - whether the output XML should be indented to make it human-readable
- maxListIdentifiersSize [integer] - how many identifiers to show per page (verb=ListIdentifiers)
- maxListRecordsSize [integer] - how many records to show per page (verb=ListRecords)
- maxListSetsSize [integer] - how many sets to show per page (verb=ListSets)
- stylesheet [relative file path] - an xsl stylesheet used by client's web browser to transform the output XML into human-readable HTML

Their location and default values are shown in the following fragment:

```
<Configuration xmlns="http://www.lyncode.com/XOAIConfiguration"
    identation="false"
    maxListIdentifiersSize="100"
    maxListRecordsSize="100"
    maxListSetsSize="100"
    stylesheet="static/style.xsl">
```

Add/Remove Metadata Formats
Each context could have its own metadata formats. So to add/remove metadata formats to/from it, just need add/remove its reference within xoai.xml, for example, imagine one need to remove the XOAI schema from:

```
<Context baseurl="request">
  <Format refid="oaidc" />
  <Format refid="mets" />
  <Format refid="xoai" />
  <Format refid="didl" />
  <Format refid="dim" />
  <Format refid="ore" />
  <Format refid="rdf" />
  <Format refid="etdms" />
  <Format refid="mods" />
  <Format refid="qdc" />
  <Format refid="marc" />
  <Format refid="uketd_dc" />
</Context>
```

Then one would have:

```
<Context baseurl="request">
  <Format refid="oaidc" />
  <Format refid="mets" />
  <Format refid="didl" />
  <Format refid="dim" />
  <Format refid="ore" />
  <Format refid="rdf" />
  <Format refid="etdms" />
  <Format refid="mods" />
  <Format refid="qdc" />
  <Format refid="marc" />
  <Format refid="uketd_dc" />
</Context>
```

It is also possible to create new metadata format by creating a specific XSLT for it. All already defined XSLT for DSpace can be found in the **[dspace]/config/crosswalks/oai/metadataFormats** directory. So after producing a new one, add the following information (location marked using brackets) inside the <*Formats*> element in [dspace]/config/crosswalks/oai/xoai.xml:

```
<Format id="[IDENTIFIER]">
  <Prefix>[PREFIX]</Prefix>
  <XSLT>metadataFormats/[XSLT]</XSLT>
  <Namespace>[NAMESPACE]</Namespace>
  <SchemaLocation>[SCHEMA_LOCATION]</SchemaLocation>
</Format>
```

where:

Parameter	Description
IDENTIFIER	The identifier used within context configurations to reference this specific format, must be unique within all Metadata Formats available.
PREFIX	The prefix used in OAI interface (metadataPrefix=PREFIX).
XSLT	The name of the XSLT file within **[dspace]/config/crosswalks/oai/metadataFormats** directory
NAMESPACE	XML Default Namespace of the created Schema
SCHEMA_LOCATION	URI Location of the XSD of the created Schema

NOTE: Changes in [dspace]/config/crosswalks/oai/xoai.xml requires reloading/restarting the servlet container.

Add/Remove Metadata Fields
The internal DSpace fields (Dublin Core) are exposed in the internal XOAI format (xml). All other metadata formats exposed via OAI are mapped from this XOAI format using XSLT (xoai.xsl itself is just an identity transformation). These XSLT stylesheets are found in the **[dspace]/config/crosswalks/oai/metadataFormats** directory. So e.g. oai_dc.xsl is a transformation from the XOAI format to the oai_dc format (unqualified Dublin Core).

Therefore exposing any DSpace metadata field in any OAI format is just a matter of modifying the corresponding output format stylesheet (This assumes the general knowledge of how XSLT works. For a tutorial, see e.g. http://www.w3schools.com/xsl/).

For example, if you have a DC field "local.note.librarian" that you want to expose in oai_dc as <dc:note> (please note that this is not a valid DC field and thus breaks compatibility), then edit oai_dc.xsl and add the following lines just above the closing tag </oai_dc:dc>:

```
<xsl:for-each select="doc:metadata/doc:element[@name='local']/doc:element[@name='note']/doc:element/
doc:element/doc:field[@name='librarian']">
    <dc:note><xsl:value-of select="." /></dc:note>
</xsl:for-each>
```

If you need to add/remove metadata fields, you're changing the output format. Therefore it is recommended to create a new metadata format(see page 0) as a copy of the one you want to modify. This way the old format will remain available along with the new one and any upgrades to the original format during DSpace upgrades will not overwrite your customizations. If you need the format to have the same name as the original format (e.g. the default oai_dc format), you can create a new context[223] in xoai.xsl containing your modified format with the original name, which will be available as /oai/context-name.

NOTE: Please, keep in mind that the OAI provider caches the transformed output, so you have to run `[dspace]/bin/dspace oai clean-cache` after any .xsl modification and reload the OAI page for the changes to take effect. When adding/removing metadata formats, making changes in [dspace]/config/crosswalks/oai/xoai.xml requires reloading/restarting the servlet container.

Driver/OpenAIRE compliance

The default OAI 2.0 installation provides two new contexts. They are:

- Driver[224] context, which only exposes Driver compliant items;
- OpenAIRE[225] context, which only exposes OpenAIRE compliant items;

However, in order to be exposed DSpace items must be compliant with Driver/OpenAIRE guide-lines.

Driver Compliance

DRIVER Guidelines for Repository Managers and Administrators on how to expose digital scientific resources using OAI-PMH and Dublin Core Metadata, creating interoperability by homogenizing the repository output. The OAI-PMH `driver` set is based on DRIVER Guidelines 2.0.

This set is used to expose items of the repository that are available for open access. It's not necessary for all the items of the repository to be available for open access.

What specific metadata values are expected?

To have items in this set, you must configure your `input-forms.xml` file in order to comply with the DRIVER Guidelines:

- Must have a publication date - dc.date.issued (already configured in DSpace items)
- dc.language must use ISO639-3[226]
- the value of dc.type must be one of the 16 types named in the guidelines[227]

How do you easily add those metadata values?

As DRIVER guidelines use Dublin Core, all the needed items are already registered in DSpace. You just need to configure the deposit process.

OpenAIRE compliance

For OpenAIRE v4 compliance, see OpenAIRE4 Guidelines Compliancy(see page 199)

223 http://wiki.duraspace.org/#OAI2.0Server-Concepts(XOAICoreLibrary)
224 http://www.driver-support.eu/
225 http://www.openaire.eu/
226 http://www-01.sil.org/iso639-3/codes.asp
227 http://guidelines.readthedocs.org/en/latest/literature/field_publicationtype.html

The OpenAIRE Guidelines 2.0 provide the OpenAIRE compatibility to repositories and aggregators. By implementing these Guidelines, repository managers are facilitating the authors who deposit their publications in the repository in complying with the EC Open Access requirements. For developers of repository platforms, the Guidelines provide guidance to add supportive functionalities for authors of EC-funded research in future versions.

The name of the set in OAI-PMH is "ec_fundedresources" and will expose the items of the repository that comply with these guidelines. These guidelines are based on top of DRIVER guidelines. See version 2.0 of the Guidelines[228].

See the Application Profile of OpenAIRE[229].

What specific metadata values are expected?

These are the OpenAIRE metadata values only, to check these and driver metadata values check page 11 of the OpenAIRE guidelines 2.0.

- dc:relation with the project ID (see p.8)
- dc:rights with the access rights information from vocabulary (possible values here[230])

Optionally:

- dc:date with the embargo end date (recommended for embargoed items)

```
1  <dc:date>info:eu-repo/date/embargoEnd/2011-05-12<dc:date>
```

How do you easily add those metadata values?

- Have a dc:relation field in `input-forms.xml` with a list of the projects. You can also use the OpenAIRE Authority Control Addon[231] to facilitate the process of finding the project.
- Just use a combo-box for dc:rights to input the 4 options:
  - info:eu-repo/semantics/closedAccess
  - info:eu-repo/semantics/embargoedAccess
  - info:eu-repo/semantics/restrictedAccess
  - info:eu-repo/semantics/openAccess
- Use an input-box for dc:date to insert the embargo end date

Relevant Links

- OAI 2.0 is a standard part of DSpace 3.0
- Download & Install OAI 2.0 for DSpace 1.8.x: http://www.lyncode.com/dspace/addons/xoai/

Sanity check your OAI interface with the OAI Validator

There is a very useful validator for OAI interfaces available at http://validator.oaipmh.com, we urge you to use this validator to confirm your OAI interface is in fact usable.

228 https://guidelines.openaire.eu/en/latest/
229 http://colab.mpdl.mpg.de/mediawiki/ESciDoc_Application_Profile_OpenAIRE
230 http://wiki.surf.nl/display/standards/info-eu-repo/#info-eu-repo-AccessRights
231 http://projecto.rcaap.pt/index.php/lang-pt/consultar-recursos-de-apoio/remository?func=fileinfo&id=354

4.4.5 OpenAIRE4 Guidelines Compliancy

4.4.5.1 Loading of Entities and Fields

OpenAIRE4 features depends on Entities the entities feature on DSpace 7 and it's based on a set of configurations. In order to have your repository compliant with OpenAIRE4 guidelines you need to follow some steps:

The default submission-forms.xml file configures the form fields that allow the creation of the specific OpenAIRE entities and their relationships. In order to use those forms you need to configure your item-submission.xml and add these to the `<submission-map>`:

item-submission.xml

```
<name-map collection-handle="123456789/2" submission-name="openAIREPublicationSubmission" />
<name-map collection-handle="123456789/3" submission-name="openAIREPersonSubmission" />
<name-map collection-handle="123456789/5" submission-name="openAIREProjectSubmission" />
<name-map collection-handle="123456789/4" submission-name="openAIREOrganizationSubmission" />
```

Please note that my `collection-handle="123456789/4"` will be different in your system and it refers to the collection that will gather a specific Entity type like Publications, Persons, Projects or Organizations.

To load OpenAIRE Entities model you must firstly run the following:

loading openaire entity-relationships model

```
[/dspace]/bin/dspace initialize-entities -f [/dspace]/config/registries/openaire4-relationships.xml
```

and load the required fields

loading openaire registries

```
[/dspace]/bin/dspace registry-loader -metadata [/dspace]/config/registries/openaire4-types.xml
[/dspace]/bin/dspace registry-loader -metadata [/dspace]/config/registries/datacite-types.xml
```

After those steps your repository will have the required fields and entities for the compliancy.

4.4.5.2 OAI interface

As decided in our Entities meeting (2019-11-19 DSpace 7 Entities WG Meeting[232]), the XOAI Default Context should only display Publications or non Entity Items. For OpenAIRE4 it will also be considered only Publications as the main Entity to be processed and all the related ones will be loaded in the process.

OpenAIRE4 is accessible in a specific OAI context through the URL:
`http://[dspace-server-url]/oai/openaire4?verb=ListRecords&metadataPrefix=oai_openaire`[233]

in order to use it, you must first ensure you have the oai.cfg setting uncommented:
`oai.enabled = true`
(NOTE: you may need to restart your tomcat service)

If you need to display additional metadata at the oai_openaire metadata format, you could rename the file:

`[/dspace/]config/spring/api/virtual-metadata.xml.openaire`

and replace it with the existing one:

`[/dspace/]config/spring/api/virtual-metadata.xml`

Please note if you do this you should restart your tomcat service container.

This additional virtual metadata will enable to represent something like this in this XML in the oai_openaire metadata format, where you have, for instance, author identifiers:

oai datacite:creators example

```
<datacite:creators>
 <datacite:creator>
   <datacite:creatorName>Evans, R.J.</datacite:creatorName>
   <datacite:affiliation>Institute of Science and Technology</datacite:affiliation>
   <datacite:nameIdentifier nameIdentifierScheme="ORCID" schemeURI="http://orcid.org"> 1234-1234-1234-1234
 </datacite:nameIdentifier>
 </datacite:creator>
</datacite:creators>
```

Then you may need to run the OAI import from the command line with the cleaning cache parameter to reload all data to OAI:
`[/dspace/]/bin/dspace oai import -c`

4.5 Ingesting Content and Metadata

This is a new top level page grouping all documentation concerning all different ways to ingest content and metadata into DSpace

- Ingesting HTML Archives(see page 201)
- SWORDv2 Server(see page 202)

232 https://wiki.lyrasis.org/display/DSPACE/2019-11-19+DSpace+7+Entities+WG+Meeting
233 http://host/server/oai/openaire4?verb=ListRecords&metadataPrefix=oai_openaire

- SWORDv1 Server(see page 216)
- Exporting and Importing Community and Collection Hierarchy(see page 226)
- Importing Items via basic bibliographic formats (Endnote, BibTex, RIS, TSV, CSV) and online services (OAI, arXiv, PubMed, CrossRef, CiNii)(see page 229)
- Registering Bitstreams via Simple Archive Format(see page 231)
- Importing and Exporting Items via Simple Archive Format(see page 233)
- Importing and Exporting Content via Packages(see page 244)
- Configurable Workflow(see page 250)
- Submission User Interface(see page 260)

The section on Batch Metadata Editing(see page 287) also contains information on how to add items through spreadsheet ingest.

4.5.1 Ingesting HTML Archives

For the most part, at present DSpace simply supports uploading and downloading of bitstreams as-is. This is fine for the majority of commonly-used file formats – for example PDFs, Microsoft Word documents, spreadsheets and so forth. HTML documents (Web sites and Web pages) are far more complicated, and this has important ramifications when it comes to digital preservation:

- Web pages tend to consist of several files – one or more HTML files that contain references to each other, and stylesheets and image files that are referenced by the HTML files.
- Web pages also link to or include content from other sites, often imperceptibly to the end-user. Thus, in a few year's time, when someone views the preserved Web site, they will probably find that many links are now broken or refer to other sites than are now out of context.In fact, it may be unclear to an end-user when they are viewing content stored in DSpace and when they are seeing content included from another site, or have navigated to a page that is not stored in DSpace. This problem can manifest when a submitter uploads some HTML content. For example, the HTML document may include an image from an external Web site, or even their local hard drive. When the submitter views the HTML in DSpace, their browser is able to use the reference in the HTML to retrieve the appropriate image, and so to the submitter, the whole HTML document appears to have been deposited correctly. However, later on, when another user tries to view that HTML, their browser might not be able to retrieve the included image since it may have been removed from the external server. Hence the HTML will seem broken.
- Often Web pages are produced dynamically by software running on the Web server, and represent the state of a changing database underneath it.

Dealing with these issues is the topic of much active research. Currently, DSpace bites off a small, tractable chunk of this problem. DSpace can store and provide on-line browsing capability for *self-contained, non-dynamic* HTML documents. DSpace allows relative links between HTML documents stored together in a single item to work. In practical terms, this means:

- No dynamic content (CGI scripts and so forth)
- All links to preserved content must be *relative links*, that do not refer to 'parents' above the 'root' of the HTML document/site:
    - *diagram.gif* is OK
    - *image/foo.gif* is OK
    - *../index.html* is only OK in a file that is at least a directory deep in the HTML document/site hierarchy
    - */stylesheet.css* is not OK (the link will break)
    - http://somedomain.com/content.html is not OK (the link will continue to link to the external site which may change or disappear)

- Any 'absolute links' (e.g. http://somedomain.com/content.html) are stored 'as is', and will continue to link to the external content (as opposed to relative links, which will link to the copy of the content stored in DSpace.) Thus, over time, the content referred to by the absolute link may change or disappear.

4.5.2 SWORDv2 Server

SWORD (Simple Web-service Offering Repository Deposit) is a protocol that allows the remote deposit of items into repositories. DSpace implements the SWORD protocol via the 'sword' web application. The specification and further information can be found at http://swordapp.org/.

SWORD is based on the Atom Publish Protocol and allows service documents to be requested which describe the structure of the repository, and packages to be deposited.

4.5.2.1 Enabling SWORD v2 Server

To enable DSpace's SWORD v2 server, set the following in your local.cfg:

```
swordv2-server.enabled = true
# Optionally, if you wish to change its path
swordv2-server.path = swordv2
```

Keep in mind, modifying these settings will require restarting your Servlet Container (usually Tomcat).

Once enabled, the SWORDv2 module will be available at ${dspace.server.url}/${swordv2-server.path}. For example, if "dspace.server.url=http://localhost:8080/server", then (by default) it will be available at http://localhost:8080/server/swordv2/

4.5.2.2 Configuring SWORD v2 Server

These are the SWORD (v2) configurations. They may be edited directly or overridden in your local.cfg config (see Configuration Reference[234]).

Configuration File:	**[dspace]/config/modules/swordv2-server.cfg**
Property:	swordv2-server.enabled
Example Value:	swordv2-server.enabled = true (default is false)

234 https://wiki.duraspace.org/display/DSDOC6x/Configuration+Reference

Informational Note:	Whether SWORDv2 module is enabled or disabled (disabled by default). Modifying this setting will require restarting your Servlet Container (usually Tomcat).
Property:	`swordv2-server.path`
Example Value:	`swordv2-server.path = swordv2`
Informational Note:	When enabled, this is the subpath where the SWORDv2 module is deployed. This path is relative to `${dspace.server.url}`. Modifying this setting will require restarting your Servlet Container (usually Tomcat).
Property:	`swordv2-server.url`
Example Value:	`swordv2-server.url = ${dspace.server.url}/${swordv2-server.path}`
Informational Note:	The base url of the SWORD 2.0 system. This defaults to `${dspace.server.url}/${swordv2-server.path}`
Property:	`swordv2-server.collection.url`
Example Value:	`swordv2-server.collection.url = http://www.myu.ac.uk/swordv2/collection`
Informational Note:	The base URL of the SWORD collection. This is the URL from which DSpace will construct the deposit location URLs for collections. This defaults to `${dspace.server.url}/${swordv2-server.path}/collection`
Property:	`swordv2-server.servicedocument.url`
Example Value:	`swordv2-server.servicedocument.url = http://www.myu.ac.uk/swordv2/servicedocument`

Informational Note:	The service document URL of the SWORD collection. The base URL of the SWORD service document. This is the URL from which DSpace will construct the service document location urls for the site, and for individual collections. This defaults to `${dspace.server.url}/${swordv2-server.path}/servicedocument`
Property:	`swordv2-server.accept-packaging.collection`
Example Value:	`swordv2-server.accept-packaging.collection.METSDSpaceSIP = http://purl.org/net/sword/package/METSDSpaceSIP` `swordv2-server.accept-packaging.collection.SimpleZip = http://purl.org/net/sword/package/SimpleZip` `swordv2-server.accept-packaging.collection.Binary = http://purl.org/net/sword/package/Binary`
Informational Note:	The accept packaging properties, along with their associated quality values where appropriate. Package format information • METSDSpaceSIP[235]: zipfile containing mets.xml file describing the resources packed together with it in the root of the zipfile. • Binary: Binary resource that should be taken in as-is, not unpacked • SimpleZip: Zip file that should be unpacked and each file in the zip should be ingested separately. No metadata provided/ingested.
Property:	`swordv2-server.accept-packaging.item`
Example Value:	`swordv2-server.accept-packaging.item.METSDSpaceSIP = http://purl.org/net/sword/package/METSDSpaceSIP` `swordv2-server.accept-packaging.item.SimpleZip = http://purl.org/net/sword/package/SimpleZip` `swordv2-server.accept-packaging.item.Binary = http://purl.org/net/sword/package/Binary`

235 https://wiki.duraspace.org/display/DSPACE/DSpaceMETSSIPProfile

Informational Note:	The accept packaging properties for items. It is possible to configure this for specific collections by adding the handle of the collection to the setting, for example `swordv2-server.accept-packaging.collection.[handle].METSDSpaceSIP = http://purl.org/net/sword-types/METSDSpaceSIP` Package format information • METSDSpaceSIP[236]: zipfile containing mets.xml file describing the resources packed together with it in the root of the zipfile. • Binary: Binary resource that should be taken in as-is, not unpacked • SimpleZip: Zip file that should be unpacked and each file in the zip should be ingested separately. No metadata provided/ingested.
Property:	`swordv2-server.accepts`
Example Value:	`swordv2-server.accepts = application/zip, image/jpeg`
Informational Note:	A comma-separated list of MIME types that SWORD will accept. To accept all mimetypes, the value can be set to "*/*"
Property:	`swordv2-server.expose-communities`
Example Value:	`swordv2-server.expose-communities = false`
Informational Note:	Whether or not the server should expose a list of all the communities to a service document request. As deposits can only be made into a collection, it is recommended to leave this set to false.
Property:	`swordv2-server.max-upload-size`
Example Value:	`swordv2-server.max-upload-size = 0`

236 https://wiki.duraspace.org/display/DSPACE/DSpaceMETSSIPProfile

Informational Note:	The maximum upload size of a package through the SWORD interface (measured in bytes). This will be the combined size of all the files, metadata, and manifest file in a package - this is different to the maximum size of a single bitstream. If this is set to 0, no maximum file size will be enforced.
Property:	`swordv2-server.keep-original-package`
Example Value:	`swordv2-server.keep-original-package = true`
Informational Note:	Should DSpace store a copy of the orignal SWORD deposit package? This will cause the deposit process to be slightly slower and for more disk to be used, however original files will be preserved. It is recommended to leave this option enabled.
Property:	`swordv2-server.bundle.name`
Example Value:	`swordv2-server.bundle.name = SWORD`
Informational Note:	The bundle name that SWORD should store incoming packages within if `swordv2-server.keep-original-package` is set to true.
Property:	`swordv2-server.bundle.deleted`
Example Value:	`swordv2-server.bundle.deleted = DELETED`
Informational Note:	The bundle name that SWORD should use to store deleted bitstreams if `swordv2-server.versions.keep` is set to true. This will be used in the case that individual files are updated or removed via SWORD. If the entire Media Resource (files in the ORIGINAL bundle) is removed this will be backed up in its entirety in a bundle of its own

Property:	`swordv2-server.keep-package-on-fail`
Example Value:	`swordv2-server.keep-package-on-fail = false`
Informational Note:	In the event of package ingest failure, provide an option to store the package on the file system. The default is false. The location can be set using the `swordv2-server.failed-package-dir` setting.
Property:	`swordv2-server.failed-package-dir`
Example Value:	`swordv2-server.failed-package-dir = /dspace/upload`
Informational Note:	If `swordv2-server.keep-package-on-fail` is set to true, this is the location where the package would be stored.
Property:	`swordv2-server.on-behalf-of.enable`
Example Value:	`swordv2-server.on-behalf-of.enable = true`
Informational Note:	Should DSpace accept mediated deposits? See the SWORD specification for a detailed explanation of deposit On-Behalf-Of another user.
Property:	`swordv2-server.on-behalf-of.update.mediators`
Example Value:	`swordv2-server.on-behalf-of.update.mediators = admin@mydspace.edu, mediator@mydspace.edu`

Informational Note:	Which user accounts are allowed to do updates on items which already exist in DSpace, on-behalf-of other users? If this is left blank, or omitted, then all accounts can mediate updates to items, which could be a security risk, as there is no implicit checking that the authenticated user is a "legitimate" mediator
Property:	`swordv2-server.verbose-description.receipt.enable`
Example Value:	`swordv2-server.verbose-description.receipt.enable = false`
Informational Note:	Should the deposit receipt include a verbose description of the deposit? For use by developers - recommend to set to "false" for production systems
Property:	`swordv2-server.verbose-description.error.enable`
Example Value:	`swordv2-server.verbose-description.error.enable = true`
Informational Note:	should the error document include a verbose description of the error? For use by developers, although you may also wish to leave this set to "true" for production systems
Property:	`swordv2-server.error.alternate.url`
Example Value:	`swordv2-server.error.alternate.url = http://mydspace.edu/contact`[237]
Informational Note:	The error document can contain an alternate url, which the client can use to follow up any issues. For example, this could point to the Contact-Us page
Property:	`swordv2-server.error.alternate.content-type`
Example Value:	`swordv2-server.error.alternate.content-type = text/html`

237 http://localhost:8080/xmlui/contact

Informational Note:	The `swordv2-server.error.alternate.url` may have an associated content type, such as `text/html` if it points to a web page. This is used to indicate to the client what content type it can expect if it follows that url.
Property:	`swordv2-server.generator.url`
Example Value:	`swordv2-server.generator.url = http://www.dspace.org/ns/sword/2.0/`
Informational Note:	The URL which identifies DSpace as the software that is providing the SWORD interface.
Property:	`swordv2-server.generator.version`
Example Value:	`swordv2-server.generator.version = 2.0`
Informational Note:	The version of the SWORD interface.
Property:	`swordv2-server.auth-type`
Example Value:	`swordv2-server.auth-type = Basic`
Informational Note:	Which form of authentication to use. Normally this is set to `Basic` in order to use HTTP Basic.
Property:	`swordv2-server.upload.tempdir`
Example Value:	`swordv2-server.upload.tempd = /dspace/upload`

Informational Note:	The location where uploaded files and packages are stored while being processed.
Property:	`swordv2-server.updated.field`
Example Value:	`swordv2-server.updated.field = dc.date.updated`
Informational Note:	The metadata field in which to store the updated date for items deposited via SWORD.
Property:	`swordv2-server.slug.field`
Example Value:	`swordv2-server.slug.field = dc.identifier.slug`
Informational Note:	The metadata field in which to store the value of the slug header if it is supplied.
Property:	`swordv2-server.author.field`
Example Value:	`swordv2-server.author.field = dc.contributor.author`
Informational Note:	The metadata field in which to store the value of the atom entry author if it supplied.
Property:	`swordv2-server.title.field`
Example Value:	`swordv2-server.title.field = dc.title`

Informational Note:	The metadata field in which to store the value of the atom entry title if it supplied.
Property:	`swordv2-server.disseminate-packaging`
Example Value:	`swordv2-server.disseminate-packaging.METSDSpaceSIP = http://purl.org/net/sword/package/METSDSpaceSIP` `swordv2-server.disseminate-packaging.SimpleZip = http://purl.org/net/sword/package/SimpleZip`
Informational Note:	Supported packaging formats for the dissemination of packages.
Property:	`swordv2-server.statement.bundles`
Example Value:	`swordv2-server.statement.bundles = ORIGINAL, SWORD, LICENSE`
Informational Note:	Which bundles should the Statement include in its list of aggregated resources? The Statement will automatically mark any bitstreams which are in the bundle identified by the `${bundle.name}` property, provided that bundle is also listed here (i.e. if you want Original Deposits to be listed in the Statement then you should add the SWORD bundle to this list)
Property:	`plugin.single.org.dspace.sword2.WorkflowManager`
Example Value:	`plugin.single.org.dspace.sword2.WorkflowManager = org.dspace.sword2.WorkflowManagerDefault`
Informational Note:	Which workflow manager to use.
Property:	`swordv2-server.workflowmanagerdefault.always-update-metadata`
Example Value	`swordv2-server.workflowmanagerdefault.always-update-metadata = true`

Informational Note	Should the WorkflowManagerDefault plugin allow updates to the item's metadata to take place on items which are in states other than the workspace (e.g. in the workflow, archive, or withdrawn) ?
Property:	`swordv2-server.workflowmanagerdefault.file-replace.enable`
Example Value	`swordv2-server.workflowmanagerdefault.file-replace.enable = false`
Informational Note	Should the server allow PUT to individual files? If this is enabled, then DSpace may be used with the DepositMO SWORD extensions, BUT the caveat is that DSpace does not formally support Bitstream replace, so this is equivalent to a DELETE and then a POST, which violates the RESTfulness of the server. The resulting file DOES NOT have the same identifier as the file it was replacing. As such it is STRONGLY RECOMMENDED to leave this option turned off unless working explicitly with DepositMO enabled client environments
Property:	`swordv2-server.mets-ingester.package-ingester`
Example Value:	`swordv2-server.mets-ingester.package-ingester = METS`
Informational Note:	Which package ingester to use for METS packages.
Property:	`swordv2-server.restore-mode.enable`
Example Value:	`swordv2-server.restore-mode.enable = false`
Informational Note:	Should the SWORD server enable restore-mode when ingesting new packages. If this is enabled the item will be treated as a previously deleted item from the repository. If the item has previously been assigned a handle then that same handle will be restored to activity.
Property:	`swordv2-server.simpledc.*`

Example Value:	`swordv2-server.simpledc.abstract = dc.description.abstractsimpledc.date = dc.datesimpledc.rights = dc.rights`
Informational Note:	Configuration of metadata field mapping used by the SimpleDCEntryIngester, SimpleDCEntryDisseminator and FeedContentDisseminator
Property:	`swordv2-server.atom.*`
Example Value	`swordv2-server.atom.author = dc.contributor.author`
Informational Note:	Configuration of metadata field mapping used by the SimpleDCEntryIngester, SimpleDCEntryDisseminator and FeedContentDisseminator
Property:	`swordv2-server.metadata.replaceable`
Example Value	`swordv2-server.metadata.replaceable = dc.description.abstract, dc.rights, dc.title.alternative, dc.identifier.citation`
Informational Note	Used by SimpleDCEntryIngester: Which metadata fields can be replaced during a PUT to the Item of an Atom Entry document? Fields listed here are the ones which will be removed when a new PUT comes through (irrespective of whether there is a new incoming value to replace them)
Property:	`swordv2-server.multipart.entry-first`
Example Value:	`swordv2-server.multipart.entry-first = false`
Informational Note:	The order of precedence for importing multipart content. If this is set to `true` then metadata in the package will override metadata in the atom entry, otherwise the metadata in the atom entry will override that from the package.
Property:	`swordv2-server.workflow.notify`

Example Value:	`swordv2-server.workflow.notify = true`
Informational Note:	If the workflow gets started (the collection being deposited into has a workflow configured), should a notification get sent?
Property:	`swordv2-server.versions.keep`
Example Value:	`swordv2-server.versions.keep = true`
Informational Note:	When content is replaced, should the old version be kept? This creates a copy of the ORIGINAL bundle with the name V_YYYY-MM-DD.X where YYYY-MM-DD is the date the copy was created, and X is an integer from 0 upwards.
Property:	`swordv2-server.state.*`
Example Value:	`swordv2-server.state.workspace.uri = http://dspace.org/state/inprogress` `swordv2-server.state.workspace.description = The item is in the user workspace` `swordv2-server.state.workflow.uri = http://dspace.org/state/inreview` `swordv2-server.state.workflow.description = The item is undergoing review prior to acceptance in the archive`
Informational Note:	Pairs of states (URI and description) than items can be in. Typical states are `workspace`, `workflow`, `archive`, and `withdrawn`.
Property:	`swordv2-server.workspace.url-template`
Example Value	`swordv2-server.workspace.url-template = http://mydspace.edu/workspaceitems/#wsid#/edit`

Informational Note	URL template for links to items in the workspace (items in the archive will use the handle). The `#wsid#` url parameter will be replaced with the workspace id of the item. The example above shows how to construct this URL for the UI.

Other configuration options exist that define the mapping between mime types, ingesters, and disseminators. A typical configuration looks like this:

```
plugin.named.org.dspace.sword2.SwordContentIngester = \
  org.dspace.sword2.SimpleZipContentIngester = http://purl.org/net/sword/package/SimpleZip, \
  org.dspace.sword2.SwordMETSIngester = http://purl.org/net/sword/package/METSDSpaceSIP, \
  org.dspace.sword2.BinaryContentIngester = http://purl.org/net/sword/package/Binary

plugin.single.org.dspace.sword2.SwordEntryIngester = \
  org.dspace.sword2.SimpleDCEntryIngester

plugin.single.org.dspace.sword2.SwordEntryDisseminator = \
  org.dspace.sword2.SimpleDCEntryDisseminator

# note that we replace ";" with "_" as ";" is not permitted in the PluginManager names
plugin.named.org.dspace.sword2.SwordContentDisseminator = \
  org.dspace.sword2.SimpleZipContentDisseminator = http://purl.org/net/sword/package/SimpleZip, \
  org.dspace.sword2.FeedContentDisseminator = application/atom+xml, \
  org.dspace.sword2.FeedContentDisseminator = application/atom+xml_type_feed

# note that we replace ";" with "_" as ";" is not permitted in the PluginManager names
plugin.named.org.dspace.sword2.SwordStatementDisseminator = \
  org.dspace.sword2.AtomStatementDisseminator = atom, \
  org.dspace.sword2.OreStatementDisseminator = rdf, \
  org.dspace.sword2.AtomStatementDisseminator = application/atom+xml_type_feed, \
  org.dspace.sword2.OreStatementDisseminator = application/rdf+xml
```

4.5.2.3 Deposit to SWORDv2 Server

If you'd like to deposit content to your repository via the installed SWORD Server, you'll need to select a SWORD Client to do so.

- Some SWORDv2 Clients are available at http://swordapp.org/sword-v2/
- It's possible to simply deposit a valid SWORDv2 Zip package via common Linux commandline tools (e.g. curl). For example:

```
# Deposit a SWORD Zip package named "sword-package.zip" into a DSpace Collection (Handle 123456789/2)
 as user "test@dspace.org"
# (Please note that you WILL need to obviously modify the Collection location, user/password and
name of the SWORD package)

curl -i --data-binary "@sword-package.zip" -H "Content-Disposition:attachment; filename=sword-
package.zip" -H "Content-Type:application/zip" -H "Packaging:http://purl.org/net/sword/package/
METSDSpaceSIP" -u test@dspace.org:[password] -X POST http://[dspace.url]/swordv2/collection/
123456789/2

# Template 'curl' command:
#curl -i --data-binary "@[zip-package-name]" -H "Content-Disposition:attachment; filename=[zip-
package-name]" -H "Content-Type:application/zip" -H "Packaging:http://purl.org/net/sword/package/
METSDSpaceSIP" -u [user]:[password] -X POST http://[dspace.url]/swordv2/collection/[collection-
handle]
```

Other example SWORDv2 commands

```
# Example of retrieving Item information via "edit-media" path in ATOM format (can be run on any item
within DSpace, but requires authentication)
# NOTE: Accept header is required, and must be a format supported by a SwordContentDisseminator plugin (see
configuration above)
curl -i -H "Accept:application/atom+xml" -u test@dspace.org:[password] -X GET http://[dspace.url]/swordv2/
edit-media/[internal-item-identifier]
```

4.5.2.4 Troubleshooting

Missing expression of encoding in XML header

If your SWORD Deposit requests are unsuccessful, please check that the XML in your initial metadata deposit correctly specifies the encoding.

If you use:

```
<?xml version="1.0"?>
```

DSpace will default to UTF-32.

So to successfully deposit an XML in UTF-8, make sure you use:

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

4.5.3 SWORDv1 Server

SWORD (Simple Web-service Offering Repository Deposit) is a protocol that allows the remote deposit of items into repositories. DSpace implements the SWORD protocol via the 'sword' web application. The version of SWORD v1

currently supported by DSpace is 1.3. The specification and further information can be found at http://swordapp.org.

SWORD is based on the Atom Publish Protocol and allows service documents to be requested which describe the structure of the repository, and packages to be deposited.

- Enabling SWORD Server(see page 217)
- Configuring SWORD Server(see page 217)
- Deposit to SWORD Server(see page 226)

4.5.3.1 Enabling SWORD Server

To enable DSpace's SWORD v1 server, set the following in your local.cfg:

```
sword-server.enabled = true
# Optionally, if you wish to change its path
sword-server.path = sword
```

Keep in mind, modifying these settings will require restarting your Servlet Container (usually Tomcat).

Once enabled, the SWORD v1 module will be available at ${dspace.server.url}/${sword-server.path}. For example, if "dspace.server.url=http://localhost:8080/server", then (by default) it will be available at http://localhost:8080/server/sword/

4.5.3.2 Configuring SWORD Server

These are the SWORD (v1) configurations. They may be edited directly or overridden in your local.cfg config (see Configuration Reference(see page 552)).

Configuration File:	**[dspace]/config/modules/sword-server.cfg**
Property:	sword-server.enabled
Example Value:	sword-server.enabled = true (default is false)
Informational Note:	Whether SWORDv1 module is enabled or disabled (disabled by default). Modifying this setting will require restarting your Servlet Container (usually Tomcat).
Property:	sword-server.path

Configuration File:	**[dspace]/config/modules/sword-server.cfg**
Example Value:	`sword-server.path = sword`
Informational Note:	When enabled, this is the subpath where the SWORDv1 module is deployed. This path is relative to `${dspace.server.url}`. Modifying this setting will require restarting your Servlet Container (usually Tomcat).
Property:	`sword-server.mets-ingester.package-ingester`
Example Value:	`sword-server.mets-ingester.package-ingester = METS`
Informational Note:	The property key tell the SWORD METS implementation which package ingester to use to install deposited content. This should refer to one of the classes configured for: `plugin.named.org.dspace.content.packager.PackageIngester` The value of sword.mets-ingester.package-ingester tells the system which named plugin for this interface should be used to ingest SWORD METS packages.
Properties:	`mets.default.ingest.crosswalk.EPDCX` `mets.default.ingest.crosswalk.*` (NOTE: These configs are in the `dspace.cfg` file as they are used by many interfaces)
Example Value:	`mets.submission.crosswalk.EPDCX = EPDCX`

Configuration File:	**[dspace]/config/modules/sword-server.cfg**
Informational Note:	Define the metadata types which can be accepted/handled by SWORD during ingest of a package. Currently, EPDCX (EPrints DC XML) is the recommended default metadata format, but others are supported. An example of an EPDCX SWORD package can be found at [dspace-src]/dspace-sword/example/example.zip. Additional metadata types can be added to this list by just defining new configurations. For example, you can map a new "mdtype" MYFORMAT to a custom crosswalk named MYFORMAT: `mets.submission.crosswalk.MYFORMAT = MYFORMAT` You'd also want to map your new custom crosswalk to a stylesheet using the next configuration (crosswalk.submission.*.stylesheet).
Property:	`crosswalk.submission.EPDCX.stylesheet` (NOTE: This configuration is in the `dspace.cfg` file)
Example Value:	`crosswalk.submission.EPDCX.stylesheet = crosswalks/sword-swap-ingest.xsl`
Informational Note:	Define the stylesheet which will be used by the self-named XSLTIngestionCrosswalk class when asked to load the SWORD configuration (as specified above). This will use the specified stylesheet to crosswalk the incoming SWAP metadata to the DIM format for ingestion. Additional crosswalk types can be added to this list by just defining new configurations. For example, you can map a custom crosswalk named MYFORMAT to use a specific "my-crosswalk.xsl" stylesheet: `crosswalk.submission.MYFORMAT.stylesheet = crosswalks/my-crosswalk.xsl` Keep in mind, you'll need to also ensure MYFORMAT crosswalk is defined by the previous configuration (mets.submission.crosswalk.*).
Property:	`sword-server.deposit.url`
Example Value:	`sword-server.deposit.url = http://www.myu.ac.uk/sword/deposit`

Configuration File:	**[dspace]/config/modules/sword-server.cfg**
Informational Note:	The base URL of the SWORD deposit. This is the URL from which DSpace will construct the deposit location URLs for collections. The default is `${dspace.server.url}/${sword-server.path}/deposit` . In the event that you are not deploying DSpace as the ROOT application in the servlet container, this will generate incorrect URLs, and you should override the functionality by specifying in full as shown in the example value.
Property:	`sword-server.servicedocument.url`
Example Value:	`sword-server.servicedocument.url = http://www.myu.ac.uk/sword/servicedocument`
Informational Note:	The base URL of the SWORD service document. This is the URL from which DSpace will construct the service document location URLs for the site, and for individual collections. The default is `${dspace.server.url}/${sword-server.path}/servicedocument`. In the event that you are not deploying DSpace as the ROOT application in the servlet container, this will generate incorrect URLs, and you should override the functionality by specifying in full as shown in the example value.
Property:	`sword-server.media-link.url`
Example Value:	`sword-server.media-link.url = http://www.myu.ac.uk/sword/media-link`
Informational Note:	The base URL of the SWORD media links. This is the URL which DSpace will use to construct the media link URLs for items which are deposited via sword. The default is `${dspace.server.url}/${sword-server.path}/media-link`. In the event that you are not deploying DSpace as the ROOT application in the servlet container, this will generate incorrect URLs, and you should override the functionality by specifying in full as shown in the example value.
Property:	`sword-server.generator.url`
Example Value:	`sword-server.generator.url = http://www.dspace.org/ns/sword/1.3.1`

Configuration File:	**[dspace]/config/modules/sword-server.cfg**
Informational Note:	The URL which identifies the SWORD software which provides the sword interface. This is the URL which DSpace will use to fill out the atom:generator element of its atom documents. The default is: http://www.dspace.org/ns/sword/1.3.1 If you have modified your SWORD software, you should change this URI to identify your own version. If you are using the standard 'dspace-sword' module you will not, in general, need to change this setting.
Property:	`sword-server.updated.field`
Example Value:	`sword-server.updated.field = dc.date.updated`
Informational Note:	The metadata field in which to store the updated date for items deposited via SWORD.
Property:	`sword-server.slug.field`
Example Value:	`sword-server.slug.field = dc.identifier.slug`
Informational Note:	The metadata field in which to store the value of the slug header if it is supplied.
Properties:	`sword-server.accept-packaging.METSDSpaceSIP.identifier` `sword-server.accept-packaging.METSDSpaceSIP.q`
Example Value:	`sword-server.accept-packaging.METSDSpaceSIP.identifier = http://purl.org/net/sword-types/METSDSpaceSIP` `sword-server.accept-packaging.METSDSpaceSIP.q = 1.0`

Configuration File:	**[dspace]/config/modules/sword-server.cfg**
Informational Note:	The accept packaging properties, along with their associated quality values where appropriate. This is a Global Setting; these will be used on all DSpace collections
Property:	`sword-server.accepts`
Example Value:	`sword-server.accepts = application/zip, foo/bar`
Informational Note:	A comma separated list of MIME types that SWORD will accept.
Properties:	`sword-server.accept-packaging.[handle].METSDSpaceSIP.identifier` `sword-server.accept-packaging.[handle].METSDSpaceSIP.q`
Example Value:	`sword-server.accept-packaging.[handle].METSDSpaceSIP.identifier = http://purl.org/net/sword-types/METSDSpaceSIP` `sword-server.accept-packaging.[handle].METSDSpaceSIP.q = 1.0`
Informational Note:	Collection Specific settings: these will be used on the collections with the given handles.
Property:	`sword-server.expose-items`
Example Value:	`sword-server.expose-items = false`

Configuration File:	**[dspace]/config/modules/sword-server.cfg**
Informational Note:	Should the server offer up items in collections as sword deposit targets. This will be effected by placing a URI in the collection description which will list all the allowed items for the depositing user in that collection on request. **NOTE:** this will require an implementation of deposit onto items, which will not be forthcoming for a short while.
Property:	`sword-server.expose-communities`
Example Value:	`sword-server.expose-communities = false`
Informational Note:	Should the server offer as the default the list of all Communities to a Service Document request. If false, the server will offer the list of all collections, which is the default and recommended behavior at this stage. **NOTE:** a service document for Communities will not offer any viable deposit targets, and the client will need to request the list of Collections in the target before deposit can continue.
Property:	`sword-server.max-upload-size`
Example Value:	`sword-server.max-upload-size = 0`
Informational Note:	The maximum upload size of a package through the sword interface, in bytes. This will be the combined size of all the files, the metadata and any manifest data. It is NOT the same as the maximum size set for an individual file upload through the user interface. If not set, or set to 0, the sword service will default to no limit.
Property:	`sword-server.keep-original-package`
Example Value:	`sword-server.keep-original-package = true`

Configuration File:	**[dspace]/config/modules/sword-server.cfg**
Informational Note:	Whether or not DSpace should store a copy of the original sword deposit package. **NOTE:** this will cause the deposit process to run slightly slower, and will accelerate the rate at which the repository consumes disk space. BUT, it will also mean that the deposited packages are recoverable in their original form. It is strongly recommended, therefore, to leave this option turned on. When set to "true", this requires that the configuration option `upload.temp.dir` (in `dspace.cfg`) is set to a valid location.
Property:	`sword-server.bundle.name`
Example Value:	`sword-server.bundle.name = SWORD`
Informational Note:	The bundle name that SWORD should store incoming packages under if sword.keep-original-package is set to true. The default is "SWORD" if not value is set
Properties:	`sword-server.keep-package-on-fail` `sword-server.failed-package.dir`
Example Value:	`sword-server.keep-package-on-fail=true` `sword-server.failed-package.dir=${dspace.dir}/upload`
Informational Note:	In the event of package ingest failure, provide an option to store the package on the file system. The default is false.
Property:	`sword-server.identify-version`
Example Value:	`sword-server.identify-version = true`

Configuration File:	**[dspace]/config/modules/sword-server.cfg**
Informational Note:	Should the server identify the sword version in a deposit response. It is recommended to leave this unchanged.
Property:	`sword-server.on-behalf-of.enable`
Example Value:	`sword-server.on-behalf-of.enable = true`
Informational Note:	Should mediated deposit via sword be supported. If enabled, this will allow users to deposit content packages on behalf of other users.
Property:	`sword-server.restore-mode.enable`
Example Value:	`sword-server.restore-mode.enable = true`
Informational Note:	Should the sword server enable restore-mode when ingesting new packages. If this is enabled the item will be treated as a previously deleted item from the repository. If the item had previously been assigned a handle then that same handle will be restored to activity. If that item had not been previously assign a handle, then a new handle will be assigned.
Property:	`plugin.named.org.dspace.sword.SWORDingester`
Example Value:	`plugin.named.org.dspace.sword.SWORDIngester = \` `  org.dspace.sword.SWORDMETSIngester = http://purl.org/net/sword-types/METSDSpaceSIP \` `  org.dspace.sword.SimpleFileIngester = SimpleFileIngester`

Configuration File:	**[dspace]/config/modules/sword-server.cfg**
Informational Note:	Configure the plugins to process incoming packages. The form of this configuration is as per the Plugin Manager's Named Plugin documentation: `plugin.named.[interface] = [implementation] = [package format identifier]` (see `dspace.cfg`). Package ingesters should implement the SWORDIngester interface, and will be loaded when a package of the format specified above in: `sword-server.accept-packaging.[package format].identifier = [package format identifier]` is received. In the event that this is a simple file deposit, with no package format, then the class named by "SimpleFileIngester" will be loaded and executed where appropriate. This case will only occur when a single file is being deposited into an existing DSpace Item.

4.5.3.3 Deposit to SWORD Server

If you'd like to deposit content to your repository via the installed SWORD Server, you'll need to select a SWORD Client to do so.

- A variety of SWORDv1 Clients (in various languages/tools) are available off of http://swordapp.org/sword-v1/
- DSpacealso comes with an optional SWORDv1 Client(see page 176) which can be enabled to deposit content from one DSpace to another.
- An example SWORDv1 ZIP package is available in the DSpace Codebase at: https://github.com/DSpace/DSpace/tree/dspace-5_x/dspace-sword/example
- Finally, it's also possible to simply deposit a valid SWORD Zip package via common Linux commandline tools (e.g. curl). For example:

```
# Deposit a SWORD Zip package named "sword-package.zip" into a DSpace Collection (Handle 123456789/2)
 as user "test@dspace.org"
# (Please note that you WILL need to obviously modify the Collection location, user/password and
name of the SWORD package)

curl -i --data-binary "@sword-package.zip" -H "Content-Disposition:filename=sword-package.zip" -H
"Content-Type:application/zip" -H "X-Packaging:http://purl.org/net/sword-types/METSDSpaceSIP" -u
test@dspace.org:[password] http://[dspace.url]/sword/deposit/123456789/2

# Template 'curl' command:
#curl -i --data-binary "@[zip-package-name]" -H "Content-Disposition:filename=[zip-package-name]" -H
"Content-Type:application/zip" -H "X-Packaging:http://purl.org/net/sword-types/METSDSpaceSIP" -u
[user]:[password] http://[dspace.url]/sword/deposit/[collection-handle]
```

4.5.4 Exporting and Importing Community and Collection Hierarchy

4.5.4.1 Community and Collection Structure Importer

This Command-Line tool gives you the ability to import a community and collection structure directory from a source XML file.

Usage

Command used:	`[dspace]/bin/dspace structure-builder`
Java class:	`org.dspace.administer.StructBuilder`
Argument: short and long (if available) forms:	Description of the argument
-f	Source xml file. The presence of this argument engages import mode.
-o	Output xml file. **Required.** A copy of the input augmented with the Handles assigned to each new Community or Collection.
-e	Email of DSpace Administrator. **Required.**

XML Import Format

The administrator need to build the source xml document in the following format:

```
<import_structure>
        <community>
                <name>Community Name</name>
                <description>Descriptive text</description>
                <intro>Introductory text</intro>
                <copyright>Special copyright notice</copyright>
                <sidebar>Sidebar text</sidebar>
                <community>
                   <name>Sub Community Name</name>
                   <community> ...[ad infinitum]...
                   </community>
                </community>
                <collection>
                   <name>Collection Name</name>
                   <description>Descriptive text</description>
                   <intro>Introductory text</intro>
                   <copyright>Special copyright notice</copyright>
                   <sidebar>Sidebar text</sidebar>
                   <license>Special licence</license>
                   <provenance>Provenance information</provenance>
                </collection>
         </community>
</import_structure>
```

The resulting output document will be as follows:

```
<import_structure>
     <community identifier="123456789/1">
                <name>Community Name</name>
                <description>Descriptive text</description>
                <intro>Introductory text</intro>
                <copyright>Special copyright notice</copyright>
                <sidebar>Sidebar text</sidebar>
                <community identifier="123456789/2">
                        <name>Sub Community Name</name>
                        <community identifier="123456789/3"> ...[ad infinitum]...
                        </community>
                </community>
                <collection identifier="123456789/4">
                        <name>Collection Name</name>
                        <description>Descriptive text</description>
                        <intro>Introductory text</intro>
                        <copyright>Special copyright notice</copyright>
                        <sidebar>Sidebar text</sidebar>
                        <license>Special licence</license>
                        <provenance>Provenance information</provenance>
                </collection>
        </community>
</import_structure>
```

This command-line tool gives you the ability to import a community and collection structure directly from a source XML file. It is executed as follows:

```
[dspace]/bin/dspace structure-builder -f /path/to/source.xml -o path/to/output.xml -e admin@user.com
```

This will examine the contents of *source.xml*, import the structure into DSpace while logged in as the supplied administrator, and then output the same structure to the output file, but including the handle for each imported community and collection as an attribute.

4.5.4.2 Community and Collection Structure Exporter

This command-line tool writes the current Community and Collection structure into an XML document in the format which is read by the importer. See above for format details.

Usage

```
[dspace]/bin/dspace structure-builder [-h] [-?] -x -e <eperson> -o <output>
```

Argument: short and long (if available) forms:	**Description of the argument**
-x, --export	Export the current structure as XML. The presence of this argument engages export mode.
-e, --eperson *email or netid*	User who is manipulating the repository's structure. **Required.** This user's rights determine access to communities and collections.
-o, --output *file path*	The exported structure is written here. **Required.**
-h or -?	Help

4.5.5 Importing Items via basic bibliographic formats (Endnote, BibTex, RIS, TSV, CSV) and online services (OAI, arXiv, PubMed, CrossRef, CiNii)

⚠ In DSpace 7.0, the Biblio-Transformation-Engine (BTE) was removed in favor of Live Import from external sources(see page 274). All online services and bibliographic formats previously supported by BTE have been moved or are being moved to the External Sources framework.

4.5.5.1 Introduction

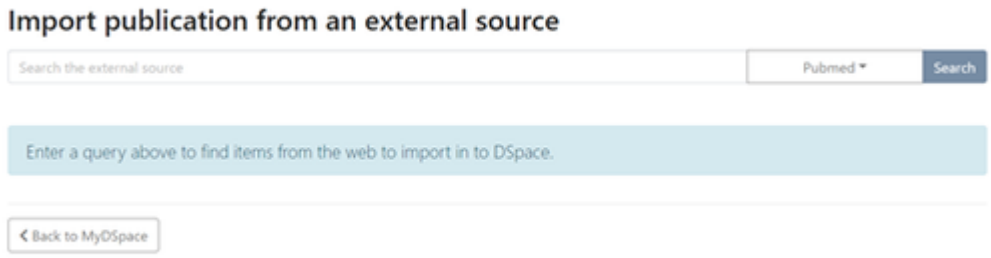
This documentation explains the features and the usage of the importer framework. The importer framework is built into both the REST API(see page 502) and User Interface(see page 367). Currently supported formats include:

- Drag & drop of Endnote, BibTex, RIS, TSV, CSV, arXiv, PubMed. From the MyDSpace page, dragging & dropping one of these files will start a new submission, extracting the metadata from the file.
- Import via ORCID, PubMed, Sherpa Journals, Sherpa Publishers. From the MyDSpace page, you can select to start a new submission by searching an external source.

4.5.5.2 Features

- Look up publications from remote sources.
- Support for multiple implementations.

4.5.5.3 Submitting starting from external sources



1. From the MyDSpace page a new submission can be started not only using the submission form but also automatically populating metadata, importing them from several online services.
2. After choosing the external source to import from and inserting a term in search bar, the system will show the list of matching results.
3. When selecting an item, the system will display the metadata to be imported, according to the configured mapping.
4. Clicking on “Start submission” the system will display the submission forms filled with the imported metadata.

4.5.5.4 Submitting starting from bibliographic file

1. From the MyDSpace page, drag & drop the bibliographic file (e.g. bibtex, endnote, etc) onto the file dropbox
2. Select the collection to submit to
3. Submission will be created, with the metadata parsed out of that bibliographic file. The bibliographic file will also be attached as a Bitstream (in case some fields could not be successfully parsed). You can choose to keep it or delete it.

4.5.5.5 More Information

More information on configuring metadata mappings for various import formats / services can be found in the Live Import from external sources(see page 274) documentation. See the "Editing Metadata Mapping" section.

4.5.6 Registering Bitstreams via Simple Archive Format

Registering is not Importing

The procedures below will **not import** the actual bitstreams into DSpace. They will merely inform DSpace of an existing location where these Bitstreams can be found. Please refer to Importing and Exporting Items via Simple Archive Format(see page 233) for information on importing metadata and bitstreams.

4.5.6.1 Overview

Registration is an alternate means of incorporating items, their metadata, and their bitstreams into DSpace by taking advantage of the bitstreams already being in storage accessible to DSpace. An example might be that there is a repository for existing digital assets. Rather than using the normal interactive ingest process or the batch import to furnish DSpace the metadata and to upload bitstreams, registration provides DSpace the metadata and the location of the bitstreams. DSpace uses a variation of the import tool to accomplish registration.

Accessible Storage

To register an item its bitstreams must reside on storage accessible to DSpace and therefore referenced by an asset store number in *dspace.cfg*. The configuration file *dspace.cfg* establishes one or more asset stores through the use of an integer asset store number. This number relates to a directory in the DSpace host's file system or a set of SRB account parameters. This asset store number is described in The *dspace.cfg* Configuration Properties File section and in the *dspace.cfg* file itself. The asset store number(s) used for registered items should generally not be the value of the *assetstore.incoming* property since it is unlikely that you will want to mix the bitstreams of normally ingested and imported items and registered items.

Registering Items Using the Item Importer

DSpace uses the same import tool(see page 233) that is used for batch import except that several variations are employed to support registration. The discussion that follows assumes familiarity with the import tool.

The DSpace Simple Archive Format(see page 233) for registration does not include the actual content files (bitstreams) being registered. The format is however a directory full of items to be registered, with a subdirectory per item. Each item directory contains a file for the item's descriptive metadata (*dublin_core.xml*) and a file listing the item's content files (*contents*), but not the actual content files themselves.

The *dublin_core.xml* file for item registration is exactly the same as for regular item import.

The *contents* file, like that for regular item import, lists the item's content files, one content file per line, but each line has the one of the following formats:

```
-r -s n -f filepath
-r -s n -f filepath\tbundle:bundlename
-r -s n -f filepath\tbundle:bundlename\tpermissions: -[r|w] 'group name'
-r -s n -f filepath\tbundle:bundlename\tpermissions: -[r|w] 'group  name'\tdescription: some text
```

where

- `-r` indicates this is a file to be registered
- `-s n` indicates the asset store number (*n*)
- `-f filepath` indicates the path and name of the content file to be registered (filepath)
- `\t` is a tab character
- `bundle:bundlename` is an optional bundle name
- `permissions: -[r|w] 'group name'` is an optional read or write permission that can be attached to the bitstream
- `description: some text` is an optional description field to add to the file
  The bundle, that is everything after the filepath, is optional and is normally not used.

The command line for registration is just like the one for regular import:

```
[dspace]/bin/dspace import -a -e joe@user.com -c collectionID -s items_dir -m mapfile
```

(or by using the long form)

```
[dspace]/bin/dspace import --add --eperson=joe@user.com --collection=collectionID --source=items_dir --
map=mapfile
```

The `--workflow` and `--test` flags will function as described in Importing Items(see page 0).

The `--delete` flag will function as described in Importing Items but the registered content files will not be removed from storage. See Deleting Registered Items.

The `--replace` flag will function as described in Importing Items but care should be taken to consider different cases and implications. With old items and new items being registered or ingested normally, there are four combinations or cases to consider. Foremost, an old registered item deleted from DSpace using `--replace` will not be removed from the storage. See Deleting Registered Items. where is resides. A new item added to DSpace using `--replace` will be ingested normally or will be registered depending on whether or not it is marked in the *contents* files with the `-r`.

Internal Identification and Retrieval of Registered Items

Once an item has been registered, superficially it is indistinguishable from items ingested interactively or by batch import. But internally there are some differences:

First, the randomly generated internal ID is not used because DSpace does not control the file path and name of the bitstream. Instead, the file path and name are that specified in the *contents* file.

Second, the *store_number* column of the bitstream database row contains the asset store number specified in the *contents* file.

Third, the *internal_id* column of the bitstream database row contains a leading flag (`-R`) followed by the registered file path and name. For example, `-Rfilepath` where *filepath* is the file path and name relative to the asset store corresponding to the asset store number. The asset store could be traditional storage in the DSpace server's file system or an SRB account.

Fourth, an MD5 checksum is calculated by reading the registered file if it is in local storage.

Registered items and their bitstreams can be retrieved transparently just like normally ingested items.

Exporting Registered Items

Registered items may be exported as described in Exporting Items. If so, the export directory will contain actual copies of the files being exported but the lines in the contents file will flag the files as registered. This means that if DSpace items are "round tripped" (see Transferring Items Between DSpace Instances(see page 0)) using the exporter and importer, the registered files in the export directory will again registered in DSpace instead of being uploaded and ingested normally.

Deleting Registered Items

If a registered item is deleted from DSpace, (either interactively or by using the `--delete` or `--replace` flags described in Importing and Exporting Items via Simple Archive Format(see page 233)) the item will disappear from DSpace but its registered content files will remain in place just as they were prior to registration. Bitstreams not registered but added by DSpace as part of registration, such as `license.txt` files, will be deleted.

4.5.7 Importing and Exporting Items via Simple Archive Format

4.5.7.1 Item Importer and Exporter

DSpace has a set of command line tools for importing and exporting items in batches, using the DSpace Simple Archive Format. Apart from the offered functionality, these tools serve as an example for users who aim to implement their own item importer.

DSpace Simple Archive Format

The basic concept behind the DSpace's Simple Archive Format is to create an archive, which is a directory containing one subdirectory per item. Each item directory contains a file for the item's descriptive metadata, and the files that make up the item.

```
archive_directory/
    item_000/
        dublin_core.xml         -- qualified Dublin Core metadata for metadata fields belonging to the 'dc'
 schema.
        metadata_[prefix].xml   -- metadata in another schema.  The prefix is the name of the schema as
registered with the metadata registry.
        contents                -- text file containing one line per filename.
        collections             -- text file that contains the handles of the collections the item will
belong to. Optional. Each handle in a row.
                                -- Collection in first line will be the owning collection.
        file_1.doc              -- files to be added as bitstreams to the item.
        file_2.pdf
    item_001/
        dublin_core.xml
        contents
        file_1.png
        ...
```

The `dublin_core.xml` or `metadata_[prefix].xml` file has the following format, where each metadata element has its own entry within a `<dcvalue>` tagset. There are currently three tag attributes available in the `<dcvalue>` tagset:

- `element` - the Dublin Core element
- `qualifier` - the element's qualifier
- `language` - (optional) ISO language code for element

```
<dublin_core>
    <dcvalue element="title" qualifier="none">A Tale of Two Cities</dcvalue>
    <dcvalue element="date" qualifier="issued">1990</dcvalue>
    <dcvalue element="title" qualifier="alternative" language="fr">J'aime les Printemps</dcvalue>
</dublin_core>
```

(Note the optional language tag attribute which notifies the system that the optional title is in French.)

Every metadata field used, must be registered via the metadata registry of the DSpace instance first. See Metadata and Bitstream Format Registries(see page 640).

Recommended Metadata

It is recommended to minimally provide "dc.title" and, where applicable, "dc.date.issued". Obviously you can (and should) provide much more detailed metadata about the Item. For more information see: Metadata Recommendations(see page 316).

The `contents` file simply enumerates, one file per line, the bitstream file names. See the following example:

```
file_1.doc
file_2.pdf
license
```

Please notice that the `license` is optional, and if you wish to have one included, you can place the file in the .../item_001/ directory, for example.

The bitstream name may optionally be followed by any of the following:

- `\tbundle:BUNDLENAME`
- `\tpermissions:PERMISSIONS`
- `\tdescription:DESCRIPTION`
- `\tprimary:true`

Where '\t' is the tab character.

'BUNDLENAME' is the name of the bundle to which the bitstream should be added. Without specifying the bundle, items will go into the default bundle, ORIGINAL.

'PERMISSIONS' is text with the following format: `-[r|w] 'group name'`

'DESCRIPTION' is text of the files description.

Primary is used to specify the primary bitstream.

Configuring `metadata_[prefix].xml` for a Different Schema

It is possible to use other Schema such as EAD, VRA Core, etc. Make sure you have defined the new schema in the DSpace Metadata Schema Registry.

1. Create a separate file for the other schema named `metadata_[prefix].xml`, where the `[prefix]` is replaced with the schema's prefix.
2. Inside the xml file use the same Dublin Core *syntax*, but on the `<dublin_core>` element include the attribute `schema=[prefix]`.
3. Here is an example for ETD metadata, which would be in the file `metadata_etd.xml`:

```
<?xml version="1.0" encoding="UTF-8"?>
<dublin_core schema="etd">
     <dcvalue element="degree" qualifier="department">Computer Science</dcvalue>
     <dcvalue element="degree" qualifier="level">Masters</dcvalue>
     <dcvalue element="degree" qualifier="grantor">Michigan Institute of Technology</dcvalue>
</dublin_core>
```

Importing Items

Before running the item importer over items previously exported from a DSpace instance, please first refer to Transferring Items Between DSpace Instances.

Command used:	`[dspace]/bin/dspace import`
Java class:	`org.dspace.app.itemimport.ItemImport`
Arguments short and (long) forms:	Description
`-a or --add`	Add items to DSpace ‡

`-r or --replace`	Replace items listed in mapfile ‡
`-d or --delete`	Delete items listed in mapfile ‡
`-s or --source`	Source of the items (directory)
`-c or --collection`	Destination Collection by its Handle or database ID
`-m or --mapfile`	Where the mapfile for items can be found (name and directory)
`-e or --eperson`	Email of eperson doing the importing
`-w or --workflow`	Send submission through collection's workflow
`-n or --notify`	Kicks off the email alerting of the item(s) has(have) been imported
`-t or --test`	Test run, do not actually import items
`-p or --template`	Apply the collection template
`-R or --resume`	Resume a failed import (Used on Add only)
`-h or --help`	Command help
`-z or --zip`	Name of zipfile

‡ These are mutually exclusive.

The item importer is able to batch import unlimited numbers of items for a particular collection using a very simple CLI command and 'arguments'.

Adding Items to a Collection from a directory

To add items to a collection, you gather the following information:

- eperson
- Collection ID (either Handle (e.g. 123456789/14) or Database ID (e.g. 2)
- Source directory where the items reside

- Mapfile. Since you don't have one, you need to determine where it will be (e.g. /Import/Col_14/mapfile)
  At the command line:

```
[dspace]/bin/dspace import --add --eperson=joe@user.com --collection=CollectionID --source=items_dir --
mapfile=mapfile
```

or by using the short form:

```
[dspace]/bin/dspace import -a -e joe@user.com -c CollectionID -s items_dir -m mapfile
```

The above command would cycle through the archive directory's items, import them, and then generate a map file which stores the mapping of item directories to item handles. **SAVE THIS MAP FILE.** You can use it for replacing or deleting (unimporting) the mapped items.

Testing. You can add `--test` (or `-t`) to the command to simulate the entire import process without actually doing the import. This is extremely useful for verifying your import files before doing the actual import.

Adding Items to a Collection from a zipfile

To add items to a collection, you gather the following information:

- eperson
- Collection ID (either Handle (e.g. 123456789/14) or Database ID (e.g. 2)
- Source directory where your zipfile containing the items resides
- Zipfile
- Mapfile. Since you don't have one, you need to determine where it will be (e.g. /Import/Col_14/mapfile)
  At the command line:

```
[dspace]/bin/dspace import --add --eperson=joe@user.com --collection=CollectionID --source=items_dir --
zip=filename.zip --mapfile=mapfile
```

or by using the short form:

```
[dspace]/bin/dspace import -a -e joe@user.com -c CollectionID -s items_dir -z filename.zip -m mapfile
```

The above command would unpack the zipfile, cycle through the archive directory's items, import them, and then generate a map file which stores the mapping of item directories to item handles. **SAVE THIS MAP FILE.** You can use it for replacing or deleting (unimporting) the mapped items.

Testing. You can add `--test` (or `-t`) to the command to simulate the entire import process without actually doing the import. This is extremely useful for verifying your import files before doing the actual import.

Replacing Items in a Collection

Replacing existing items is relatively easy. Remember that mapfile you saved above? Now you will use it. The command (in short form):

```
[dspace]/bin/dspace import -r -e joe@user.com -c collectionID -s items_dir -m mapfile
```

Long form:

```
[dspace]/bin/dspace import --replace --eperson=joe@user.com --collection=collectionID --source=items_dire
--mapfile=mapfile
```

Deleting or Unimporting Items in a Collection

You are able to unimport or delete items provided you have the mapfile. Remember that mapfile you saved above? The command is (in short form):

```
[dspace]/bin/dspace import -e joe@user.com -d -m mapfile
```

In long form:

```
[dspace]/bin/dspace import --eperson=joe@user.com --delete --mapfile mapfile
```

Other Options

- **Workflow**. The importer usually bypasses any workflow assigned to a collection. But add the `--workflow` (`-w`) argument will route the imported items through the workflow system.
- **Templates**. If you have templates that have constant data and you wish to apply that data during batch importing, add the `--template` (`-p`) argument.
- **Resume**. If, during importing, you have an error and the import is aborted, you can use the `--resume` (`-R`) flag to resume the import where you left off after you fix the error.
- **Specifying the owning collection on a per-item basis from the command line administration tool**
  If you omit the -c flag, which is otherwise mandatory, the ItemImporter searches for a file named "collections" in each item directory. This file should contain a list of collections, one per line, specified either by their handle, or by their internal db id. The ItemImporter then will put the item in each of the specified collections. The owning collection is the collection specified in the first line of the collections file. If both the -c flag is specified and the collections file exists in the item directory, the ItemImporter will ignore the collections file and will put the item in the collection specified on the command line.
  Since the collections file can differ between item directories, this gives you more fine-grained control of the process of batch adding items to collections.

UI Batch Import

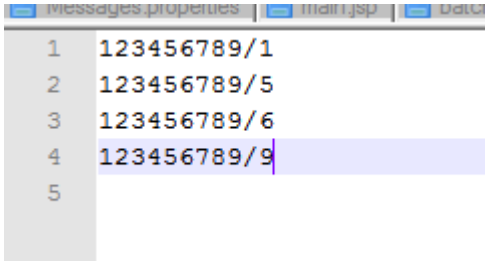
> **DSpace 7.0 does not yet support**
>
> Batch Import via the UI is not available in DSpace 7.0. It is scheduled to be restored in a later 7.x release (currently 7.1), see DSpace Release 7.0 Status[238]. The below screenshots/process are outdated and will need updating once this feature is rebuilt in 7.x.

Batch import can also take place via the Administrator's UI. The steps to follow are:

238 https://wiki.lyrasis.org/display/DSPACE/DSpace+Release+7.0+Status

A. Prepare the data

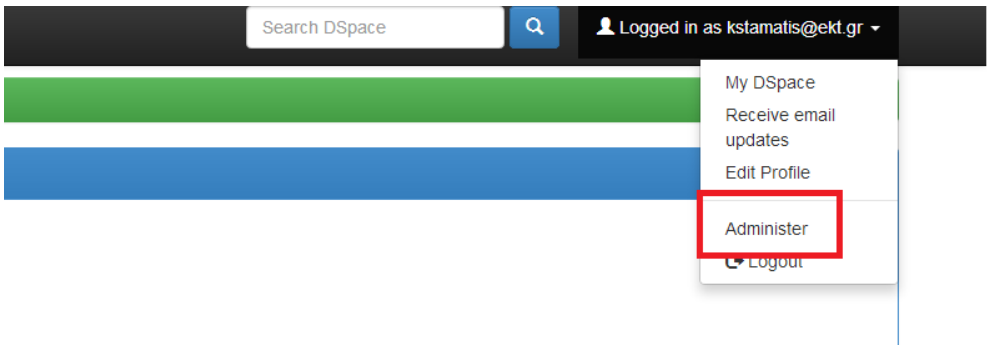
1. Items, i.e. the metadata and their bitstreams, must be in the Simple Archive Format described earlier in this chapter. Thus, for each item there must be a separate directory that contains the corresponding files of the specific item.
2. Moreover, in each item directory, there can be another file that describes the collection or the collections that this item will be added to. The name of this file must be "collections" and it is optional. It has the following format:



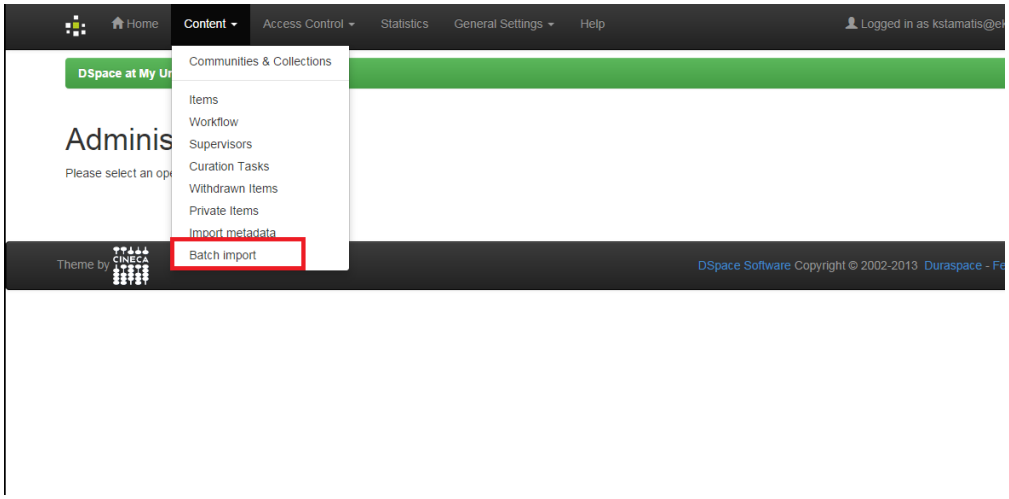
   Each line contains the handle of the collection. The collection in the first line is the owning collection while the rest are the other collections that the item should belong to.
3. Compress the item directories into a zip file. Please note that you need to zip the actual item directories and not just the directory that contains the item directories. Thus, the final zip file must directly contain the item directories.
4. Place the zip file in a public domain URL, like Dropbox or Google Drive or wherever you have access to do so. Since such a zip file can be very big in size, the batch import UI needs the URL to download it for a public location rather than just upload it and get a timeout exception

B. Import the items via the UI

1. Login as an administrator.
2. Find the menu on the top right of the page, and select the "Administer" option.

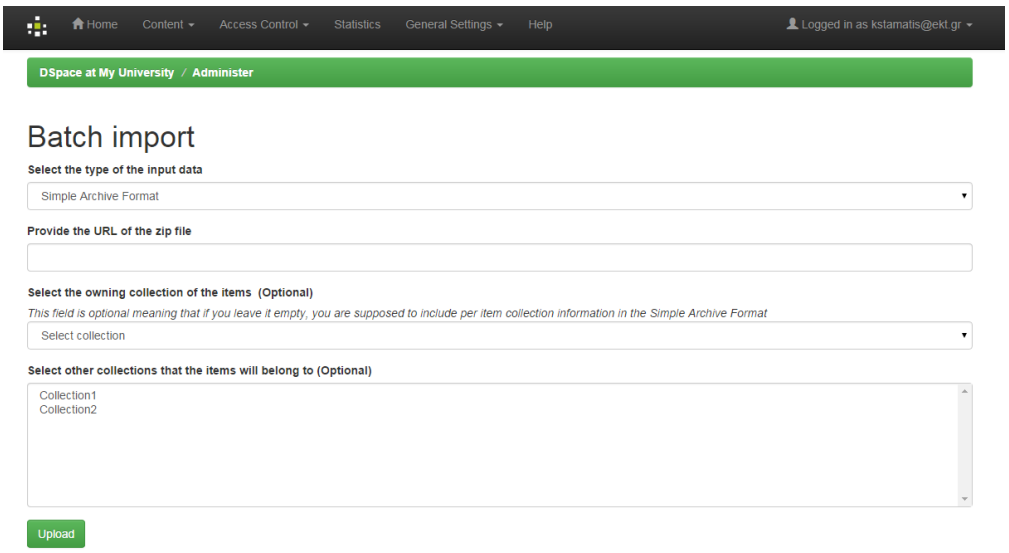


3. Select the "Batch Import" option from the "Content" drop down menu on the top of the page.



4. Fill in the form that appears as follows:

- Field #1: select the type of the input data that you want to batch import. Be sure to select "Simple Archive Format" in this drop down menu.
- Field #2: Copy/Paste the public URL where the zip file mentioned earlier is located.
- Filed #3: Select the owning collection of the items you are importing. This field is optional, meaning that if you leave it empty, you must include per item collection information (via the "collections" file mentioned before) in the Simple Archive Format.
- Field #4: Select the other collections the item will belong to. You can select more than one collection by just holding down the Ctrl key on your keyboard. If you select the owning collection in this multiselect input control, it will be ignored at the very end.



Comments:

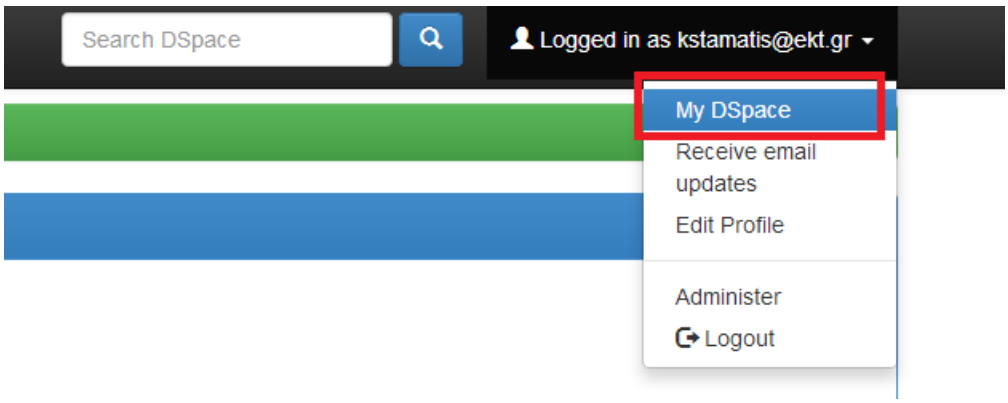
1) If you select an owning collection from this form, then the "collections" file that may be included in the item will be ignored.

2) If you do not specify an owning collection, and for some items no "collections" file exists in the item directory, then the item will not be imported in DSpace

Finally, when you submit the form you will receive a message informing you that the import process is being executed in the background (since it may take long). At the end, you will receive a success or failure email (to the email address of your DSpace account) informing you of the status of the import.

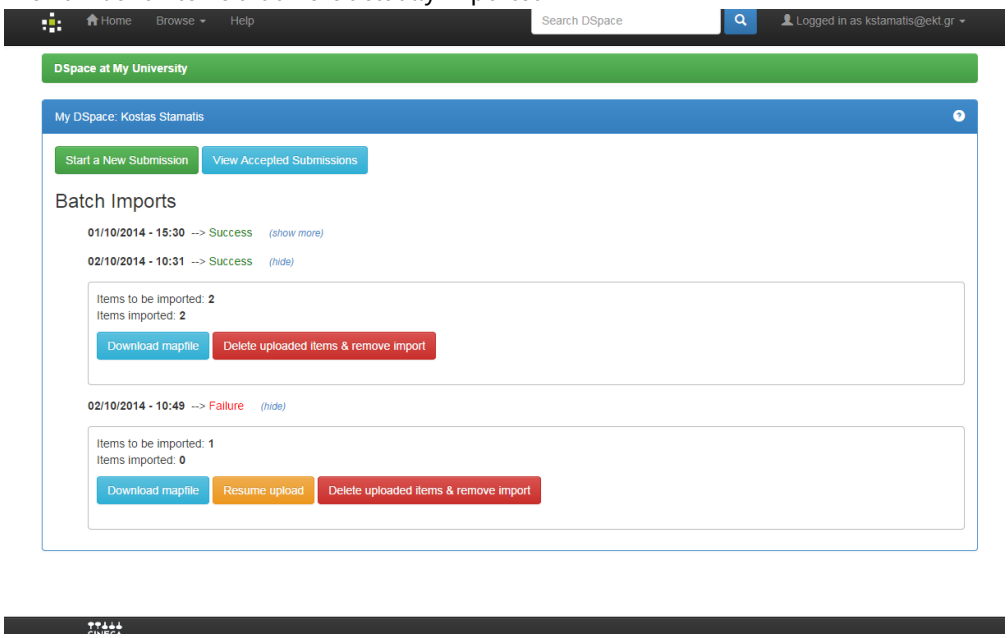
C. View past batch imports (that have be done via the UI)

1. Login.
2. Visit "My DSpace" page.



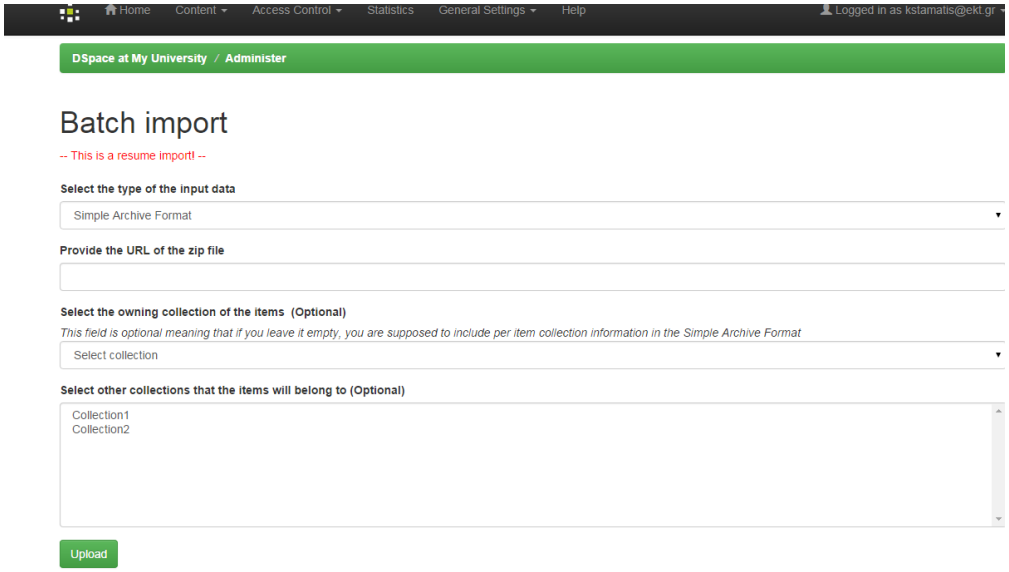
3. On the next page, you can see the history of batch imports. For each import, the following information is available:

   The status of the batch import (success or failure)
   The number of items that the user tried to import
   The number of items that were actually imported



Moreover, the user can take the following actions:

Download the map file that was produced during the import. This file contains a list of items that were imported with the corresponding handle assigned to them by DSpace.
Delete the imported items. Everything that was imported will be deleted (including the history directory in the "[dspace]/import" directory)
In case of failure, the user can "Resume" the import. The user is taken to the upload form again, but the system recognizes the initial import (and the map file) in order to resume the old import. There is a red label in the form that informs the user about the "Resume" form.



Exporting Items

The item exporter can export a single item or a collection of items, and creates a DSpace simple archive in the aforementioned format[239] for each exported item. The items are exported in a sequential order in which they are retrieved from the database. As a consequence, the sequence numbers of the item subdirectories (item_000, item_001) are not related to DSpace handle or item IDs.

Command used:	`[dspace]/bin/dspace export`
Java class:	*org.dspace.app.itemexport.ItemExport*
Arguments short and (long) forms:	Description
`-t` or `--type`	Type of export. *COLLECTION* will inform the program you want the whole collection. *ITEM* will be only the specific item. (You will actually key in the keywords in all caps. See examples below.)

93 https://wiki.duraspace.org/display/DSDOC4x/Importing+and+Exporting+Items+via+Simple+Archive+Format#ImportingandExportingItemsviaSimpleArchiveFormat-DSpaceSimpleArchiveFormat

`-i` or `--id`	The ID or Handle of the Collection or Item to export.
`-d` or `--dest`	The destination path where you want the file of items to be placed.
`-n` or `--number`	Sequence number to begin with. Whatever number you give, this will be the name of the first directory created for your export. The layout of the export directory is the same as the layout used for import.
`-m` or `--migrate`	Export the item/collection for migration. This will remove the handle and any other metadata that will be re-created in the new instance of DSpace.
-x or --exclude-bitstreams	Do not export bitstreams. See the usage scenario below.
`-h` or `--help`	Brief Help.

Exporting a Collection

The CLI command to export the items of a collection:

```
[dspace]/bin/dspace export --type=COLLECTION --id=collectionID_or_handle --dest=/path/to/destination --
number=seq_num
```

Short form:

```
[dspace]/bin/dspace export -t COLLECTION -i collectionID_or_handle -d /path/to/destination -n seq_num
```

The keyword *COLLECTION* means that you intend to export an entire collection. The ID can either be the database ID or the handle. The exporter will begin numbering the simple archives with the sequence number that you supply.

Exporting a Single Item

To export a single item use the keyword *ITEM* and give the item ID as an argument:

```
[dspace]/bin/dspace export --type=ITEM --id=itemID_or_handle --dest=/path/to/destination --number=seq_num
```

Short form:

```
[dspace]/bin/dspace export -t ITEM -i itemID_or_handle -d /path/to/destination -n seq_num
```

Each exported item will have an additional file in its directory, named "handle". This will contain the handle that was assigned to the item, and this file will be read by the importer so that items exported and then imported to another machine will retain the item's original handle.

The `-m` Argument

Using the `-m` argument will export the item/collection and also perform the migration step. It will perform the same process that the next section Exchanging Content Between Repositories(see page 178) performs. We recommend that section to be read in conjunction with this flag being used.

The `-x` Argument

Using the -x argument will do the standard export except for the bitstreams which will not be exported. If you have full SAF without bitstreams and you have the bitstreams archive (which might have been imported into DSpace earlier) somewhere near, you could symlink[240] original archive files into SAF directories and have an exported collection which almost doesn't occupy any space but otherwise is identical to the exported collection (i.e. could be imported into DSpace). In case of huge collections -x mode might be substantially faster than full export.

4.5.8 Importing and Exporting Content via Packages

4.5.8.1 Package Importer and Exporter

This command-line tool gives you access to the Packager plugins. It can *ingest* a package to create a new DSpace Object (Community, Collection or Item), or *disseminate* a DSpace Object as a package.

To see all the options, invoke it as:

```
[dspace]/bin/dspace packager --help
```

This mode also displays a list of the names of package ingestion and dissemination plugins that are currently installed in your DSpace. Each Packager plugin also may allow for custom options, which may provide you more control over how a package is imported or exported. You can see a listing of all specific packager options by invoking `--help` (or `-h`) with the `--type` (or `-t`) option:

```
[dspace]/bin/dspace packager --help --type METS
```

240 http://en.wikipedia.org/wiki/Symbolic_link

The above example will display the normal help message, while also listing any additional options available to the "METS" packager plugin.

Supported Package Formats

DSpace comes with several pre-configured package ingestion and dissemination plugins, which allow you to import/export content in a variety of formats.

Pre-Configured Submission Package (SIP) Types

- AIP - Ingests content which is in the DSpace Archival Information Package (AIP) format(see page 439). This is used as part of the DSpace AIP Backup and Restore(see page 411) process
- DSPACE-ROLES - Ingests DSpace users/groups in the DSPACE-ROLES XML Schema(see page 452). This is primarily used by the DSpace AIP Backup and Restore(see page 411) process to ingest/replace DSpace Users & Groups.
- METS - Ingests content which is in the DSpace METS SIP format[241]
- PDF - Ingests a single PDF file (where basic metadata is extracted from the file properties in the PDF Document).

Pre-Configured Dissemination Package (DIP) Types

- AIP - Exports content which is in the DSpace Archival Information Package (AIP) format(see page 439). This is used as part of the DSpace AIP Backup and Restore(see page 411) process
- DSPACE-ROLES - Exports DSpace users/groups in the DSPACE-ROLES XML Schema(see page 452). This is primarily used by the DSpace AIP Backup and Restore(see page 411) process to export DSpace Users & Groups.
- METS - Exports content in the DSpace METS SIP format[242]

For a list of all package ingestion and dissemination plugins that are currently installed in your DSpace, you can execute:

```
[dspace]/bin/dspace packager --help
```

Some packages ingestion and dissemination plugins also have custom options/parameters. For example, to see a listing of the custom options for the "METS" plugin, you can execute:

```
[dspace]/bin/dspace packager --help --type METS
```

Ingesting

Ingestion Modes & Options

When ingesting packages DSpace supports several different "modes". (Please note that not all packager plugins may support all modes of ingestion)

1. Submit/Ingest Mode (`-s` option, default) – submit package to DSpace in order to create a new object(s)
2. Restore Mode (`-r` option) – restore pre-existing object(s) in DSpace based on package(s). This also attempts to restore all handles and relationships (parent/child objects). This is a specialized type of "submit", where the object is created with a known Handle and known relationships.

241 https://wiki.lyrasis.org/display/DSPACE/DSpaceMETSSIPProfile
242 https://wiki.lyrasis.org/display/DSPACE/DSpaceMETSSIPProfile

3. Replace Mode (`-r -f` option) – replace existing object(s) in DSpace based on package(s). This also attempts to restore all handles and relationships (parent/child objects). This is a specialized type of "restore" where the contents of existing object(s) is replaced by the contents in the AIP(s). By default, if a normal "restore" finds the object already exists, it will back out (i.e. rollback all changes) and report which object already exists.

Ingesting a Single Package
To ingest a single package from a file, give the command:

```
[dspace]/bin/dspace packager -e [user-email] -p [parent-handle] -t [packager-name] /full/path/to/package
```

Where *[user-email]* is the e-mail address of the E-Person under whose authority this runs; *[parent-handle]* is the Handle of the Parent Object into which the package is ingested, *[packager-name]* is the plugin name of the package ingester to use, and */full/path/to/package* is the path to the file to ingest (or *"-"* to read from the standard input).

Here is an example that loads a PDF file with internal metadata as a package:

```
[dspace]/bin/dspace packager -e admin@myu.edu -p 4321/10 -t PDF thesis.pdf
```

This example takes the result of retrieving a URL and ingests it:

```
wget -O - http://alum.mit.edu/jarandom/my-thesis.pdf | [dspace]/bin/dspace packager -e admin@myu.edu -p
4321/10  -t PDF -
```

Ingesting Multiple Packages at Once
Some Packager plugins support bulk ingest functionality using the `--all` (or `-a`) flag. When `--all` is used, the packager will attempt to ingest all child packages referenced by the initial package (and continue on recursively). Some examples follow:

- For a Site-based package - this would ingest **all** Communities, Collections & Items based on the located package files
- For a Community-based package - this would ingest that Community and all SubCommunities, Collections and Items based on the located package files
- For a Collection - this would ingest that Collection and all contained Items based on the located package files
- For an Item – this just ingest the Item (including all Bitstreams & Bundles) based on the package file.

Here is a basic example of a bulk ingest 'packager' command template:

```
[dspace]/bin/dspace packager -s -a -t AIP -e <eperson> -p <parent-handle> <file-path>
```

for example:

```
[dspace]/bin/dspace packager -s -a -t AIP -e admin@myu.edu -p 4321/12 collection-aip.zip
```

The above command will ingest the package named "collection-aip.zip" as a child of the specified Parent Object (handle="4321/12"). The resulting object is assigned a new Handle (since `-s` is specified). In addition, any child packages directly referenced by "collection-aip.zip" are also recursively ingested (a new Handle is also assigned for each child AIP).

Not All Packagers Support Bulk Ingest

Because the packager plugin must know how to locate all child packages from an initial package file, not all plugins can support bulk ingest. Currently, in DSpace the following Packager Plugins support bulk ingest capabilities:

- METS Packager Plugin
- AIP Packager Plugin(see page 411)

Restoring/Replacing using Packages

Restoring is slightly different than just **ingesting**. When restoring, the packager makes every attempt to restore the object as it **used to be** (including its handle, parent object, etc.).

There are currently three restore modes:

1. Default Restore Mode (`-r`) = Attempt to restore object (and optionally children). Rollback all changes if any object is found to already exist.
2. Restore, Keep Existing Mode (`-r -k`) = Attempt to restore object (and optionally children). If an object is found to already exist, skip over it (and all children objects), and continue to restore all other non-existing objects.
3. Force Replace Mode (`-r -f`) = Restore an object (and optionally children) and **overwrite** any existing objects in DSpace. Therefore, if an object is found to already exist in DSpace, its contents are replaced by the contents of the package. *WARNING: This mode is potentially dangerous as it will permanently destroy any object contents that do not currently exist in the package. You may want to first perform a backup, unless you are sure you know what you are doing!*

Default Restore Mode
By default, the restore mode (`-r` option) will rollback all changes if any object is found to already exist. The user will be informed if which object already exists within their DSpace installation.

Use this 'packager' command template:

```
[dspace]/bin/dspace packager -r -t AIP -e <eperson> <file-path>
```

For example:

```
[dspace]/bin/dspace packager -r -t AIP -e admin@myu.edu aip4567.zip
```

Notice that unlike `-s` option (for submission/ingesting), the `-r` option does not require the Parent Object (`-p` option) to be specified if it can be determined from the package itself.

In the above example, the package "aip4567.zip" is restored to the DSpace installation with the Handle provided within the package itself (and added as a child of the parent object specified within the package itself). If the object is found to already exist, all changes are rolled back (i.e. nothing is restored to DSpace)

Restore, Keep Existing Mode
When the "Keep Existing" flag (`-k` option) is specified, the restore will attempt to skip over any objects found to already exist. It will report to the user that the object was found to exist (and was not modified or changed). It will then continue to restore all objects which do not already exist. This flag is most useful when attempting a bulk restore (using the `--all` (or `-a`) option.

One special case to note: If a Collection or Community is found to already exist, its child objects are also skipped over. So, this mode will not auto-restore items to an existing Collection.

Here's an example of how to use this 'packager' command:

```
[dspace]/bin/dspace packager -r -a -k -t AIP -e <eperson> <file-path>
```

For example:

```
[dspace]/bin/dspace packager -r -a -k -t AIP -e admin@myu.edu aip4567.zip
```

In the above example, the package "aip4567.zip" is restored to the DSpace installation with the Handle provided within the package itself (and added as a child of the parent object specified within the package itself). In addition, any child packages referenced by "aip4567.zip" are also recursively restored (the `-a` option specifies to also restore all child pacakges). They are also restored with the Handles & Parent Objects provided with their package. If any object is found to already exist, it is skipped over (child objects are also skipped). All non-existing objects are restored.

Force Replace Mode
When the "Force Replace" flag (`-f` option) is specified, the restore will **overwrite** any objects found to already exist in DSpace. In other words, existing content is deleted and then replaced by the contents of the package(s).

Potential for Data Loss

Because this mode actually **destroys** existing content in DSpace, it is potentially dangerous and may result in data loss! It is recommended to always perform a full backup (assetstore files & database) before attempting to replace any existing object(s) in DSpace.

Here's an example of how to use this 'packager' command:

```
[dspace]/bin/dspace packager -r -f -t AIP -e <eperson> <file-path>
```

For example:

```
[dspace]/bin/dspace packager -r -f -t AIP -e admin@myu.edu aip4567.zip
```

In the above example, the package "aip4567.zip" is restored to the DSpace installation with the Handle provided within the package itself (and added as a child of the parent object specified within the package itself). In addition, any child packages referenced by "aip4567.zip" are also recursively ingested. They are also restored with the Handles & Parent Objects provided with their package. *If any object is found to already exist, its contents are replaced by the contents of the appropriate package.*

If any error occurs, the script attempts to rollback the entire replacement process.

Disseminating

Disseminating a Single Object

To disseminate a single object as a package, give the command:

```
[dspace]/bin/dspace packager -d -e [user-email] -i [handle] -t [packager-name] [file-path]
```

Where *[user-email]* is the e-mail address of the E-Person under whose authority this runs; *[handle]* is the Handle of the Object to disseminate; *[packager-name]* is the plugin name of the package disseminator to use; and *[file-path]* is the path to the file to create (or "-" to write to the standard output). For example:

```
[dspace]/bin/dspace packager -d -e admin@myu.edu -i 4321/4567 -t METS 4567.zip
```

The above code will export the object of the given handle (4321/4567) into a METS file named "4567.zip".

Disseminating Multiple Objects at Once

To export an object hierarchy, use the `-a` (or `--all`) package parameter.

For example, use this 'packager' command template:

```
[dspace]/bin/dspace packager -d -a -e [user-email] -i [handle] -t [packager-name][file-path]
```

for example:

```
[dspace]/bin/dspace packager -d -a -t METS -e admin@myu.edu -i 4321/4567 4567.zip
```

The above code will export the object of the given handle (4321/4567) into a METS file named "4567.zip". In addition it would export all children objects to the same directory as the "4567.zip" file.

Archival Information Packages (AIPs)

Since DSpace 1.7, DSpace can backup and restore all of its contents as a set of AIP Files(see page 439). This includes all Communities, Collections, Items, Groups and People in the system.

This feature came out of a requirement for DSpace to better integrate with DuraCloud[243], and other backup storage systems. One of these requirements is to be able to essentially "backup" local DSpace contents into the cloud (as a type of offsite backup), and "restore" those contents at a later time.

Essentially, this means DSpace can export the entire hierarchy (i.e. bitstreams, metadata and relationships between Communities/Collections/Items) into a relatively standard format (a METS-based, AIP format(see page 439)). This entire hierarchy can also be re-imported into DSpace in the same format (essentially a restore of that content in the same or different DSpace installation).

For more information, see the section on AIP backup & Restore for DSpace(see page 411).

243 https://duracloud.org

METS packages

Since DSpace 1.4 release, the software includes a package disseminator and matching ingester for the DSpace METS SIP (Submission Information Package) format. They were created to help end users prepare sets of digital resources and metadata for submission to the archive using well-defined standards such as METS[244], MODS[245], and PREMIS[246]. The plugin name is *METS* by default, and it uses MODS for descriptive metadata.

The DSpace METS SIP profile is available at: DSpaceMETSSIPProfile[247]

4.5.9 Configurable Workflow

4.5.9.1 Introduction

As of DSpace 7, Configurable Workflow is the only workflow system available in DSpace. It has fully replaced the older "traditional/basic workflow" system. One major difference is that Configurable Workflow is dynamic – if a user is added to a workflow approval task *after* a workflow has already begun, they will immediately get access to any existing items in workflow. Previously, this was not possible in the "traditional" workflow system.

244 http://www.loc.gov/standards/mets/
245 http://www.loc.gov/standards/mods/
246 http://www.loc.gov/standards/premis/
247 https://wiki.lyrasis.org/display/DSPACE/DSpaceMETSSIPProfile

The primary focus of the workflow framework is to create a more flexible solution for the administrator to configure, and even to allow an application developer to implement custom steps, which may be configured in the workflow for the collection through a simple configuration file. The concept behind this approach was modeled on the configurable submission system already present in DSpace.

For more information, see the Configurable Workflow Introductory Video[248]

> **Workflow Data Migration**
>
> You will also need to follow the Data Migration Procedure(see page 0) below.

4.5.9.2 Data Migration

Workflowitem conversion/migration scripts

Depending on the workflow that is used by a DSpace installation, different scripts can be used when migrating to the new workflow.

Automatic migration

As part of the upgrade to DSpace 7 or above, all your old policies, roles, tasks and workflowitems will be automatically updated from the original workflow to the Configurable Workflow framework. This is done via this command:

```
[dspace]/bin/dspace database migrate ignored
```

The "ignored" parameter will tell DSpace to run any previously-ignored migrations on your database. As the Configurable Workflow migrations have existed in the DSpace codebase for some time, this is the only way to force them to be run.

For more information on the "database migrate" command, please see Database Utilities(see page 462).

Java based migration

In case your DSpace installation uses a customized version of the workflow, the migration script might not work properly and a different approach is recommended. Therefore, an additional Java based script has been created that restarts the workflow for all the workflowitems that exist in the original workflow framework. The script will take all the existing workflowitems and place them in the first step of the configurable workflow framework thereby taking into account the XML configuration that exists at that time for the collection to which the item has been submitted. This script can also be used to restart the workflow for workflowitems in the original workflow but not to restart the workflow for items in the configurable workflow.

To execute the script, run the following CLI command:

```
[dspace]/bin/dspace dsrun org.dspace.xmlworkflow.migration.RestartWorkflow -e admin@myrespository.org
```

The following arguments can be specified when running the script:

248 http://youtu.be/_Z52gne55so

- -e: specifies the username of an administrator user
- -n: if sending submissions through the workflow, send notification emails
- -p: the provenance description to be added to the item
- -h: help

4.5.9.3 Configuration

Main workflow configuration

> As of DSpace 7, the `workflow.xml` configuration file has been migrated to use Spring Bean syntax (instead of a custom XML format). The structure of this XML has changed. If you need help migrating your old `workflow.xml` file (which started with a `<wf-config>` tag) to the new format (using `<bean>` tags), an XSLT script is available: workflow-migration.xsl[249]

The workflow main configuration can be found in the workflow.xml file, located in **`[dspace]/config/spring/api/workflow.xml`**. An example of this workflow configuration file can be found below.

249 https://wiki.lyrasis.org/download/attachments/104566678/workflow-migration.xsl?api=v2&modificationDate=1622152622820&version=1

```
<beans>
    <bean class="org.dspace.xmlworkflow.XmlWorkflowFactoryImpl">
        <property name="workflowMapping">
            <util:map>
                <entry key="defaultWorkflow" value-ref="defaultWorkflow"/>
<!--            <entry key="123456789/4" value-ref="selectSingleReviewer"/>-->
<!--            <entry key="123456789/5" value-ref="scoreReview"/>-->
            </util:map>
        </property>
    </bean>

    <!--Standard DSpace workflow-->
    <bean name="defaultWorkflow" class="org.dspace.xmlworkflow.state.Workflow">
        <property name="firstStep" ref="reviewstep"/>
        <property name="steps">
            <util:list>
                <ref bean="reviewstep"/>
                <ref bean="editstep"/>
                <ref bean="finaleditstep"/>
            </util:list>
        </property>
    </bean>

    <bean id="{workflow.id}"
          class="org.dspace.xmlworkflow.state.Workflow">
        <!-- Another workflow configuration-->
    </bean>

    <!-- Role beans.  See below. -->

    <!-- Step beans.  See below. -->

</beans>
```

workflowFactory bean (org.dspace.xmlworkflow.XmlWorkflowFactoryImpl)

The workflow map contains a mapping between collections in DSpace and a workflow configuration, and is defined by the `workflowMapping` property of the workflow factory. Similar to the configuration of the submission process, the mapping can be done based on the handle of the collection. The mapping with "defaultWorkflow" as the value for the collection mapping, will be used for the collections not occurring in other mapping tags. Each mapping is defined by a "`entry`" element with two attributes:

- key: can either be a collection handle or "defaultWorkflow"
- value-ref: the value of this attribute points to one of the workflow configurations defined by the "Workflow" beans

workflow beans (org.dspace.xmlworkflow.state.Workflow)

The workflow bean is a repeatable XML element and represents one workflow process. It requires the following:

- "`name`" attribute: a unique name used for the identification of the workflow and used in the workflow to collection mapping

- "`firstStep`" property: the identifier of the first step of the workflow. This step will be the entry point of this workflow-process. When a new item has been committed to a collection that uses this workflow, the step configured in the "`firstStep`" property will he the first step the item will go through.
- "`steps`" property: a list of all steps within this workflow (in the order they will be processed).

role beans (org.dspace.xmlworkflow.Role)

Each workflow step has defined "role" property. A role represents one or more existing DSpace EPersons or Groups and can be used to assign them to one or more steps in the workflow process. One role is represented by one "role" bean and has the following:

- "`id`" attribute: a unique identifier (in one workflow process) for the role
- "`description`" property: optional attribute to describe the role
- "`scope`" property: optional attribute that is used to find our group and must have one of the following values, which are defined as constant fields of `org.dspace.xmlworkflow.Role.Scope`:
  - COLLECTION: The collection value specifies that the group will be configured at the level of the collection. This type of groups is the same as the type that existed in the original workflow system. In case no value is specified for the scope attribute, the workflow framework assumes the role is a collection role.
  - REPOSITORY: The repository scope uses groups that are defined at repository level in DSpace. The name attribute should exactly match the name of a group in DSpace.
  - ITEM: The item scope assumes that a different action in the workflow will assign a number of EPersons or Groups to a specific workflow-item in order to perform a step. These assignees can be different for each workflow item.
- "`name`" property: The name specified in the name attribute of a role will be used to lookup an eperson group in DSpace. The lookup will depend on the scope specified in the "`scope`" attribute:
  - COLLECTION: The workflow framework will look for a group containing the name specified in the name attribute and the ID of the collection for which this role is used.
  - REPOSITORY: The workflow framework will look for a group with the same name as the name specified in the `name` attribute.
  - ITEM: in case the item scope is selected, the name of the role attribute is not required.

```
<bean id="reviewer" class="org.dspace.xmlworkflow.Role">
    <property name="scope" value="#{ T(org.dspace.xmlworkflow.Role.Scope).COLLECTION}"/>
    <property name="name" value="Reviewer"/>
    <property name="description" value="The people responsible for this step are able to edit the metadata
of incoming submissions, and then accept or reject them."/>
</bean>
```

step beans (org.dspace.xmlworkflow.state.Step)

The step element represents one step in the workflow process. A step represents a number of actions that must be executed by one specified role. In case no `role` attribute is specified, the workflow framework assumes that the DSpace system is responsible for the execution of the step and that no user interface will be available for each of the actions in this step. The step element has the following in order to further configure it:

- "`name`" attribute: The name attribute specifies a unique identifier for the step. This identifier will be used when configuring other steps in order to point to this step. This identifier can also be used when configuring the start step of the workflow item.
- "`userSelectionMethod`" property: This attribute defines the `UserSelectionAction` that will be used to determine how to attach users to this step for a workflow-item. The value of this attribute must refer to the

identifier of an action bean in the workflow-actions.xml. Examples of the user attachment to a step are the currently used system of a task pool or as an alternative directly assigning a user to a task.
- "`role`" property: optional attribute that must point to the `id` attribute of a role element specified for the workflow. This role will be used to define the epersons and groups used by the `userSelectionMethod`.
- RequiredUsers

```
<bean name="reviewstep" class="org.dspace.xmlworkflow.state.Step">
    <property name="userSelectionMethod" ref="claimaction"/>
    <property name="role" ref="reviewer"/>
    <property name="outcomes">
        <util:map>
            <entry key="#{ T(org.dspace.xmlworkflow.state.actions.ActionResult).OUTCOME_COMPLETE}"
                   value-ref="editstep"/>
        </util:map>
    </property>
    <property name="actions">
        <util:list>
            <ref bean="reviewaction"/>
        </util:list>
    </property>
</bean>
```

Each step contains a number of actions that the workflow item will go through. In case the action has a user interface, the users responsible for the exectution of this step will have to execute these actions before the workflow item can proceed to the next action or the end of the step.

There is also an optional subsection that can be defined for a step part called "`outcomes`". This can be used to define outcomes for the step that differ from the one specified in the nextStep attribute. Each action returns an integer depending on the result of the action. The default value is "0" and will make the workflow item proceed to the next action or to the end of the step.
In case an action returns a different outcome than the default "0", the alternative outcomes will be used to lookup the next step. The "`outcomes`" element contains a number of steps, each having a status attribute. This status attribute defines the return value of an action. The value of the element will be used to lookup the next step the workflow item will go through in case an action returns that specified status.

Workflow actions configuration

API configuration

The workflow actions configuration is located in the `[dspace]/config/spring/api/` directory and is named "`workflow-actions.xml`". This configuration file describes the different Action Java classes that are used by the workflow framework. Because the workflow framework uses Spring framework for loading these action classes, this configuration file contains Spring configuration.

This file contains the beans for the actions and user selection methods referred to in the `workflow.xml`. In order for the workflow framework to work properly, each of the required actions must be part of this configuration.

```
<beans
    xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:util="http://www.springframework.org/schema/util"
    xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/
beans/spring-beans-2.0.xsd
                        http://www.springframework.org/schema/util http://www.springframework.org/
schema/util/spring-util-2.0.xsd">

    <!-- At the top are our bean class identifiers --->
    <bean id="{action.api.id}" class="{class.path}" scope="prototype"/>
    <bean id="{action.api.id.2}" class="{class.path}" scope="prototype"/>

     <!-- Below the class identifiers come the declarations for out actions/userSelectionMethods -->

     <!-- Use class workflowActionConfig for an action -->
     <bean id="{action.id}" class="oorg.dspace.xmlworkflow.state.actions.WorkflowActionConfig"
scope="prototype">
         <constructor-arg type="java.lang.String" value="{action.id}"/>

         <property name="processingAction" ref="{action.api.id}"/>
         <property name="requiresUI" value="{true/false}"/>
     </bean>

     <!-- Use class UserSelectionActionConfig for a user selection method -->
     <!--User selection actions-->
     <bean id="{action.api.id.2}" class="org.dspace.xmlworkflow.state.actions.UserSelectionActionConfig"
 scope="prototype">
         <constructor-arg type="java.lang.String" value="{action.api.id.2}"/>

         <property name="processingAction" ref="{user.selection.bean.id}"/>
         <property name="requiresUI" value="{true/false}"/>
     </bean>
</beans>
```

Two types of actions are configured in this Spring configuration file:

- User selection action: This type of action is always the first action of a step and is responsible for the user selection process of that step. In case a step has no role attached, no user will be selected and the `NoUserSelectionAction` is used.
- Processing action: This type of action is used for the actual processing of a step. Processing actions contain the logic required to execute the required operations in each step. Multiple processing actions can be defined in one step. These user and the workflow item will go through these actions in the order they are specified in the workflow configuration unless an alternative outcome is returned by one of them.

User Selection Action

Each user selection action that is used in the workflow configuration refers to a bean definition in the `workflow-actions.xml` file. In order to define a new user selection action, the following XML code is used:

```
<bean id="{action.api.id.2}" class="org.dspace.xmlworkflow.state.actions.UserSelectionActionConfig"
scope="prototype">
    <constructor-arg type="java.lang.String" value="{action.api.id.2}"/>

    <property name="processingAction" ref="{user.selection.bean.id}"/>
    <property name="requiresUI" value="{true/false}"/>
</bean>
```

This bean defines a new UserSelectionActionConfig and the following child tags:

- constructor-arg: This is a constructor argument containing the ID of the task. This is the same as the `id` attribute of the bean and is used by the workflow configuration to refer to this action.
- property `processingAction`: This tag refers the the ID of the API bean, responsible for the implementation of the API side of this action. This bean should also be configured in this XML.
- property `requiresUI`: In case this property is true, the workflow framework will expect a user interface for the action. Otherwise the framework will automatically execute the action and proceed to the next one.

Processing Action
Processing actions are configured similarly to the user selection actions. The only difference is that these processing action beans are implementations of the WorkflowActionConfig class instead of the UserSelectionActionConfig class.

4.5.9.4 Authorizations

Currently, the authorizations are always granted and revoked based on the tasks that are available for certain users and groups. The types of authorization policies that is granted for each of these is always the same:

- READ
- WRITE
- ADD
- DELETE

4.5.9.5 Database

The workflow uses a separate metadata schema named `workflow`. The fields this schema contains can be found in the `[dspace]/config/registries` directory and in the file `workflow-types.xml`. This schema is only used when using the score reviewing system at the moment, but one could always use this schema if metadata is required for custom workflow steps.

The following tables have been added to the DSpace database. All tables are prefixed with 'cwf_' to avoid any confusion with the existing workflow related database tables:

cwf_workflowitem

The cwf_workflowitem table contains the different workflowitems in the workflow. This table has the following columns:

- workflowitem_id: The identifier of the workflowitem and primary key of this table
- item_id: The identifier of the DSpace item to which this workflowitem refers.
- collection_id: The collection to which this workflowitem is submitted.
- multiple_titles: Specifies whether the submission has multiple titles (important for submission steps)
- published_before: Specifies whether the submission has been published before (important for submission steps)

- multiple_files: Specifies whether the submission has multiple files attached (important for submission steps)

cwf_collectionrole

The cwf_collectionrole table represents a workflow role for one collection. This type of role is the same as the roles that existed in the original workflow meaning that for each collection a separate group is defined to described the role. The cwf_collectionrole table has the following columns:

- collectionrol_id: The identifier of the collectionrole and the primaty key of this table
- role_id: The identifier/name used by the workflow configuration to refer to the collectionrole
- collection_id: The collection identifier for which this collectionrole has been defined
- group_id: The group identifier of the group that defines the collection role

cwf_workflowitemrole

The cwf_workflowitemrole table represents roles that are defined at the level of an item. These roles are temporary roles and only exist during the execution of the workflow for that specific item. Once the item is archived, the workflowitemrole is deleted. Multiple rows can exist for one workflowitem with e.g. one row containing a group and a few containing epersons. All these rows together make up the workflowitemrole The cwf_workflowitemrole table has the following columns:

- workflowitemrole_id: The identifier of the workflowitemrole and the primaty key of this table
- role_id: The identifier/name used by the workflow configuration to refer to the workflowitemrole
- workflowitem_id: The cwf_workflowitem identifier for which this workflowitemrole has been defined
- group_id: The group identifier of the group that defines the workflowitemrole role
- eperson_id: The eperson identifier of the eperson that defines the workflowitemrole role

cwf_pooltask

The cwf_pooltask table represents the different task pools that exist for a workflowitem. These task pools can be available at the beginning of a step and contain all the users that are allowed to claim a task in this step. Multiple rows can exist for one task pool containing multiple groups and epersons. The cwf_pooltask table has the following columns:

- pooltask_id: The identifier of the pooltask and the primaty key of this table
- workflowitem_id: The identifier of the workflowitem for which this task pool exists
- workflow_id: The identifier of the workflow configuration used for this workflowitem
- step_id: The identifier of the step for which this task pool was created
- action_id: The identifier of the action that needs to be displayed/executed when the user selects the task from the task pool
- eperson_id: The identifier of an eperson that is part of the task pool
- group_id: The identifier of a group that is part of the task pool

cwf_claimtask

The cwf_claimtask table represents a task that has been claimed by a user. Claimed tasks can be assigned to users or can be the result of a claim from the task pool. Because a step can contain multiple actions, the claimed task defines the action at which the user has arrived in a particular step. This makes it possible to stop working halfway the step and continue later. The cwf_claimtask table contains the following columns:

- claimtask_id: The identifier of the claimtask and the primary key of this table
- workflowitem_id: The identifier of the workflowitem for which this task exists
- workflow_id: The id of the workflow configuration that was used for this workflowitem
- step_id: The step that is currenlty processing the workflowitem

- action_id: The action that should be executed by the owner of this claimtask
- owner_id: References the eperson that is responsible for the execution of this task

cwf_in_progress_user

The cwf_in_progess_user table keeps track of the different users that are performing a certain step. This table is used because some steps might require multiple users to perform the step before the workflowitem can proceed. The cwf_in_progress_user table contains the following columns:

- in_progress_user_id: The identifier of the in progress user and the primary key of this table
- workflowitem_id: The identifier of the workflowitem for which the user is performing or has performed the step.
- user_id: The identifier of the eperson that is performing or has performe the task
- finished: Keeps track of the fact that the user has finished the step or is still in progress of the execution

4.5.9.6 Additional workflow steps/actions and features

Optional workflow steps: Select single reviewer workflow

This workflow makes it possible to assign a single user to review an item. This workflow configuration skips the task pool option meaning that the assigned reviewer no longer needs to claim the task. The configuration consists of the following 2 steps.

- AssignStep: During the assignstep, a user has the ability to select a responsible user to review the workflowitem. This means that for each workflowitem, a different user can be selected. Because a user is assigned, the task pool is no longer required.
- ReviewStep: The start of the reviewstep is different than the typical task pool. Instead of having a task pool, the user will be automatically assigned to the task. However, the user still has the option to reject the task (in case he or she is not responsible for the assigned task) or review the item. In case the user rejects the task, the workflowitem will be sent to the another step in the workflow as an alternative to the default outcome.

Optional workflow steps: Score review workflow

The score review system allows reviewers to give the reviewed item a rating. Depending on the results of the rating, the item will be approved to go to the next workflow step or will be sent to an alternative step. The scrore review workflow consists of the following 2 steps.

- ScoreReviewStep: The group of responsible users for the score reviewing will be able to claim the task from the taskpool. Dependingn on the configuration, a different number of users can be required to execute the task. This means that the task will be available in the task pool until the required number of users has at least claimed the task. Once everyone of them has finished the task, the next (automatic) processing step is activated.
- EvaluationStep: During the evaluationstep, no user interface is required. The workflow system will automatically execute the step that evaluates the different scores. In case the average score is more than a configurable percentage, the item is approved, otherwise it is rejected. (The minimum average score is set by adjusting the `minimumAcceptanceScore` property passed to `evaluationactionAPI` in `config/spring/api/workflow-actions.xml`.)

Workflow overview features

The DSpace UI also provides a feature to allow Administrators to see & administer all active workflows (workitems). This feature is provided in the "Administer Workflow" menu option. Currently, the Administrator has the ability to permanently delete the workflowitem, or to send it back to the original submitter.

4.5.10 Submission User Interface

This page explains various customization and configuration options that are available within DSpace for the Item Submission user interface.

DSpace Submission Configuration changed in v7.0

The name and structure of the Submission configuration files changed in 7.0. The DSpace 6.x (and below) "item-submission.xml" and "input-forms.xml" configuration files are no longer supported. In 7.x and above, the format of the "item-submission.xml" file has been updated, and the older "input-forms.xml" has been replaced by a new "submission-forms.xml".

You can choose to either start fresh with the new v7 configuration files (see documentation below) and/or use the "./dspace submission-forms-migrate" script to migrate your old configurations into new ones. See the Upgrading DSpace(see page 75) guide (step on "Update your DSpace Configurations") for more information on using the migration script.

4.5.10.1 Default Submission Process

The DSpace Submission process consists of a series of "steps", where each "step" corresponds to one or "sections" in the Submission UI. By default, the DSpace Submission process includes the following steps/sections, in this order:

1. "Select Collection" (id="collection"), appears as dropdown: If not already selected, the user must select a collection to deposit the Item into. As of DSpace 7, you also can change the Collection you are submitting into at any time. However, be aware that there may be some metadata lost if the Collection you switch two uses a *different* submission form & you already began entering metadata in the current submission.
2. "Describe" sections (id="traditionalpageone" and "traditionalpagetwo"): This is where the user may enter descriptive metadata about the Item. This step may consist of one or more sections of metadata entry. By default, there are two sections of metadata-entry . For information on modifying the metadata entry pages, please see Custom Metadata-entry Pages for Submission(see page 0) section below.
3. "Upload" section (id="upload"): This is where the user may upload one or more files to associate with the Item. As of DSpace 7, you can also drag and drop files anywhere on the page to trigger an upload. For more information on file upload, also see Configuring the File Upload step(see page 271) below.
4. "License" section (id="license"): This is where the user **must** agree to the repository distribution license in order to complete the deposit. This repository distribution license is defined in the `[dspace]/config/default.license` file. It can also be customized per-collection from the Collection Edit UI.
5. "Deposit" button: Once all required fields/sections are completed, the "Deposit" button becomes enabled. After clicking it, the new Item will either become immediately available or undergo a workflow approval process (depending on the Collection policies). For more information on the workflow approval process see Configurable Workflow(see page 250)

To modify or reorganize these submission steps, just modify the `[dspace]/config/item-submission.xml` file. Please see the section below on Reordering/Removing/Adding Submission Steps(see page 264).

You can also choose to have different submission processes for different DSpace Collections. For more details, please see the section below on Assigning a custom Submission Process to a Collection(see page 265).

Optional Steps

DSpace also ships with several optional steps which you may choose to enable if you wish. In no particular order:

- "Access" step **(NOT YET AVAILABLE in 7.0, but coming soon)**: This step allows the user to (optionally) modify access rights or set an embargo during the deposit of an Item. For more information on this step, and Embargo options in general, please see the Embargo(see page 113) documentation.
- "CC License" step (id="cclicense"): This step allows the user to (optionally) assign a Creative Commons license to a particular Item. Please see the Configuring Creative Commons License(see page 601) section of the Configuration documentation for more details.

- "Extraction" step (id="extractionstep"): This step will automatically attempt to extract metadata from uploaded files and populate it in the submission form. By default it is disabled, as it populates metadata automatically (without notifying the user).
- Various Configurable Entities(see page 134) related steps: These steps are "Describe" steps that are specific to different Entity types. They provide a list of metadata fields of specific interest to those Entities.

To enable any of these optional submission steps, just uncomment the step definition within the `[dspace]/config/item-submission.xml` file. Please see the section below on Reordering/Removing/Adding Submission Steps(see page 264).

You can also choose to enable certain steps only for specific DSpace Collections. For more details, please see the section below on Assigning a custom Submission Process to a Collection(see page 265).

4.5.10.2 Understanding the Submission Configuration Files

The `[dspace]/config/item-submission.xml` contains the submission configurations for the DSpace UI. This configuration file contains detailed documentation within the file itself, which should help you better understand how to best utilize it.

The Structure of *item-submission.xml*

> **The structure of this file changed slightly in DSpace 7**
>
> As of DSpace 7, the following structural changes were made to item-submission.xml:
>
> - Step definitions under <step-definitions> now use the <step-definition> tag (previously, in 6.x, the tag was named <step>)
> - Every step definition now needs to be defined under <step-definitions> (previously, in 6.x, you could also define them in <submission-process>), and have a unique ID
> - Each <step-definition> now only represents a single "section" of the Submission UI. (previously, in 6.x, some steps like Describe represented multiple pages)
> - An attribute "mandatory=[true|false]" was added to the <step> element. When true, that section is always displayed to the user. When false, it's not displayed by default, but instead must be activated explicitly by the user by choosing to add the section in the Submission UI.
> - The old <workflow-editable> element has been replaced with a <scope> element which defines when/how this <step> should be displayed.

Because this file is in XML format, you should be familiar with XML before editing this file. By default, this file contains the "traditional" Item Submission Process for DSpace, which consists of the following Steps (in this order):

Select Collection -> Describe (two steps) -> Upload -> License -> Complete

If you would like to customize the steps used or the ordering of the steps, you can do so within the *<submission-definition>* section of the *item-submission.xml* .

In addition, you may also specify different Submission Processes for different DSpace Collections. This can be done in the *<submission-map>* section. The *item-submission.xml* file itself documents the syntax required to perform these configuration changes.

Defining Steps (*<step>*) within the *item-submission.xml*

This section describes how Steps of the Submission Process are defined within the *item-submission.xml*.

Where to place your <*step-definition*>

The <*step-definition*> always appear within the <*step-definitions*> section of the *item-submission.xml* configuration file.

- This section allows all <*step*> definitions to be defined globally (i.e. so they may be used in multiple <*submission-process*> definitions). Steps defined in this section **must** define a unique *id* which can be used to reference this step.
- For example:

```
<step-definitions>
    <step-definition id="custom-step">
       ...
    </step>
    ...
</step-definitions>
```

- The above step definition could then be referenced from within a <*submission-process*> as simply <*step id="custom-step"/*>

The ordering of <*step*> tags matter!

The ordering of the <*step*> tags within a <*submission-process*> definition directly corresponds to the order in which those steps will appear!

For example, the following defines a Submission Process where the *License* step directly precedes the *Describe* step (more information about the structure of the information under each <step> tag can be found in the section on Structure of the <step> Definition below):

```
<submission-process>
    <!--Step 1 will be to Sign off on the License-->
    <step id="license"/>

    <!--Step 2 & 3 will be to ask for metadata-->
    <step id="traditionalpageone"/>
    <step id="traditionalpagetwo"/>

    ...[other steps]...
</submission-process>
```

Structure of the <step-definition> tag

The structure of the <step-definition> tag is as follows:

```
<step-definition id="traditionalpageone" mandatory="true">
    <heading>submit.progressbar.describe.stepone</heading>
    <processing-class>org.dspace.app.rest.submit.step.DescribeStep</processing-class>
    <type>submission-form</type>
    <!-- <scope visibility="hidden" visibilityOutside="hidden">submission</scope> -->
</step-definition>
```

Each *step* contains the following elements/attributes. The required elements are so marked:

- **mandatory** *(attribute)*: [true|false] When true, the step's section is displayed by default to all users in the UI. When false, the step is not displayed and must be activated explicitly by the user by selecting it in the UI or supplying data of interest to the section.
- **heading**: Partial I18N key (defined in the UI's language packs) which corresponds to the text that should be displayed in section header for this step. This partial I18N key is prefixed with "submission.sections.". Therefore, the full i18n key is "submission.sections.[heading]" in the User Interface's language packs (e.g. en.json5 for English)
- **processing-class (Required):** Full Java path to the Processing Class for this Step. This Processing Class **must** perform the primary processing of any information gathered in this step. All valid step processing classes must extend the abstract `org.dspace.submit.AbstractProcessingStep` class (or alternatively, extend one of the pre-existing step processing classes in `org.dspace.submit.step.*`)
- **type (Required)**: The type of step defined. Most steps are of type "submission-form", which means they directly map to a `<form>` defined in the `submission-forms.xml` configuration file. In this situation, the `<step-definition>` "id" attribute MUST map to a `<form>` "name" attribute defined in submission-forms.xml. Any value is allowed, and only "submission-form" has a special meaning at this time.
- **scope:** Optionally, allows you to limit the "scope" of this particular step, and define whether the step is visible outside that scope. Valid scope values include "submission" (limited to the submission form) and "workflow" (limited to workflow approval process). You can also define a "visibilityOutside" attribute which can be set to "read-only" (in other scopes you can see this step but not edit it), or "hidden" (in other scopes you cannot see this step).

4.5.10.3 Reordering/Removing/Adding Submission Steps

The removal of existing steps and reordering of existing steps is a relatively easy process!

Reordering steps

1. Locate the `<submission-process>` tag which defines the Submission Process that you are using. If you are unsure which Submission Process you are using, it's likely the one with `name="traditional"`, since this is the traditional DSpace submission process.
2. Reorder the `<step>` tags within that `<submission-process>` tag. Be sure to move the *entire* `<step>` tag.

Removing one or more steps

1. Locate the `<submission-process>` tag which defines the Submission Process that you are using. If you are unsure which Submission Process you are using, it's likely the one with `name="traditional"`, since this is the traditional DSpace submission process.
2. Comment out (i.e. surround with `<!--` and `-->`) the `<step>` tags which you want to remove from that *<submission-process>* tag. Be sure to comment out the *entire* `<step>` *tag.*
    - *Hint:* You cannot remove the *"collection"* step, as an DSpace Item cannot exist without belonging to a Collection.

Adding one or more optional steps

1. Locate the `<submission-process>` tag which defines the Submission Process that you are using. If you are unsure which Submission Process you are using, it's likely the one with `name="traditional"`, since this is the traditional DSpace submission process.
2. Uncomment (i.e. remove the `<!--` and `-->`) the `<step>` tag(s) which you want to add to that `<submission-process>` tag. Be sure to uncomment the *entire* `<step>` *tag.*

4.5.10.4 Assigning a custom Submission Process to a Collection

Assigning a custom submission process to a Collection in DSpace involves working with the *submission-map* section of the *item-submission.xml*. For a review of the structure of the *item-submission.xml* see the section above on Understanding the Submission Configuration File.

Each *name-map* element within *submission-map* associates a collection with the name of a submission definition. Its *collection-handle* attribute is the Handle of the collection. Its *submission-name* attribute is the submission definition name, which must match the *name* attribute of a *submission-process* element (in the *submission-definitions* section of *item-submission.xml*.

For example, the following fragment shows how the collection with handle "12345.6789/42" is assigned the "custom" submission process:

```
<submission-map>
    <name-map collection-handle="12345.6789/42" submission-name="custom" />
    ...
</submission-map>

<submission-definitions>
    <submission-process name="custom">
    ...
</submission-definitions>
```

It's a good idea to keep the definition of the *default* name-map, so there is always a default for collections which do not have a custom form set.

Getting A Collection's Handle

You will need the *handle* of a collection in order to assign it a custom form set. To discover the handle, go to the Community or Collection in the DSpace UI. Look for the "Permanent URI" listed near the top of the page. It should look something like:

```
http://myhost.my.edu/handle/12345.6789/42
```

The handle is everything after "handle/" (in the above example it is "12345.6789/42"). It should look familiar to any DSpace administrator. That is what goes in the *collection-handle* attribute of your *name-map* element.

4.5.10.5 Custom Metadata-entry Steps for Submission

Introduction

This section explains how to customize the Web forms used by submitters and editors to enter and modify the metadata for a new item. These metadata web forms are controlled by the *Describe* step within the Submission Process. However, they are also configurable via their own XML configuration file `[dspace]/config/submission-forms.xml`.

In this configuration you can create alternate metadata forms, which can then be mapped to a "submission-form" step in the "item-submission.xml" (see above).

In creating custom metadata forms, you can choose:

- Which fields appear on each form, and their sequence. (Keep in mind, each "form" represents to a "step" or section)
- Labels, prompts, and other text associated with each field.
- Ability to display smaller fields side-by-side in a single "row"
- List of available choices for each menu-driven field.

All of the custom metadata-entry forms for a DSpace instance are controlled by a single XML file, `submission-forms.xml`, in the *config* subdirectory under the DSpace home, `[dspace]/config/submission-forms.xml`. DSpace comes with a number of sample forms which implement the traditional metadata-entry forms, and also serves as a well-documented example. Some default forms include:

- "bitstream-metadata" - This is a special form which defines the metadata fields available for every uploaded bitstream (file)
- "traditionalpageone" - A sample form which is used by the first "Describe" step defined in item-submission.xml
- "traditionalpagetwo" - A sample form which is used by the second "Describe" step defined in item-submission.xml
- A number of sample forms for various out-of-the-box Configurable Entities(see page 134). These forms all have a corresponding <step> defined in item-submission.xml. In conjunction to those <step> definitions, these forms may be used to submit new Entities of specific types. Usually this is done by mapping that Entity-specific submission-process (in item-submission.xml) to a Collection which is used for new submissions of that Entity.

The rest of this section explains how to create your own sets of custom forms.

Describing Custom Metadata Forms

The description of a set of fields through which submitters enter their metadata is called a *form* (in the UI, each "form" is displayed in a separate collapsible section). A form is identified by a unique symbolic *name*. In the XML structure, the *form* is broken down into *rows of fields*. This allows you to place smaller fields side-by-side in a single, horizontal row, or alternatively decide to display one field per row.

The Structure of *submission-forms.xml*

> **The name & structure of this file changed slightly in DSpace 7**
>
> As of DSpace 7, the following structural changes were made to this configuration:
>
> - input-forms.xml (v6) was renamed to submission-forms.xml
> - <form-map> top-level element was removed. All Collection mappings are now in item-submission.xml
> - <page> element under <form> was removed. As described below, <form> element now represent a single section of the submission process.
> - <row> element under <form> was added. As described below, multiple fields can now be displayed in one horizontal row.
> - A new form named "bitstream-metadata" was introduced to allow you to configure which metadata is requested for a bitstream during submission.

The XML configuration file has a single top-level element, *input-forms*, which contains two elements in a specific order. The outline is as follows:

```
<input-forms>

    <--  Form Set Definitions -->
    <form-definitions>
        <form name="traditionalpageone">
            ...
        </form>
        ...
    </form-definitions>

    <--  Name/Value Pairs used within Multiple Choice Widgets -->
    <form-value-pairs>
        <value-pairs value-pairs-name="common_iso_languages" dc-term="language_iso">
            ...
        </value-pairs>
        ...
    </form-value-pairs>
</input-forms>
```

Using a form in a submission process for a Collection

Keep in mind, the "submission-forms.xml" only defines *forms* and *value-pairs* (used for specific fields like selectboxes). To enable a form requires also updating the "item-submission.xml" configuration to use that form (see also above):

1. In "item-submission.xml", a <step-definition> of type "submission-form" must be created, with an "id" matching the *name* of the *form* (see above for more details on step-definition)
2. In "item-submission.xml", a <submission-process> must be created/updated to use that newly defined "step".
3. Finally, also in "item-submission.xml", a Collection must be setup to use that submission process in the <submission-map> section.

So, if you modify submission-forms.xml, you may need to double check your changes will be used in your item-submission.xml.

Adding a Form

You can add a new form by creating a new *form* element within the *form-definitions* element. It has one attribute, *name*, which as described above must match the "id" of a <step-definition> in "item-submission.xml".

Forms and Pages
The content of the *form* is a sequence of *row* elements. Each of these corresponds to a single, horizontal row, containing metadata input fields. The rows are presented in sequence, with the first row displayed at the top of the form. A form is displayed as a section (or step) within the submission process.

A *form* may contain any number of rows. A row generally only contains one or two input fields (including more than one input field may require the "style" setting, see below). Each field defines an interactive dialog where the submitter enters one of the Dublin Core metadata items.

Composition of a Field
Each *field* contains the following elements, in the order indicated. The required sub-elements are so marked:

- **dc-schema** (Required) : Name of metadata schema employed, e.g. *dc* for Dublin Core. This value must match the value of the *schema* element defined in *dublin-core-types.xml*
- **dc-element** (Required) : Name of the Dublin Core element entered in this field, e.g. *contributor*.

- **dc-qualifier**: Qualifier of the Dublin Core element entered in this field, e.g. when the field is *contributor.advisor* the value of this element would be *advisor*. Leaving this out means the input is for an unqualified DC element.
- **language**: If set to *true* a drop down menu will be shown, containing languages. The selected language will be used as language tag of the metadata field. A compulsory argument *value-pairs-name* must be given containing the name of the value pair that contains all the languages: e.g. `<language value-pairs-name="common_iso_languages">true</language>`.
- **repeatable**: Value is *true* when multiple values of this field are allowed, *false* otherwise. When you mark a field repeatable, the UI will add an "Add more" control to the field, allowing the user to ask for more fields to enter additional values. Intended to be used for arbitrarily-repeating fields such as subject keywords, when it is impossible to know in advance how many input boxes to provide. Repeatable fields also support reordering of values.
- **label** (Required): Text to display as the label of this field, describing what to enter, e.g. "*Your Advisor's Name*".
- **input-type** (Required): Defines the kind of interactive widget to put in the form to collect the Dublin Core value. Content must be one of the following keywords:
    - **onebox** – A single text-entry box (i.e. a normal input textbox)
    - **twobox** – A pair of simple text-entry boxes, used for *repeatable* values. *(By default, this input type is unused.)*
    - **textarea** – Large block of text that can be entered on multiple lines, e.g. for an abstract.
    - **name** – Personal name, with separate fields for family name and first name. When saved they are appended in the format 'LastName, FirstName'. *(By default, this input type is unused. Author fields now use the "onebox" type to support different types of names.)*
    - **date** – Calendar date. When required, demands that at least the year be entered.
    - **series** – Series/Report name and number. Separate fields are provided for series name and series number, but they are appended (with a semicolon between) when saved.
    - **dropdown** – Choose value(s) from a "drop-down" menu list. **Note:** You must also include a value for the *value-pairs-name* attribute to specify a list of menu entries from which to choose. Use this to make a choice from a restricted set of options, such as for the *language* item.
    - **qualdrop_value** – Enter a "qualified value", which includes *both* a qualifier from a drop-down menu and a free-text value. Used to enter items like alternate identifiers and codes for a submitted item, e.g. the DC *identifier* field. **Note:** As for the *dropdown* type, you must include the *value-pairs-name* attribute to specify a menu choice list.
    - **list** – Choose value(s) from a checkbox or radio button list. If the *repeatable* attribute is set to *true*, a list of checkboxes is displayed. If the *repeatable* attribute is set to *false*, a list of radio buttons is displayed. **Note:** You must also include a value for the *value-pairs-name* attribute to specify a list of values from which to choose. *(By default, this input type is unused.)*
    - **tag** - A free-text field which allows you to add multiple labels/tags as values. An example is the "Subject Keywords" field. **Note:** A tag field MUST be marked as repeatable.
- **hint** (Required): Content is the text that will appear as a "hint", or instructions, below the input field. Can be left empty, but the tag must be present.
- **required**: When this element is included with any content, it marks the field as a required input. If the user saves the form without entering a value for this field, that text is displayed as a warning message. For example, *<required>You must enter a title.</required>* Note that leaving the required element empty will *not* mark a field as required, e.g.:*<required></required>*
- **vocabulary:** When specified, this field uses a controlled vocabulary(see page 284) defined in `[dspace]/config/controlled-vocabularies/[name].xml`. This setting may be used to provide auto-complete functionality, for example in the "Subject Keywords" field (which uses the "tag" input type). See also the "Configuring Controlled Vocabularies" section below.
- **regex:** When specified, this field will be validated against the Regular Expression, and only successfully validating values will be saved. An example is commented out in the default "Author" field.
- **style:** When specified, this provides a CSS style recommendation to the UI for how to style that field. This is primarily used when displaying multiple fields per row, so that you can tell the UI how many columns each

field should use in that row. Keep in mind, these styles should follow the Bootstrap Grid System[250], where the number of columns adds up to 12. An example can be see in the default "Date of Issue" and "Publisher" fields, which are configured to use 4 (col-sm-4) and 8 (col-sm-8) columns respectively.

Item type Based Metadata Collection

> Not yet supported in DSpace 7.0. This feature will be coming in a future 7.x release. See DSpace Release 7.0 Status[251]

A field can be made visible depending on the value of *dc.type*. A new field element, <type-bind>, has been introduced to facilitate this. In this example the field will only be visible if a value of "thesis" or "ebook" has been entered into *dc.type* on an earlier page:

```
<field>
    <dc-schema>dc</dc-schema>
    <dc-element>identifier</dc-element>
    <dc-qualifier>isbn</dc-qualifier>
    <label>ISBN</label>
    <type-bind>thesis,ebook</type-bind>
</field>
```

Configuring Controlled Vocabularies

DSpace now supports controlled vocabularies to confine the set of keywords that users can use while describing items. The need for a limited set of keywords is important since it eliminates the ambiguity of a free description system, consequently simplifying the task of finding specific items of information. The controlled vocabulary allows the user to choose from a defined set of keywords organized in an tree (taxonomy) and then use these keywords to describe items while they are being submitted.

The taxonomies are described in XML following this (very simple) structure:

```
<node id="acmccs98" label="ACMCCS98">
    <isComposedBy>
        <node id="A." label="General Literature">
            <isComposedBy>
                <node id="A.0" label="GENERAL"/>
                <node id="A.1" label="INTRODUCTORY AND SURVEY"/>
                ...
            </isComposedBy>
        </node>
        ...
    </isComposedBy>
</node>
```

You are free to use any application you want to create your controlled vocabularies. A simple text editor should be enough for small projects. Bigger projects will require more complex tools. You may use Protegé to create your taxonomies, save them as OWL and then use a XML Stylesheet (XSLT) to transform your documents to the

250 https://getbootstrap.com/docs/4.0/layout/grid/
251 https://wiki.lyrasis.org/display/DSPACE/DSpace+Release+7.0+Status

appropriate format. Future enhancements to this add-on should make it compatible with standard schemas such as OWL or RDF.

New vocabularies should be placed in `[dspace]/config/controlled-vocabularies/` and must be according to the structure described.

Vocabularies need to be associated with the corresponding metadata fields. Edit the file `[dspace]/config/submission-forms.xml` and place a `"vocabulary"` tag under the `"field"` element that you want to control. Set value of the `"vocabulary"` element to the name of the file that contains the vocabulary, leaving out the extension (the add-on will only load files with extension "*.xml"). For example:

```
<field>
    <dc-schema>dc</dc-schema>
    <dc-element>subject</dc-element>
    <dc-qualifier></dc-qualifier>
    <repeatable>true</repeatable>
    <label>Subject Keywords</label>
    <input-type>onebox</input-type>
    <hint>Enter appropriate subject keywords or phrases below.</hint>
    <required></required>
    <vocabulary>srsc</vocabulary>
</field>
```

The vocabulary element has an optional boolean attribute closed that can be used to force input only with the Javascript of controlled-vocabulary add-on. The default behaviour (i.e. without this attribute) is as set closed="false". This allow the user also to enter the value in free way.

The following vocabularies are currently available by default:

- **nsi** - *nsi.xml* - The Norwegian Science Index
- **srsc** - *srsc.xml* - Swedish Research Subject Categories

Adding Value-Pairs

Finally, your custom form description needs to define the "value pairs" for any fields with input types that refer to them. Do this by adding a *value-pairs* element to the contents of *form-value-pairs*. It has the following required attributes:

- **value-pairs-name** – Name by which an *input-type* refers to this list.
- **dc-term** – Dublin Core field for which this choice list is selecting a value.

Each *value-pairs* element contains a sequence of *pair* sub-elements, each of which in turn contains two elements:

- **displayed-value** – Name shown (on the web page) for the menu entry.
- **stored-value** – Value stored in the DC element when this entry is chosen. Unlike the HTML *select* tag, there is no way to indicate one of the entries should be the default, so the first entry is always the default choice.

Example
Here is a menu of types of common identifiers:

```
<value-pairs value-pairs-name="common_identifiers" dc-term="identifier">
    <pair>
        <displayed-value>Gov't Doc #</displayed-value>
        <stored-value>govdoc</stored-value>
    </pair>
    <pair>
        <displayed-value>URI</displayed-value>
        <stored-value>uri</stored-value>
    </pair>
    <pair>
        <displayed-value>ISBN</displayed-value>
        <stored-value>isbn</stored-value>
    </pair>
</value-pairs>
```

It generates the following HTML, which results in the menu widget below. (Note that there is no way to indicate a default choice in the custom input XML, so it cannot generate the HTML *SELECTED* attribute to mark one of the options as a pre-selected default.)

```
<select name="identifier_qualifier_0">
    <option VALUE="govdoc">Gov't Doc #</option>
    <option VALUE="uri">URI</option>
    <option VALUE="isbn">ISBN</option>
</select>
```

Deploying Your Custom Forms

The DSpace web application only reads your custom form definitions when it starts up, so it is important to remember:

- *You must always restart Tomcat* (or whatever servlet container you are using) for changes made to the *submission-forms.xml* and/or *item-submission.xml* to take effect.

Any mistake in the syntax or semantics of the form definitions, such as poorly formed XML or a reference to a nonexistent field name, may result in errors in the DSpace REST API & UI. The exception message (at the top of the stack trace in the *dspace.log* file) usually has a concise and helpful explanation of what went wrong. Don't forget to stop and restart the servlet container before testing your fix to a bug.

4.5.10.6 Configuring the File Upload step

Basic Settings

The *Upload* step in the DSpace submission process has two configuration options which can be set with your *[dspace]/config/dspace.cfg* configuration file. They are as follows:

- *upload.max*- The maximum size of a file (in bytes) that can be uploaded from the UI. It defaults to 536870912 bytes (512MB). You may set this to -1 to disable any file size limitation.
    - *Note:* Increasing this value or setting to -1 does **not** guarantee that DSpace will be able to successfully upload larger files via the web, as large uploads depend on many other factors including bandwidth, web server settings, internet connection speed, etc.

- *webui.submit.upload.required* - Whether or not all users are *required* to upload a file when they submit an item to DSpace. It defaults to 'true'. When set to 'false' users will see an option to skip the upload step when they submit a new item.

Modifying metadata form presented for Bitstreams

After uploading a file (bitstream) in the Submission UI, you can optionally edit that bitstream's metadata. The form displayed on that edit screen is built by the "bitstream-metadata" form defined in submission-forms.xml. You can modify that form to change the fields captured for a Bitstream. *However, the "dc.title" field is REQUIRED in order to store the name of the file.*

```
<form-definitions>
    <!-- Form used for entering in Bitstream/File metadata after uploading a file -->
    <form name="bitstream-metadata">
        ...
    </form>
</form-definitions>
```

Modifying access conditions (embargo, etc.) presented for Bitstreams

After uploading a file (bitstream) in the Submission UI, you can optionally edit that bitstream's access conditions. This allows you to embargo a bitstream, lease it, or limit it to Administrators only.

These access conditions are defined in a new Spring Bean configuration file `[dspace]/config/spring/api/access-conditions.xml`

- The "uploadConfigurationService" bean maps an existing "UploadConfiguration" bean (default is "uploadConfigurationDefault") to a specific step/section name used in item-submission.xml.

```
<!-- This default configuration says the <step-definition id="upload"> defined in item-
submission.xml uses "uploadConfigurationDefault" -->
<bean id="uploadConfigurationService" class="org.dspace.submit.model.UploadConfigurationService">
    <property name="map">
        <map>
            <entry key="upload" value-ref="uploadConfigurationDefault" />
        </map>
    </property>
</bean>
```

- One or more UploadConfiguration beans may exist, providing different options for different upload sections. An "UploadConfiguration" consists of several properties:
    - **name** (Required): The unique name of this upload configuration
    - **configurationService** (Required): reference to the DSpace ConfigurationService (should always be "org.dspace.services.ConfigurationService").
    - **metadata** (Required): The metadata "form" to use for this upload configuration. The value specified here MUST correspond to a <form> defined in your submission-forms.xml. In the below example, the "bitstream-metadata" form is used by the "uploadConfigurationDefault" bean...meaning that form will be used to capture metadata about the uploaded bitstream.
    - **options** (Required): list of all "AccessConditionOption" beans to enable. This list will be shown to the user to let them select which access restrictions to place on each bitstream.
    - **maxSize:** Optionally, you can specify a maximum size of file accepted by this UploadConfiguration. If unspecified, default is to use "upload.max" in dspace.cfg, or have no maximum.

    - **required**: Optionally, you can specify if a file upload is required for this UploadConfiguration. If true, upload is required and users cannot complete a submission without uploading at least one file. If false, no upload is required to complete the submission. If unspecified, default is to use "webui.submit.upload.required" configuration in dspace.cfg, which defaults to "true" (file upload required).
    -
```
<bean id="uploadConfigurationDefault" class="org.dspace.submit.model.UploadConfiguration">
    <property name="name" value="upload"></property>
    <property name="configurationService" ref="org.dspace.services.ConfigurationService"/>
    <property name="metadata" value="bitstream-metadata" />
    <property name="options">
        <!-- This is the list of access options which will be displayed on the "bitstream-
metadata" form -->
        <list>
            <ref bean="openAccess"/>
            <ref bean="lease"/>
            <ref bean="embargoed" />
            <ref bean="administrator"/>
        </list>
    </property>
</bean>
```

- Any number of "AccessConditionOption" beans may be added for applying different types of access permissions to uploaded files (based on which one the user selects). These beans are easy to add/update, and just require the following
    - **ID** (Required): Each defined bean MUST have a unique "id" and have "class=org.dspace.submit.model.AccessConditionOption".
    - **groupName**: Optionally, define a specific DSpace Group which this Access Condition relates to. This group will be saved to the ResourcePolicy when this access condition is applied.
    - **name**: Give a unique name for this Access Condition. This name is stored in the ResourcePolicy "name" when this access condition is applied.
    - **hasStartDate**: If the access condition is time-based, you can decide whether a start date is required. (true = required start date, false = disabled/not required). This start date will be saved to the ResourcePolicy when this access condition is applied.
    - **startDateLimit**: If the access condition is time-based, you can optionally set an start date limit (e.g. +36MONTHS). This field is used to set an upper limit to the start date based on the current date. In other words, a value of "+36MONTHS" means that users cannot set a start date which is more than 3 years from today. This setting's value uses Solr's Date Math Syntax[252], and is always based on today (NOW).
    - **hasEndDate**: If the access condition is time-based, you can enable/disable whether an end date is required. (true = required end date, false = disabled/not required). This end date will be saved to the ResourcePolicy when this access condition is applied.
    - **endDateLimit**: If the access condition is time-based, you can optionally set an end date limit (e.g. +6MONTHS). This field is used to set an upper limit to the start date based on the current date. In other words, a value of "+6MONTHS" means that users cannot set an end date which is more than 6 months from today. This setting's value use Solr's Date Math Syntax[253], and is always based on today (NOW).

252 https://solr.apache.org/guide/7_5/working-with-dates.html#date-math
253 https://solr.apache.org/guide/7_5/working-with-dates.html#date-math

```
<!-- Example access option named "embargo", which lets users specify a future date
    (not more than 3 years from now) when this file will be available to Anonymous users -->
<bean id="embargoed" class="org.dspace.submit.model.AccessConditionOption">
        <property name="groupName" value="Anonymous"/>
        <property name="name" value="embargo"/>
        <property name="hasStartDate" value="true"/>
        <property name="startDateLimit" value="+36MONTHS"/>
        <property name="hasEndDate" value="false"/>
</bean>
```

- By default, DSpace comes with three out-of-the-box Access Conditions (which you can customize/change based on local requirements)
  - "administrator" - access restricts the bitstream to the Administrator group immediately (after submission completes)
  - "openAccess" - makes the bitstream immediately accessible to Anonymous group (after submission completes)
  - "embargo" - embargoes the bitstream for a period of time (maximum of 3 years, as defined in startDateLimit default setting), after which it becomes anonymously accessible.
  - "lease" - makes the bitstream anonymously accessible immediately (after submission completes), but that access *expires* after a period of time (maximum of 6 months, as defined in endDateLimit default setting). After that date it is no longer accessible (except to Administrators)

4.5.10.7 Creating new Submission Steps Programmatically.

First, a brief warning: *Creating a new Submission Step requires some Java knowledge, and is therefore recommended to be undertaken by a Java programmer whenever possible.*

In most scenarios, this is NOT necessary, as it's much easier to configure a custom Submission Step using DescribeStep or similar.

That being said, at a higher level, creating a new Submission Step requires the following (in this relative order):

1. Create a new Step Processing class
   - This class **must** extend the abstract `org.dspace.submit.AbstractProcessingStep` class and implement all methods defined by that abstract class.
   - This class should be built in such a way that it can process the input gathered from the UI
2. Add a valid Step Definition to the `item-submission.xml` configuration file.
   - This may also require that you add an I18N (Internationalization) key for this step's *heading to the UI*
   - For more information on *<step-definition> tags* within the `item-submission.xml`, see the section above on Defining Steps (*<step>*) within the `item-submission.xml`.
3. For the UI, you will need to..
   - Add a new section type to SectionsType enum matching to the type of step you are creating
   - Create a new Component for this new SectionsType, annotated with "@renderSectionFor()".... see existing section components under src/app/submission/sections for examples.
   - *(Other steps may be necessary... this process has not been fully documented at this time.)*

4.5.10.8 Live Import from external sources

General Framework

Introduction

This framework is used by both the REST API(see page 502) and User Interface(see page 367) to help enhance or enrich submissions. One examples usage is in Importing Items via basic bibliographic formats (Endnote, BibTex, RIS, TSV, CSV) and online services (OAI, arXiv, PubMed, CrossRef, CiNii)(see page 229)

Features

- lookup publications from remote sources
- Support for multiple implementations

Abstraction of input format

The importer framework does not enforce a specific input format. Each importer implementation defines which input format it expects from a remote source. The import framework uses generics to achieve this. Each importer implementation will have a type set of the record type it receives from the remote source's response. This type set will also be used by the framework to use the correct MetadataFieldMapping for a certain implementation. Read "Implementation of an import source" below for more information and how to enable the framework.

Editing Metadata Mapping

At a simple level, metadata mapping configurations are all in Spring configs in `[dspace.dir]/config/spring/api/`

In that directory, you'll find a mapping file per import source, e.g. "arxiv-integration.xml", "bibtex-integration.xml", "endnote-integration.xml", "pubmed-integration.xml", etc.

There are two different mapping types.

1. First, mapping from a file-based import (e.g. bibtex, endnote, ris, etc) to a DSpace metadata field.

a. The list of all of the enabled mappings can be found in a "MetadataFieldConfig" <util:map>, usually at the top of the config file.

```
<util:map id="bibtexMetadataFieldMap" key-
type="org.dspace.importer.external.metadatamapping.MetadataFieldConfig"
             value-
type="org.dspace.importer.external.metadatamapping.contributor.MetadataContributor">
    <description>Defines which metadatum is mapped on which metadatum. Note that while the
key must be unique it
        only matters here for postprocessing of the value. The mapped MetadatumContributor
has full control over
            what metadatafield is generated.
    </description>
    <!-- These entry tags are the enabled mappings. The "value-ref" must map to a <bean> ID.
-->
    <entry key-ref="dcTitle" value-ref="bibtexTitleContrib" />
    <entry key-ref="dcAuthors" value-ref="bibtexAuthorsContrib" />
    <entry key-ref="dcJournal" value-ref="bibtexJournalContrib" />
    <entry key-ref="dcIssued" value-ref="bibtexIssuedContrib" />
    <entry key-ref="dcJissn" value-ref="bibtexJissnContrib" />
</util:map>
```

b. Each field in the file is mapped to a DSpace metadata field in a "SimpleMetadataContributor" bean definition. *NOTE: a large number of DSpace defined metadata fields are already configured as MetadataFieldConfig beans in the "dublincore-metadata-mapper.xml" Spring Config in the same directory. These may be reused in other configurations.*

```
<!-- This example bean for BibTex says the "title" key in the BibTex" file should be mapped
to the DSpace metadata field
     defined in the "dcTitle" bean.  This "dcTitle" bean is found in "dublincore-metadata-
mapper.xml" and obviously maps to "dc.title" -->
<bean id="bibtexTitleContrib" class="org.dspace.importer.external.metadatamapping.contributor
.SimpleMetadataContributor">
    <property name="field" ref="dcTitle"/>
    <property name="key" value="title" />
</bean>
```

2. Second, mapping from an external API query import (e.g. arxiv, pubmed, etc) to a DSpace metadata field.
   a. Similar to above, The list of all of the enabled mappings can be found in a "MetadataFieldConfig" <util:map>, usually at the top of the config file.

```
<util:map id="arxivMetadataFieldMap" key-
type="org.dspace.importer.external.metadatamapping.MetadataFieldConfig"
            value-
type="org.dspace.importer.external.metadatamapping.contributor.MetadataContributor">
    <description>Defines which metadatum is mapped on which metadatum. Note that while the
key must be unique it
        only matters here for postprocessing of the value. The mapped MetadatumContributor
has full control over
        what metadatafield is generated.
    </description>
    <!-- These entry tags are the enabled mappings. The "value-ref" must map to a <bean> ID.
-->
    <entry key-ref="arxiv.title" value-ref="arxivTitleContrib"/>
    <entry key-ref="arxiv.summary" value-ref="arxivSummaryContrib"/>
    <entry key-ref="arxiv.published" value-ref="arxivPublishedContrib"/>
    <entry key-ref="arxiv.arxiv.doi" value-ref="arxivDoiContrib"/>
    <entry key-ref="arxiv.arxiv.journal_ref" value-ref="arxivJournalContrib"/>
    <entry key-ref="arxiv.category.term" value-ref="arxivCategoryTermContrib"/>
    <entry key-ref="arxiv.author.name" value-ref="arxivAuthorContrib"/>
    <entry key-ref="arxiv.identifier.other" value-ref="arxivOtherContrib"/>
</util:map>
```

b. Each field in the file is mapped to a DSpace metadata field, usually in a "SimpleXPathMetadatumContributor" bean definition which also uses a "MetadataFieldConfig" bean. *NOTE: a large number of DSpace defined metadata fields are already configured as MetadataFieldConfig beans in the "dublincore-metadata-mapper.xml" Spring Config in the same directory. These may be reused in other configurations.*

```
<!-- This first bean define an XPath query ("ns:title") to map to a field (ID="arxiv.title")
in DSpace -->
<bean id="arxivTitleContrib" class="org.dspace.importer.external.metadatamapping.contributor.
SimpleXpathMetadatumContributor">
    <property name="field" ref="arxiv.title"/>
    <property name="query" value="ns:title"/>
    <property name="prefixToNamespaceMapping" ref="arxivBasePrefixToNamespaceMapping"/>
</bean>
<!-- This second bean then defines which DSpace field to use when "arxiv.title" is
references. In other words, between these two beans,
     the "ns:title" XPath query value is saved to "dc.title". -->
<bean id="arxiv.title" class="org.dspace.importer.external.metadatamapping.MetadataFieldConfi
g">
    <constructor-arg value="dc.title"/>
</bean>
```

Transformation to DSpace Item

The framework produces an 'ImportRecord' that is completely decoupled from DSpace. It contains a set of metadata DTO's that contain the notion of schema,element and qualifier. The specific implementation is responsible for populating this set. It is then very simple to create a DSpace item from this list.

Implementation of an import source for External Sources

Each external source/API importer implementation must at least implements `org.dspace.importer.external.service.components.QuerySource`, which provides the query method used by the framework to retrieve data from the remote source (e.g. Pubmed, ArXiv, etc). Each external source importer must implements, according to the provider APIs, the declared methods.
An useful abstract for remote sources is `org.dspace.importer.external.service.components.AbstractRemoteMetadataSource`.This class contains functionality to handle request timeout and to retry requests. Using this abstract, the query method must implements java.util.concurrent.Callable.

Implementation of an import source for files
Each file importer implementation must at least implements `org.dspace.importer.external.service.components.FileSource`, which provides the basic methods used by the framework to parse and load data from the file (e.g. CSV, Endnote, etc).

Each importer must implements the method:

```
public List<ImportRecord> getRecords(InputStream inputStream) throws FileSourceException;
```

This method is responsible to transform the input data into an ImportRecord list, which will then managed by the top layer of the framework.

The conversion from raw data to an ImportRecord could be done using the framework too, using the metadata mapping structure (see below).

File sources needs to know which file extensions they have to supports. This is done by the default method `isValidSourceForFile` in `FileSource`, and is controlled by the entries in the list returned by declared method `public List<String> getSupportedExtensions();`

An useful abstract for file source is `org.dspace.importer.external.service.components.AbstractPlainMetadataSource`. It should be used whenever it is possible to model the data in the file as a list of key-value lists (e.g. for CSV files, any row is a key value list).

Mapping raw data to Metadata

The framework core is a mid-layer component which allow the conversion of raw data into metadata (ImportRecord) using xml configurable spring beans.

The core of this approach is `org.dspace.importer.externa.service.AbstractImportMetadataSourceService`. Any service that wants to generate metadata from raw data should go through this abstract.

Our service then should extends AbstractImportMetadataSourceService, and use `transformSourceRecords` to transform raw data into ImportRecords.

The most relevant concept in the framework is `private MetadataFieldMapping<RecordType, MetadataContributor<RecordType>> metadataFieldMapping`. This is the place where the framework take the mapping between row data and the associated metadatum. This map must be injected in the service, and will be used by `transformSourceRecords` to convert the data.

`RecordType` is a generic type, which rapresent a single entry of the list of data, and will be mapped to a single ImportRecord. Any metadatum
will be mapped to a specific field in the RecordType using a Contributor as described in Metadata mapping.

Inherited methods

Method getImportSource() should return a unique identifier. Importer implementations should not be called directly, but class *org.dspace.importer.external.service.ImportService* should be called instead. This class contains the same methods as the importer implementations, but with an extra parameter 'url'. This url parameter should contain the same identifier that is returned by the getImportSource() method of the importer implementation you want to use.

The other inherited methods are used to query the remote source.

Spring configuration for External Sources
In order to make the live import providers available, them must be mapped as spring beans into `dspace-api/src/main/resources/spring/spring-dspace-addon-import-services.xml`.

This is an example of a provider which allow to import both files and remote source.

```
<bean id="PubmedImportService"
      class="org.dspace.importer.external.pubmed.service.PubmedImportMetadataSourceServiceImpl"
scope="singleton">
    <property name="metadataFieldMapping" ref="PubmedMetadataFieldMapping"/>
    <property name="supportedExtensions">
        <list>
            <value>xml</value>
        </list>
    </property>
    ...
</bean>
```

Here is defined the service responsible to fetch and transform the data `PubmedImportMetadataSourceServiceImpl`, which is an extension of `AbstractImportMetadataSourceService` as described above.

The field `metadataFieldMapping` is an instance of `Map<MetadataFieldConfig,MetadataContributor>` and contains the effective
mapping.

`supportedExtensions` is the file extension this provider supports.

To expose this provider as Live Import provider, we need to construct a bean of type `org.dspace.external.provider.impl.LiveImportDataProvider` in the following way

```
<bean id="pubmedLiveImportDataProvider" class="org.dspace.external.provider.impl.LiveImportDataProvider">
    <property name="metadataSource" ref="PubmedImportService"/>
    <property name="sourceIdentifier" value="pubmed"/>
    <property name="recordIdMetadata" value="dc.identifier.other"/>
</bean>
```

where `metadataSource` is the bean referencing to live import service as described in “Metadata mapping”, `sourceIdentifier` the name of
the provider in the live import framework and `recordIdMetadata` the metadatum used as id of the ImportRecord.

Metadata mapping
When using an implementation of `AbstractImportSourceService`, a mapping of remote record fields to DSpace metadata fields can be created.

first create an implementation of class `AbstractMetadataFieldMapping` with the same type set used for the importer implementation.

Then create a spring configuration file in `[dspace.dir]/config/spring/api`.

Each DSpace metadata field that will be used for the mapping must first be configured as a spring bean of class*org.dspace.importer.external.metadatamapping.MetadataFieldConfig*.

```
    <bean id="dc.title" class="org.dspace.importer.external.metadatamapping.M
etadataFieldConfig">
        <constructor-arg value="dc.title"/>
    </bean>
```

NOTE: A large number of these MetadataFieldConfig definitions are already provided out-of-the-box in `[dspace.dir]/config/spring/api/dublincore-metadata-mapper.xml` This allows most service-specific Spring configurations to just reuse those existing MetadataFieldConfig definitions

Now this metadata field can be used to create a mapping. To add a mapping for the "dc.title" field declared above, a new spring bean configuration of a class
class *org.dspace.importer.external.metadatamapping.contributor.MetadataContributor* needs to be added. This interface contains a type argument. The type needs to match the type used in the implementation of AbstractImportSourceService. The responsibility of each MetadataContributor implementation is to generate a set of metadata from the retrieved document. How it does that is completely opaque to the AbstractImportSourceService but it is assumed that only one entity (i.e. item) is fed to the metadatum contributor.

For example `java SimpleXpathMetadatumContributor implements MetadataContributor<OMElement>` can parse a fragment of xml and generate one or more metadata values.

This bean expects 2 property values:

- field: A reference to the configured spring bean of the DSpace metadata field. e.g. the "dc.title" bean declared above.
- query: The xpath expression used to select the record value returned by the remote source.

```
    <bean id="titleContrib" class="org.dspace.importer.external.metadatamappi
ng.contributor.SimpleXpathMetadatumContributor">
        <property name="field" ref="dc.title"/>
        <property name="query" value="dc:title"/>
    </bean>
```

Multiple record fields can also be combined into one value. To implement a combined mapping first create a *SimpleXpathMetadatumContributor* as explained above for each part of the field.

```
    <bean id="lastNameContrib" class="org.dspace.importer.external.metadatama
pping.contributor.SimpleXpathMetadatumContributor">
        <property name="field" ref="dc.contributor.author"/>
        <property name="query" value="x:authors/x:author/x:surname"/>
    </bean>
    <bean id="firstNameContrib" class="org.dspace.importer.external.metadatam
apping.contributor.SimpleXpathMetadatumContributor">
        <property name="field" ref="dc.contributor.author"/>
        <property name="query" value="x:authors/x:author/x:given-name"/>
    </bean>
```

Note that namespace prefixes used in the xpath queries are configured in bean "FullprefixMapping" in the same spring file.

```
    <util:map id="FullprefixMapping" key-type="java.lang.String" value-type="j
ava.lang.String">
        <description>Defines the namespace mappin for the
SimpleXpathMetadatum contributors</description>
        <entry key="http://purl.org/dc/elements/1.1/" value="dc"/>
        <entry key="http://www.w3.org/2005/Atom" value="x"/>
    </util:map>
```

Then create a new list in the spring configuration containing references to all *SimpleXpathMetadatumContributor* beans that need to be combined.

```
   <util:list id="combinedauthorList" value-type="org.dspace.importer.externa
l.metadatamapping.contributor.MetadataContributor" list-class="java.util.Link
edList">

        <ref bean="lastNameContrib"/>
        <ref bean="firstNameContrib"/>
    </util:list>
```

Finally create a spring bean configuration of class*org.dspace.importer.external.metadatamapping.contributor.CombinedMetadatumContributor*. This bean expects 3 values:

- field: A reference to the configured spring bean of the DSpace metadata field. e.g. the "dc.title" bean declared above.
- metadatumContributors: A reference to the list containing all the single record field mappings that need to be combined.
- separator: These characters will be added between each record field value when they are combined into one field.

```
    <bean id="authorContrib" class="org.dspace.importer.external.metadatamapp
ing.contributor.CombinedMetadatumContributor">
        <property name="separator" value=", "/>
        <property name="metadatumContributors" ref="combinedauthorList"/>
        <property name="field" ref="dc.contributor.author"/>
    </bean>
```

Each contributor must also be added to the "MetadataFieldMap" used by
the *MetadataFieldMapping* implementation. Each entry of this map maps a metadata field bean to a contributor.
For the contributors created above this results in the following configuration:

```
    <util:map id="org.dspace.importer.external.metadatamapping.MetadataFieldC
onfig"
              value-type="org.dspace.importer.external.metadatamapping.contri
butor.MetadataContributor">
        <entry key-ref="dc.title" value-ref="titleContrib"/>
        <entry key-ref="dc.contributor.author" value-ref="authorContrib"/>
    </util:map>
```

Note that the single field mappings used for the combined author mapping are not added to this list.

Available Metadata Contributor classes

Class	Description
SimpleXpathMetadatumContributor	Use an XPath expression to map the XPath result to a metadatum
SimpleMetadataContributor	This contributor is used in plain metadata as exposed above. Mapping is easy because it is based on the key used in the DTO.
CombinedMetadatumContributor	Use a LinkedList of MetadataContributor to combine into the value the resulting value for each contributor.

Framework Sources Implementations

PubMed Integration
Introduction
First read the base documentation on external importing (see above). This documentation explains the implementation of the importer framework using PubMed (http://www.ncbi.nlm.nih.gov/pubmed) as an example.

Publication Lookup URL
To be able to do the lookup for our configured import-service, we need to be able to know what URL to use to check for publications. This URL the `publication-lookup.url` setting defined within the `[dspace.dir]/config/modules/publication-lookup.cfg`. You may choose to modify this setting or override it within your local.cfg.

This setting can be modified in one of two ways:

- You can choose to specific a single, specific URL. This will tell the lookup service to only use one location to lookup publication information. Valid URLs are any that are defined as a `baseAddress` for beans within the `[src]/dspace-api/src/main/resources/spring/spring-dspace-addon-import-services.xml` Spring config file.
  - For example, this setting will ONLY use PubMed for lookups: `publication-lookup.url=http://eutils.ncbi.nlm.nih.gov/entrez/eutils/`
- By default, `publication-lookup.url` is set to an asterisk ('*'). This default value will attempt to lookup the publication using ALL configured importServices in the `[src]/dspace-api/src/main/resources/spring/spring-dspace-addon-import-services.xml` Spring config file

PubMed Metadata Mapping
The PubMed metadata mappings are defined in the `[dspace.dir]/config/spring/api/pubmed-integration.xml` Spring configuration file. These metadata mappings can be tweaked as desired. The format of this file is described in the "Metadata mapping" section above

PubMed specific classes Config
These classes are simply implementations based of the base classes defined in importer/external. They add characteristic behavior for services/mapping for the PubMed specific data.

Metadata mapping classes
- "PubmedFieldMapping". An implementation of AbstractMetadataFieldMapping, linking to the bean that serves as the entry point of other metadata mapping
- "PubmedDateMetadatumContributor"/"PubmedLanguageMetadatumContributor". Pubmed specific implementations of the "MetadataContributor" interface

Service classes
- "GeneratePubmedQueryService". Generates the pubmed query which is used to retrieve the records. This is based on a given item.
- "PubmedImportMetadataSourceServiceImpl". Child class of "AbstractImportMetadataSourceService", retrieving the records from pubmed.

ArXiv Integration
ArXiv Metadata Mapping
The ArXiv metadata mappings are defined in the `[dspace.dir]/config/spring/api/arxiv-integration.xml` Spring configuration file. These metadata mappings can be tweaked as desired. The format of this file is described in the "Metadata mapping" section above

4.5.10.9 Simple HTML Fragment Markup

A few features of the UI submission user interface, such as the deposit license text, can be marked up using a subset of HTML. Only these elements may be used:

- h1
- h2
- h3
- h4
- h5
- p
- a
- b
- i
- u

- ol
- li
- img

Do not try to make the content into a complete HTML document. It is just a fragment which will be textually inserted (*not* framed) into a larger document.

4.6 Items and Metadata

4.6.1 Authority Control of Metadata Values

4.6.1.1 WORK IN PROGRESS

4.6.1.2 Introduction

With DSpace you can describe digital objects such as text files, audio, video or data to facilitate easy retrieval and high quality search results. These descriptions are organized into metadata fields that each have a specific designation. For example: dc.title stores the title of an object, while dc.subject is reserved for subject keywords.

For many of these fields, including title and abstract, free text entry is the proper choice, as the values are likely to be unique. Other fields are likely to have values drawn from controlled sets. Such fields include unique names, subject keywords, document types and other classifications. For those kinds of fields the overall quality of the repository metadata increases if values with the same meaning are normalized across all items. Additional benefits can be gained if unique identifiers are associated as well in addition to canonical text values associated with a particular metadata field.

This page covers features included in the DSpace submission forms that allow repository managers to enforce the usage of normalized terms for those fields where this is required in their institutional use cases. DSpace offers

simple and straightforward features, such as definitions of simple text values for dropdowns, as well as more elaborate integrations with external vocabularies such as the Library of Congress Naming Authority.

4.6.1.3 Simple choice management for DSpace submission forms

The DSpace Submission forms, defined in the submission-forms.xml file, allows the inclusion of value pairs that can be organized in lists in order to populate dropdowns or other multiple choice elements. If you explore the default submission-forms.xml file, you can see that a number of such value pair lists are already pre defined.

Example

```
<value-pairs value-pairs-name="common_identifiers" dc-term="identifier">
    <pair>
        <displayed-value>Gov't Doc #</displayed-value>
        <stored-value>govdoc</stored-value>
    </pair>
    <pair>
        <displayed-value>URI</displayed-value>
        <stored-value>uri</stored-value>
    </pair>
    <pair>
        <displayed-value>ISBN</displayed-value>
        <stored-value>isbn</stored-value>
    </pair>
</value-pairs>
```

It generates the following HTML, which results in the menu widget below.

```
<select name="identifier_qualifier_0">
    <option VALUE="govdoc">Gov't Doc #</option>
    <option VALUE="uri">URI</option>
    <option VALUE="isbn">ISBN</option>
</select>
```

A list of value pairs has following required attributes:

- **value-pairs-name** – Name by which an *input-type* refers to this list.
- **dc-term** – Dublin Core field for which this choice list is selecting a value.

Each *value-pairs* element contains a sequence of *pair* sub-elements, each of which in turn contains two elements:

- **displayed-value** – Name shown (on the web page) for the menu entry.
- **stored-value** – Value stored in the DC element when this entry is chosen. Unlike the HTML *select* tag, there is no way to indicate one of the entries should be the default, so the first entry is always the default choice.

Use simple choice management to add language tags to metadata fields

DSpace uses the simple choice management to provide a controlled list of language tags. Out of the box DSpace comes with a list of ISO language tags. You can add further language lists or use the provided one to let submitters tag languages of metadata fields. Take a look at the part of this documentation about the configuration of the Submission User Interface(see page 260).

4.6.1.4 Hierarchical Taxonomies and Controlled Vocabularies

The value pairs system works well for short and flat lists of choices. DSpace offers a second way of structuring and managing more complex, hierarchical controlled vocabularies. In contrast to the value pairs system, these controlled vocabularies are managed in separate XML files in the `[dspace]/config/controlled-vocabularies/` directory instead of being entered straight into submission-forms.xml

The taxonomies are described in XML according to this structure:

```
<node id="acmccs98" label="ACMCCS98">
    <isComposedBy>
        <node id="A." label="General Literature">
            <isComposedBy>
                <node id="A.0" label="GENERAL"/>
                <node id="A.1" label="INTRODUCTORY AND SURVEY"/>
                ...
            </isComposedBy>
        </node>
        ...
    </isComposedBy>
</node>
```

As you can see, each node element has an id and label attribute. It can contain the isComposedBy element, which in its turn, consists of a list of other nodes.

You are free to use any application you want to create your controlled vocabularies. A simple text editor should be enough for small projects. Bigger projects will require more complex tools. You may use Protegé to create your taxonomies, save them as OWL and then use a XML Stylesheet (XSLT) to transform your documents to the appropriate format. Future enhancements to this add-on should make it compatible with standard schemas such as OWL or RDF.

How to invoke a controlled vocabulary from submission-forms.xml

Vocabularies need to be associated with the correspondent DC metadata fields. Edit the file `[dspace]/config/submission-forms.xml` and place a `"vocabulary"` tag under the `"field"` element that you want to control. Set value of the `"vocabulary"` element to the name of the file that contains the vocabulary, leaving out the extension (the add-on will only load files with extension "*.xml"). For example:

```
<field>
    <dc-schema>dc</dc-schema>
    <dc-element>subject</dc-element>
    <dc-qualifier></dc-qualifier>
    <repeatable>true</repeatable>
    <label>Subject Keywords</label>
    <input-type>onebox</input-type>
    <hint>Enter appropriate subject keywords or phrases below.</hint>
    <required></required>
    <vocabulary>srsc</vocabulary>
</field>
```

The vocabulary element has an optional boolean attribute `closed` that can be used to force input only with the Javascript of controlled-vocabulary add-on. The default behaviour (i.e. without this attribute) is `closed="false"`. This allows the user to enter values as free text in addition to selecting them from the controlled vocabulary.

The following vocabularies are currently available by default:

- **nsi** - *nsi.xml* - The Norwegian Science Index
- **srsc** - *srsc.xml* - Swedish Research Subject Categories

4.6.1.5 Authority Control: Enhancing DSpace metadata fields with Authority Keys

The aforementioned features only deal with text representations of controlled values. DSpace also offers support for adding authority keys and confidence values to a specific text value entered in a metadata field. The following terminology applies in the description of this area of DSpace functionality:

- **Authority** An *authority* is an external source of fixed values for a given domain, each unique value identified by a *key*. For example, the OCLC LC Name Authority Service[254], ORCID or VIAF.
- **Authority Record** The information associated with one of the values in an authority; may include alternate spellings and equivalent forms of the value, etc.
- **Authority Key** An opaque, hopefully persistent, identifier corresponding to exactly one record in the authority.

The fact that this functionality deals with **external** sources of authority makes it inherently different from the functionality for controlled vocabularies. Another difference is that the authority control is asserted *everywhere* metadata values are changed, including unattended/batch submission, SWORD package submission, and the administrative UI.

How it works

TODO

Original source:

Authority Control of Metadata Values[255] original development proposal for DSpace 1.6

4.6.2 Batch Metadata Editing

254 http://www.oclc.org/research/researchworks/authority/default.htm
255 https://wiki.lyrasis.org/display/DSPACE/Authority+Control+of+Metadata+Values

4.6.2.1 Batch Metadata Editing Tool

DSpace provides a batch metadata editing tool. The batch editing tool is able to produce a comma delimited file in the CSV format. The batch editing tool facilitates the user to perform the following:

- Batch editing of metadata (e.g. perform an external spell check)
- Batch additions of metadata (e.g. add an abstract to a set of items, add controlled vocabulary such as LCSH)
- Batch find and replace of metadata values (e.g. correct misspelled surname across several records)
- Mass move items between collections
- Mass deletion, withdrawal, or re-instatement of items
- Enable the batch addition of new items (without bitstreams) via a CSV file
- Re-order the values in a list (e.g. authors)

For information about configuration options for the Batch Metadata Editing tool, see Batch Metadata Editing Configuration(see page 294)

Export Function

Web Interface Export

Batch metadata exports (to CSV) can be performed from the Administrative menu:

- Login as an Administrative user
- In Sidebase, select "Export" → "Metadata". Type in the Community/Collection name.
    - Alternatively, browse to the Community or Collection you wish to export, and then go to "Export" → "Metadata". That Community/Collection will be preselected.
- Click "Export". A new Process will be created (in "Processes" menu). Once completed, download the resulting CSV.

DSpace 7.0 does not yet support all features

In DSpace 7.0, it is not possible to export to CSV based on search results. This feature existed in 6.x, and it is scheduled to be restored in a later 7.x release (currently 7.1). See DSpace Release 7.0 Status[256].

Please see below documentation for more information on the CSV format(see page 291) and actions that can be performed by editing the CSV(see page 292).

Command Line Export

The following table summarizes the basics.

Command used:	`[dspace]/bin/dspace metadata-export`

256 https://wiki.lyrasis.org/display/DSPACE/DSpace+Release+7.0+Status

Java class:	org.dspace.app.bulkedit.MetadataExport
Arguments short and (long) forms):	Description
`-f` or `--file`	Required. The filename of the resulting CSV.
`-i` or `--id`	The Item, Collection, or Community handle or Database ID to export. If not specified, **all** items will be exported.
`-a` or `--all`	Include all the metadata fields that are not normally changed (e.g. provenance) or those fields you configured in the `[dspace]/config/modules/bulkedit.cfg` to be ignored on export.
`-h` or `--help`	Display the help page.

To run the batch editing exporter, at the command line:

```
[dspace]/bin/dspace metadata-export -f name_of_file.csv -i 1023/24
```

Example:

```
[dspace]/bin/dspace metadata-export -f /batch_export/col_14.csv -i /1989.1/24
```

In the above example we have requested that a collection, assigned handle '*1989.1/24*' export the entire collection to the file '*col_14.csv*' found in the '*/batch_export*' directory.

Please see below documentation for more information on the CSV format(see page 291) and actions that can be performed by editing the CSV(see page 292) .

Import Function

Importing large CSV files

It is not recommended to import CSV files of more than 1,000 lines (i.e. 1,000 items). When importing files larger than this, it may be difficult for an Administrator to accurately verify the changes that the import tool states it will make. In addition, depending on the memory available to the DSpace site, large files may cause 'Out Of Memory' errors part way through the import process.

Web Interface Import

Batch metadata imports (from CSV) can be performed from the Administrative menu:

- First, complete all editing of the CSV(see page 292) and save your changes
- Login as an Administrative User
- In sidebar, select "Import" → "Metadata" and drag & drop the CSV file

DSpace 7.0 does not yet support all features

In DSpace 7.0, metadata import will occur immediately & results will be reported. In 6.x, after uploading the CSV, you were first presented with a summary of the changes that were to be performed, allowing you to review and choose whether to apply them or cancel. This "preview" feature will be restored in a later 7.x release (currently 7.1), see DSpace Release 7.0 Status[257].

Command Line Import

The following table summarizes the basics.

Command used:	`[dspace]/bin/dspace metadata-import`
Java class:	org.dspace.app.bulkedit.MetadataImport
Arguments short and (long) forms:	Description
`-f` or `--file`	Required. The filename of the CSV file to load.
`-s` or `--silent`	Silent mode. The import function does not prompt you to make sure you wish to make the changes.
`-e` or `--email`	The email address of the user. This is only required when adding new items.
`-w` or `--workflow`	When adding new items, the program will queue the items up to use the Collection Workflow processes.
`-n` or `--notify`	when adding new items using a workflow, send notification emails.
`-t` or `--template`	When adding new items, use the Collection template, if it exists.

257 https://wiki.lyrasis.org/display/DSPACE/DSpace+Release+7.0+Status

`-h` or `--help`	Display the brief help page.

Silent Mode should be used carefully. It is possible (and probable) that you can overlay the wrong data and cause irreparable damage to the database.

To run the batch importer, at the command line:

```
[dspace]/bin/dspace metadata-import -f name_of_file.csv
```

Example

```
[dspace]/bin/dspace metadata-import -f /dImport/col_14.csv
```

If you are wishing to upload new metadata **without** bitstreams, at the command line:

```
[dspace]/bin/dspace metadata-import -f /dImport/new_file.csv -e joe@user.com -w -n -t
```

In the above example we threw in all the arguments. This would add the metadata and engage the workflow, notification, and templates to all be applied to the items that are being added.

CSV Format

The CSV (comma separated values) files that this tool can import and export abide by the RFC4180[258] CSV format. This means that new lines, and embedded commas can be included by wrapping elements in double quotes. Double quotes can be included by using two double quotes. The code does all this for you, and any good csv editor such as Excel or OpenOffice will comply with this convention.

All CSV files are also in UTF-8 encoding in order to support all languages.

File Structure

The first row of the CSV must define the metadata values that the rest of the CSV represents. ***The first column must always be "id" which refers to the item's internal database ID***. *All other columns are optional.* The other columns contain the dublin core metadata fields that the data is to reside.

A typical heading row looks like:

```
id,collection,dc.title,dc.contributor,dc.date.issued,etc,etc,etc.
```

Subsequent rows in the csv file relate to items. A typical row might look like:

```
350,2292,Item title,"Smith, John",2008
```

If you want to store multiple values for a given metadata element, they can be separated with the double-pipe '||' (or another character that you defined in your `modules/bulkedit.cfg` file). For example:

258 http://www.ietf.org/rfc/rfc4180.txt

```
Horses||Dogs||Cats
```

Elements are stored in the database in the order that they appear in the CSV file. You can use this to order elements where order may matter, such as authors, or controlled vocabulary such as Library of Congress Subject Headings.

Editing the CSV

If you are editing with Microsoft Excel, be sure to open the CSV in Unicode/UTF-8 encoding

By default, Microsoft Excel may not correctly open the CSV in Unicode/UTF-8 encoding. This means that special characters may be improperly displayed and also can be "corrupted" during re-import of the CSV.

You need to tell Excel this CSV is Unicode, by importing it as follows. (*Please note these instructions are valid for MS Office 2007 and 2013. Other Office versions may vary*)

- First, open Excel (with an empty sheet/workbook open)
- Select "Data" tab
- Click "From Text" button (in the "External Data" section)
- Select your CSV file
- Wizard Step 1
    - Choose "Delimited" option
    - Start import at row: 1
    - In the "File origin" selectbox, select "65001 : Unicode (UTF-8)"
        - NOTE: these encoding options are sorted alphabetically, so "Unicode (UTF-8)" appears near the bottom of the list.
    - Click Next
- Wizard Step 2
    - Select "Comma" as the only delimiter
    - Click Next
- Wizard Step 3
    - Select "Text" as the "Column data format" (*Unfortunately, this must be done for each column individually in Excel*)
        - At a minimum, you MUST ensure all date columns (e.g. dc.date.issued) are treated as "Text" so that Excel doesn't autoconvert DSpace's YYYY-MM-DD format into MM/DD/YYYY
        - To avoid such autoconversion, it is safest to ensure each column is treated as "Text". Unfortunately, this means selecting each column one-by-one and choosing "Text" as the "Column data format".
    - Click Finish
- Choose whether to open CSV in the existing sheet or a new one

Tips to Simplify the Editing Process

When editing a CSV, here's a couple of basic tips to keep in mind:

1. The "id" column MUST remain intact. This column also must always have a value in it.
2. To simplify the CSV, you can simply remove any columns you do NOT wish to edit (except for "id" column, see #1). Don't worry, removing the entire column won't delete metadata (see #3)
3. When importing a CSV file, the importer will *overlay* the metadata onto what is already in the repository to determine the differences. It *only* acts on the contents of the CSV file, rather than on the complete item metadata. This means that the CSV file that is exported can be manipulated quite substantially before being re-imported. Rows (items) or Columns (metadata elements) can be removed and will be ignored.
    a. For example, if you only want to edit "dc.subject", you can remove ALL columns EXCEPT for "id" and "dc.subject" so that you can just manipulate the "dc.subject" field. On import, DSpace will see that you've only included the "dc.subject" field in your CSV and therefore will only update the "dc.subject" metadata field for any items listed in that CSV.
4. Because removing an entire column does NOT delete metadata value(s), if you actually wish to delete a metadata value you should leave the column intact, and simply clear out the appropriate row's value (in that column).

Editing Collection Membership

Items can be moved between collections by editing the collection handles in the 'collection' column. Multiple collections can be included. The first collection is the 'owning collection'. The owning collection is the primary collection that the item appears in. Subsequent collections (separated by the field separator) are treated as mapped collections. These are the same as using the map item functionality in the DSpace user interface. To move items between collections, or to edit which other collections they are mapped to, change the data in the collection column.

Adding Metadata-Only Items

New metadata-only items can be added to DSpace using the batch metadata importer. To do this, enter a plus sign '+' in the first 'id' column. The importer will then treat this as a new item. If you are using the command line importer, you will need to use the -e flag to specify the user email address or id of the user that is registered as submitting the items.

Deleting Metadata

It is possible to perform metadata deletes across the board of certain metadata fields from an exported file. For example, let's say you have used keywords (dc.subject) that need to be removed *en masse*. You would leave the column (dc.subject) intact, but remove the data in the corresponding rows.

Performing 'actions' on items

It is possible to perform certain 'actions' on items. This is achieved by adding an 'action' column to the CSV file (after the id, and collection columns). There are three possible actions:

1. ***'expunge'*** This permanently deletes an item. Use with care! This action must be enabled by setting 'allowexpunge = true' in `modules/bulkedit.cfg`
2. ***'withdraw'*** This withdraws an item from the archive, but does not delete it.
3. ***'reinstate'*** This reinstates an item that has previously been withdrawn.

If an action makes no change (for example, asking to withdraw an item that is already withdrawn) then, just like metadata that has not changed, this will be ignored.

Migrating Data or Exchanging data

It is possible that you have data in one Dublin Core (DC) element and you wish to really have it in another. An example would be that your staff have input Library of Congress Subject Headings in the Subject field (dc.subject) instead of the LCSH field (dc.subject.lcsh). Follow these steps and your data is migrated upon import:

1. Insert a new column. The first row should be the new metadata element. (We will refer to it as the TARGET)
2. Select the column/rows of the data you wish to change. (We will refer to it as the SOURCE)
3. Cut and paste this data into the new column (TARGET) you created in Step 1.
4. Leave the column (SOURCE) you just cut and pasted from empty. Do not delete it.

Common Issues

Metadata values in CSV export seem to have duplicate columns[259]
DSpace responds with "No changes were detected" when CSV is uploaded
Unfortunately, this response may be caused in many ways

- It's possible the CSV was not saved properly after editing. Check that the edits are in the CSV, and that there were no backend errors in the DSpace logs (which would be an indication of an invalid or corrupted CSV file)
- Depending on the version of DSpace, you may be encountering this known bug with processing linebreaks in CSV files: DS-3245[260]
- If you are setting a new embargo date in the CSV, ensure that the embargo lift date is a future date. It's been reported that past dates may cause DSpace to ignore item changes.

4.6.2.2 Batch Metadata Editing Configuration

The Batch Metadata Editing Tool(see page 287) allows the administrator to extract from the DSpace database a set of records for editing via a CSV file. It provides an easier way of editing large collections.

A full list of all available Batch Metadata Editing Configurations:

Configuration File:	**`[dspace]/config/modules/bulkedit.cfg`**
Property:	`bulkedit.valueseparator`
Example Value:	`bulkedit.valueseparator = ||`

259 https://wiki.duraspace.org/display/DSPACE/TechnicalFAQ#TechnicalFAQ-MetadatavaluesinCSVexportseemtohaveduplicatecolumns
260 https://jira.duraspace.org/browse/DS-3245

Configuration File:	**`[dspace]/config/modules/bulkedit.cfg`**
Informational note	The delimiter used to separate values within a single field. For example, this will place the double pipe between multiple authors appearing in one record (Smith, William || Johannsen, Susan). This applies to any metadata field that appears more than once in a record. The user can change this to another character.
Property:	`bulkedit.fieldseparator`
Example Value:	`bulkedit.fieldseparator = ,`
Informational note	The delimiter used to separate fields (defaults to a comma for CSV). Again, the user could change it something like '$'. If you wish to use a tab, semicolon, or hash (#) sign as the delimiter, set the value to be `tab`, `semicolon` or `hash`. `bulkedit.fieldseparator = tab`
Property:	`bulkedit.authorityseparator`
Example Value:	`bulkedit.authorityseparator = ::`
Informational note	The delimiter used to separate authority data (defaults to a double colon ::)
Property:	`bulkedit.gui-item-limit`
Example Value:	`bulkedit.gui-item-limit = 20`
Informational note	When using the WEBUI, this sets the limit of the number of items allowed to be edited in one processing. There is no limit when using the CLI.
Property:	`bulkedit.ignore-on-export`

Configuration File:	**[dspace]/config/modules/bulkedit.cfg**
Example Value:	`bulkedit.ignore-on-export = dc.date.accessioned, \` `dc.date.available, \` `dc.date.updated,` `dc.description.provenance`
Informational note	Metadata elements to exclude when exporting via the user interfaces, or when using the command line version and not using the -a (all) option.
Property:	`bulkedit.allowexpunge`
Example Value:	`bulkedit.allowexpunge = false`
Informational note	Should the 'action' column allow the 'expunge' method. By default this is set to false
Property	`bulkedit.allow-bulk-deletion`
Example Value:	`bulkedit.allow-bulk-deletion = dspace.agreements.end-user`
Informational note	Comma separated list of metadata fields that can be deleted in bulk using the 'metadata-deletion' script. By default only the 'dspace.agreements.end-user' field can be deleted in bulk, as doing so allows an Administrator to force all users to re-review the End User Agreement on their next login. However, you may choose to enable additional fields. Keep in mind, any fields listed here may be batch deleted from the Processes UI & such metadata deletions cannot be undone.

4.6.3 DOI Digital Object Identifier

- Adding support for other Registration Agencies(see page 306)

4.6.3.1 Persistent Identifier

It is good practice to use Persistent Identifiers to address items in a digital repository. There are many different systems for Persistent Identifiers: Handle[261], DOI[262], urn:nbn[263], purl[264] and many more. It is far out of the scope of this document to discuss the differences of all these systems. For several reasons the Handle System is deeply integrated in DSpace, and DSpace makes intensive use of it. With DSpace 3.0 the Identifier Service(see page 0) was introduced that makes it possible to also use external identifier services within DSpace.

DOIs are Persistent Identifiers like Handles are, but as many big publishing companies use DOIs they are quite well-known to scientists. Some journals ask for DOIs to link supplemental material whenever an article is submitted. Beginning with DSpace 4.0 it is possible to use DOIs in parallel to the Handle System within DSpace. By "using DOIs" we mean automatic generation, reservation and registration of DOIs for every item that enters the repository. These newly registered DOIs will not be used as a means to build URLs to DSpace items. Items will still rely on handle assignment for the item urls.

4.6.3.2 DOI Registration Agencies

To register a DOI one has to enter into a contract with a DOI registration agency which is a member of the International DOI Foundation. Several such agencies exist. Different DOI registration agencies have different policies. Some of them offer DOI registration especially or only for academic institutions, others only for publishing companies. Most of the registration agencies charge fees for registering DOIs, and all of them have different rules describing for what kind of item a DOI can be registered. To make it quite clear: to register DOIs with DSpace you have to enter into a contract with a DOI registration agency.

DataCite[265] is an international initiative to promote science and research, and a member of the International DOI Foundation. The members of DataCite act as registration agencies for DOIs. Some DataCite members provide their own APIs to reserve and register DOIs; others let their clients use the DataCite API directly. Starting with version 4.0 DSpace supports the administration of DOIs by using the DataCite API directly or by using the API from EZID (which is a service of the University of California Digital Library). This means you can administer DOIs with DSpace if your registration agency allows you to use the DataCite API directly or if your registration agency is EZID.

Configure DSpace to use the DataCite API

If you use a DOI registration agency that lets you use the DataCite API directly, you can follow the instructions below to configure DSpace. In case EZID is your registration agency the configuration of DSpace is documented here: Configure DSpace to use EZID service for registration of DOIs(see page 305).

To use DOIs within DSpace you have to configure several parts of DSpace:

- enter your DOI prefix and the credentials to use the API from DataCite in dspace.cfg,
- configure the script which generates some metadata,
- activate the DOI mechanism within DSpace,
- configure a cron job which transmits the information about new and changed DOIs to the registration agency.

261 http://www.handle.net/
262 http://www.doi.org/
263 http://tools.ietf.org/search/rfc3188
264 http://purl.oclc.org/docs/index.html
265 http://www.datacite.org

dspace.cfg

After you enter into a contract with a DOI registration agency, they'll provide you with user credentials and a DOI prefix. You have to enter these in the dspace cfg. Here is a list of DOI configuration options in dspace.cfg:

Configuration File:	**[dspace]/config/dspace.cfg**
Property:	`identifier.doi.user`
Example Value:	`identifier.doi.user = user123`
Informational Note:	Username to login into the API of the DOI registration agency. You'll get it from your DOI registration agency.
Property:	`identifier.doi.password`
Example Value:	`identifier.doi.password = top-secret`
Informational Note:	Password to login into the API of the DOI registration agency. You'll get it from your DOI registration agency.
Property:	`identifier.doi.prefix`
Example Value:	`identifier.doi.prefix = 10.5072`
Informational Note:	The prefix you got from the DOI registration agency. All your DOIs start with this prefix, followed by a slash and a suffix generated from DSpace. The prefix can be compared with a namespace within the DOI system.
Property:	`identifier.doi.namespaceseparator`
Example Value:	`identifier.doi.namespaceseparator = dspace-`
Informational Note:	This property is optional. If you want to use the same DOI prefix in several DSpace installations or with other tools that generate and register DOIs it is necessary to use a namespace separator. All the DOIs that DSpace generates will start with the DOI prefix, followed by a slash, the namespace separator and some number generated by DSpace. For example, if your prefix is 10.5072 and you want all DOIs generated by DSpace to look like 10.5072/dspace-1023 you have to set this as in the example value above.

Configuration File:	**[dspace]/config/dspace.cfg**
Property:	crosswalk.dissemination.DataCite.publisher
Example Value:	crosswalk.dissemination.DataCite.publisher = My University Press
Informational Note:	The name of the entity which published the item.
Property:	crosswalk.dissemination.DataCite.hostingInstitution
Example Value:	crosswalk.dissemination.DataCite.hostingInstitution = My University
Informational Note:	The name of the entity which hosts this instance of the object. If not configured, it will default to the value of crosswalk.dissemination.DataCite.publisher.
Property:	crosswalk.dissemination.DataCite.dataManager
Example Value:	crosswalk.dissemination.DataCite.dataManager = My University Department of Geology
Informational Note:	If not configured, it will default to the value of crosswalk.dissemination.DataCite.publisher.

ⓘ Please don't use the test prefix 10.5072 with DSpace. The test prefix 10.5072 differs from other prefixes: It answers GET requests for all DOIs even for DOIs that are unregistered. DSpace checks that it mint only unused DOIs and will create an Error: "Register DOI ... failed: DOI_ALREADY_EXISTS". Your registration agency can provide you an individual test prefix, that you can use for tests.

Metadata conversion

To reserve or register a DOI, DataCite requires that metadata be supplied which describe the object that the DOI addresses. The file [dspace]/config/crosswalks/DIM2DataCite.xsl controls the conversion of metadata from the DSpace internal format into the DataCite format. You have to add your DOI prefix, namespace separator and the name of your institution to this file:

\[dspace\]/config/crosswalks/DIM2DataCite.xsl

```
<!--
    Document   : DIM2DataCite.xsl
    Created on : January 23, 2013, 1:26 PM
    Author     : pbecker, ffuerste
    Description: Converts metadata from DSpace Intermediat Format (DIM) into
                 metadata following the DataCite Schema for the Publication and
                 Citation of Research Data, Version 2.2
-->
<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-2.2"
                version="1.0">

    <!-- CONFIGURATION -->
    <!-- Please add your DOI-Prefix and your namespace separator here (e.g. 10.5072-dspace-). -->
    <xsl:variable name="prefix">10.5072-dspace-</xsl:variable>
    <!-- The content of the following variable will be used as element publisher. -->
    <xsl:variable name="publisher">My University</xsl:variable>
    <!-- The content of the following variable will be used as element contributor with contributorType
datamanager. -->
    <xsl:variable name="datamanager"><xsl:value-of select="$publisher" /></xsl:variable>
    <!-- The content of the following variable will be used as element contributor with contributorType
hostingInstitution. -->
    <xsl:variable name="hostinginstitution"><xsl:value-of select="$publisher" /></xsl:variable>
    <!-- Please take a look into the DataCite schema documentation if you want to know how to use these
elements.
         http://schema.datacite.org -->


    <!-- DO NOT CHANGE ANYTHING BELOW THIS LINE EXCEPT YOU REALLY KNOW WHAT YOU ARE DOING! -->
...
```

Just change the value in the variable named "publisher".

If you want to know more about the DataCite Schema, have a look at the documentation[266]. If you change this file in a way that is not compatible with the DataCite schema, you won't be able to reserve and register DOIs anymore. Do not change anything if you're not sure what you're doing.

Identifier Service

The Identifier Service manages the generation, reservation and registration of identifiers within DSpace. You can configure it using the config file located in [dspace]/config/spring/api/identifier-service.xml. In the file you should already find the code to configure DSpace to register DOIs. Just read the comments and remove the comment signs around the two appropriate beans.

After removing the comment signs the file should look something like this (I removed the comments to make the listing shorter):

266 http://schema.datacite.org

\[dspace\]/config/spring/api/identifier-service.xml

```
<!--
    Copyright (c) 2002-2010, DuraSpace.  All rights reserved
    Licensed under the DuraSpace License.

    A copy of the DuraSpace License has been included in this
    distribution and is available at: http://www.dspace.org/license
-->

<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
           http://www.springframework.org/schema/beans/spring-beans-2.5.xsd">

    <bean id="org.dspace.identifier.IdentifierService"
          class="org.dspace.identifier.IdentifierServiceImpl"
          autowire="byType"
          scope="singleton"/>

    <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" />
    </bean>

    <bean id="org.dspace.identifier.doi.DOIConnector"
        class="org.dspace.identifier.doi.DataCiteConnector"
        scope="singleton">
        <property name='DATACITE_SCHEME' value='https'/>
        <property name='DATACITE_HOST' value='mds.test.datacite.org'/>
        <property name='DATACITE_DOI_PATH' value='/doi/' />
        <property name='DATACITE_METADATA_PATH' value='/metadata/' />
        <property name='disseminationCrosswalkName' value="DataCite" />
    </bean>
</beans>
```

If you use other IdentifierProviders beside the DOIIdentifierProvider there will be more beans in this file.

Please pay attention to configure the property DATACITE_HOST. Per default it is set to the DataCite test server. To reserve real DOIs you will have to change it to mds.datacite.org. Ask your registration agency if you're not sure about the correct address.

DSpace should send updates to DataCite whenever the metadata of an item changes. To do so you have to change the dspace.cfg again. You should remove the comments in front of the two following properties or add them to the dspace.cfg:

\[dspace\]/config/dspace.cfg

```
event.consumer.doi.class = org.dspace.identifier.doi.DOIConsumer
event.consumer.doi.filters = Item+Modify_Metadata
```

Then you should add 'doi' to the property `event.dispatcher.default.consumers`. After adding it, this property may look like this:

\[dspace\]/config/dspace.cfg

```
event.dispatcher.default.consumers = versioning, discovery, eperson, harvester, doi
```

DOIs using DataCite and Item Level Versioning

If you enabled Item Level Versioning(see page 307) you should enable the `VersionedDOIIdentifierProvider` instead of the `DOIIdentifierProvider`. The `VersionedDOIIdentifierProvider` ensures that newer versions of the same Item gets a DOI looking as the DOI of the first version of and item, extended by a dot and the version number. With DSpace 6 this also became the default for handles if Item Level Versioning is enabled. In the configuration file `[dspace]/config/spring/api/identifier-service.xml` you'll find the possiblity to enable the `VersionedDOIIdentifierProvider`. If you want to use versioned DOIS, please comment out the `DOIIdentifierProvider` as only one of both DOIProviders should be enabled at the same time.

Command Line Interface

To make DSpace resistant to outages of DataCite we decided to separate the DOI support into two parts. When a DOI should be generated, reserved or minted, DSpace does this in its own database. To perform registration and/or reservation against the DOI registration agency a job has to be started using the command line. Obviously this should be done by a cron job periodically. In this section we describe the command line interface, in case you ever want to use it manually. In the next section you'll see the cron job that transfers all DOIs designated for reservation and/or registration.

The command line interface in general is documented here: Command Line Operations(see page 460).

The command used for DOIs is '`doi-organiser`'. You can use the following options:

Option (short)	Option (long)	Parameter	Description
-d	--delete-all		Transmit information to the DOI registration agency about all DOIs that were deleted.
	--delete-doi	DOI	Transmit information to the DOI registration agency that the specified DOI was deleted. The DOI must already be marked for deletion; you cannot use this command to delete a DOI for an exisiting item.
-h	--help		Print online help.
-l	--list		List all DOIs whose changes were not committed to the registration agency yet.

Option (short)	Option (long)	Parameter	Description
-q	--quiet		The doi-organiser sends error reports to the mail address configured in the property alert.recipient in dspace.cfg. If you use this option no output should be given to stdout. If you do not use this option the doi-organiser writes information about successful and unsuccessful operations to stdout and stderr. You can find information in dspace.log of course.
-r	--register-all		Transmit information about all DOIs that should be registered.
	--register-doi	DOI | ItemID | handle	If a DOI is marked for registration, you can trigger the registration at the DOI registration agency by this command. Specify either the DOI, the ID of the item, or its handle.
-s	--reserve-all		Transmit to the DOI registration agency information about all DOIs that should be reserved.
	--reserve-doi	DOI | ItemID | handle	If a DOI is marked for registration, you can trigger the registration at the DOI registration agency by this command. Specify either the DOI, the ID of the item, or its handle.
-u	--update-all		If a DOI is reserved for an item, the metadata of the item will be sent to DataCite. This command transmits new metadata for items whose metadata were changed since the DOI was reserved.
	--update-doi	DOI | ItemID | handle	If a DOI needs an update of the metadata of the item it belongs to, you can trigger this update with this command. Specify either the DOI, the ID of the item, or its handle.

Currently you cannot generate new DOIs with this tool. You can only send information about changes in your local DSpace database to the registration agency.

'cron' job for asynchronous reservation/registration

When a DOI should be reserved, registered, deleted or its metadata updated, DSpace just writes this information into its local database. A command line interface is supplied to send the necessary information to the registration agency. This behavior makes it easier to react to outages or errors while using the API. This information should be sent regularly, so it is a good idea to set up a cron job instead of doing it manually.

There are four commands that should be run regularly:

- Update the metadata of all items that have changed since their DOI was reserved.
- Reserve all DOIs marked for reservation
- Register all DOIs marked for registration
- Delete all DOIs marked for deletion

In DSpace, a DOI can have the state "registered", "reserved", "to be reserved", "to be registered", "needs update", "to be deleted", or "deleted". After updating an item's metadata the state of its assigned DOI is set back to the last state it had before. So, e.g., if a DOI has the state "to be registered" and the metadata of its item changes, it will be set to the state "needs update". After the update is performed its state is set to "to be registered" again. Because of this behavior **the order of the commands above matters**: the update command must be executed before all of the other commands above.

The cron job should perform the following commands with the rights of the user your DSpace installation runs as:

```
[dspace]/bin/dspace doi-organiser -u -q
[dspace]/bin/dspace doi-organiser -s -q
[dspace]/bin/dspace doi-organiser -r -q
[dspace]/bin/dspace doi-organiser -d -q
```

The doi-organiser sends error messages as email and logs some additional information. The option -q tells DSpace to be quiet. If you don't use this option the doi-organiser will print messages to stdout about every DOI it successfully reserved, registered, updated or deleted. Using a cron job these messages would be sent as email.

In case of an error, consult the log messages. If there is an outage of the API of your registration agency, DSpace will not change the state of the DOIs so that it will do everything necessary when the cron job starts the next time and the API is reachable again.

The frequency the cron job runs depends on your needs and your hardware. The more often you run the cron job the faster your new DOIs will be available online. If you have a lot of submissions and want the DOIs to be available really quickly, you probably should run the cron job every fifteen minutes. If there are just one or two submissions per day, it should be enough to run the cron job twice a day.

To set up the cron job, you just need to run the following command as the *dspace* UNIX user:

```
crontab -e
```

The following line tells cron to run the necessary commands twice a day, at 1am and 1pm. Please notice that the line starting with the numbers is one line, even it it should be shown as multiple lines in your browser.

```
# Send information about new and changed DOIs to the DOI registration agency:
0 1,13 * * *  [dspace]/bin/dspace doi-organiser -u -q ; [dspace]/bin/dspace doi-organiser -s -q ; [dspace]/
bin/dspace doi-organiser -r -q ; [dspace]/bin/dspace doi-organiser -d -q
```

Limitations of DataCite DOI support

Every DSpace installation expects to be the only application that generates DOIs which start with the prefix and the namespace separator you configured. DSpace does not check whether a DOI it generates is reserved or registered already.

That means if you want to use other applications or even more than one DSpace installation to register DOIs with the same prefix, you'll have to use a unique namespace separator for each of them. Also you should not generate DOIs manually with the same prefix and namespace separator you configured within DSpace. For example, if your prefix is 10.5072 you can configure one DSpace installation to generate DOIs starting with 10.5072/papers-, a second installation to generate DOIs starting with 10.5072/data- and another application to generate DOIs starting with 10.5072/results-.

DOIs will be used in addition to Handles. This implementation does not replace Handles with DOIs in DSpace. That means that DSpace will still generate Handles for every item, every collection and every community, and will use those Handles as part of the URL of items, collections and communities.

DSpace currently generates DOIs for items only. There is no support to generate DOIs for Communities and collections yet.

When using DSpace's support for the DataCite API probably not all information would be restored when using the AIP Backup and Restore (see DS-1836[267]). The DOIs included in metadata of Items will be restored, but DSpace won't update the metadata of those items at DataCite anymore. You can even get problems when minting new DOIs after you restored older once using AIP.

Configure DSpace to use EZID service for registration of DOIs

The EZID IdentifierProvider operates synchronously, so there is much less to configure. You will need to uncomment the `org.dspace.identifier.EZIDIdentifierProvider` bean in `config/spring/api/identifier-service.xml` to enable DOI registration through EZID.

In `config/dspace.cfg` you will find a small block of settings whose names begin with `identifier.doi.ezid`. You should uncomment these properties and give them appropriate values. Sample values for a test account are supplied.

name	meaning
identifier.doi.ezid.shoulder	The "shoulder" is the DOI prefix issued to you by the EZID service. DOIs minted by this instance of DSpace will be the concatenation of the "shoulder" and a locally unique token.
identifier.doi.ezid.user identifier.doi.ezid.password	The username and password by which you authenticate to EZID.
identifier.doi.ezid.publisher	You may specify a default value for the required `datacite.publisher` metadatum, for use when the Item has no publisher.
crosswalk.dissemination.DataCite.publisher	Should match identifier.doi.ezid.publisher.
crosswalk.dissemination.DataCite.hostingInstitution	Name of the hosting institution. If not configured, it will be set to the value of crosswalk.dissemination.DataCite.publisher.
crosswalk.dissemination.DataCite.dataManager	Name of the data manager. If not configured, it will be set to the value of crosswalk.dissemination.DataCite.publisher.

267 https://jira.duraspace.org/browse/DS-1836

Back in `config/spring/api/identifier-service.xml` you will see some other configuration of the `EZIDIdentiferProvider` bean. In most situations, the default settings should work well. But, here's an explanation of options available:

- *EZID Provider / Registrar settings:* By default, the EZIDIdentifierProvider is configured to use the CDLib provider (ezid.cdlib.org[268]) in the `EZID_SCHEME`, `EZID_HOST` and `EZID_PATH` settings. In most situations, the default values should work for you. However, you may need to modify these values (especially the `EZID_HOST`) if you are registered with a different EZID provider. In that situation, please check with your provider for valid "host" and "path" settings. If your provider provides EZID service at a particular path on its host, you may set that in `EZID_PATH`.
    - NOTE: As of the writing of this documentation, the default CDLib provider settings should also work for institutions that use Purdue (ezid.lib.purdue.edu[269]) as a provider. Currently, Purdue and CDLib currently share the same infrastructure, and both `ezid.cdlib.org`[270] and `ezid.lib.purdue.edu`[271] point to the same location.
- *Metadata mappings:* You can alter the mapping between DSpace and EZID metadata, should you choose. The `crosswalk` property is a map from DSpace metadata fields to EZID fields, and can be extended or changed. The `key` of each `entry` is the name of an EZID metadata field; the `value` is the name of the corresponding DSpace field, from which the EZID metadata will be populated.
- *Crosswalking / Transforms:* You can also supply transformations to be applied to field values using the `crosswalkTransform` property. Each `key` is the name of an EZID metadata field, and its `value` is the name of a Java class which will convert the value of the corresponding DSpace field to its EZID form. The only transformation currently provided is one which converts a date to the year of that date, named `org.dspace.identifier.ezid.DateToYear`. In the configuration as delivered, it is used to convert the date of issue to the year of publication. You may create new Java classes with which to supply other transformations, and map them to metadata fields here. If an EZID metadatum is not named in this map, the default mapping is applied: the string value of the DSpace field is copied verbatim.

Limitations of EZID DOI support

DOIs will be used in addition to Handles. This implementation does not replace Handles with DOIs in DSpace. That means that DSpace will continue to generate Handles for every item, every collection and every community, and will use those Handles as part of the URL of items, collections and communities.

Currently, the EZIDIdentifierProvider has a known issue where it stores its DOIs in the `dc.identifier` field, instead of using the `dc.identifier.uri` field (which is the one used by DataCite DOIs and Handles). See DS-2199[272] for more details. This will be corrected in a future version of DSpace.

DSpace currently generates DOIs for items only. There is no support to generate DOIs for Communities and Collections yet.

4.6.3.3 Adding support for other Registration Agencies

If you want DSpace to support other registration agencies, you just have to write a Java class that implements the interface DOIConnector ([dspace-source]/dspace-api/src/main/java/org/dspace/identifier/doi/DOIConnector.java). You might use the DataCiteConnector ([dspace-source]/dspace-api/src/main/java/org/dspace/identifier/doi/DataCiteConnector.java) as an example. After developing your own DOIConnector, you configure DSpace as if you were using the DataCite API directly. Just use your DOIConnector when configuring the IdentifierService instead of the DataCiteConnector.

---

268 http://ezid.cdlib.org/
269 https://ezid.lib.purdue.edu/
270 http://ezid.cdlib.org
271 http://ezid.lib.purdue.edu
272 https://jira.duraspace.org/browse/DS-2199

4.6.4 Item Level Versioning

4.6.4.1 What is Item Level Versioning?

Versioning is a new functionality to build the history of an item. Users will have the opportunity to create new version of an existing item any time the will make a change.

4.6.4.2 Important warnings - read before enabling

DSpace 7.0 does not yet support

Item versioning is not fully available in DSpace 7.0. While you *can* view existing item versions, it is not possible to create new versions. Full support for item versioning is scheduled to be restored in a later 7.x release (currently 7.1), see DSpace Release 7.0 Status[273].

AIP Backup & Restore functionality only works with the Latest Version of Items

If you are using the AIP Backup and Restore(see page 411) functionality to backup / restore / migrate DSpace Content, you must be aware that the "Item Level Versioning" feature is **not yet compatible** with AIP Backup & Restore. **Using them together may result in accidental data loss.** Currently the AIPs that

273 https://wiki.lyrasis.org/display/DSPACE/DSpace+Release+7.0+Status

DSpace generates only store the *latest version* of an Item. Therefore, past versions of Items will always be lost when you perform a restore / replace using AIP tools. See DS-1382[274].

DSpace 6+ changed the way Handles are created for versioned Items

Starting with 6.0, the way DSpace crates Handles for versioned Items was changed. If you want to keep the old behavior of DSpace 4 and 5 you have to enable the `VersionedHandleIdentifierProviderWithCanonicalHandles` in the XML configuration files `[dspace]/config/spring/api/identifier-service.xml`. See IdentifierServiceOverride(see page 312) below for details and the comments in the configuration file.

4.6.4.3 Enabling Item Level Versioning

By default, Item Level Versioning is disabled in DSpace.

As Item Level versioning is not yet supported in DSpace 7.0, documentation will be coming soon.

4.6.4.4 Initial Requirements

The Item Level Versioning implementation builds on following requirements identified by the stakeholders who supported this contribution: Initial Requirements Analysis[275]

1. What should be *Versionable*
    a. Versioning happens at the level of an Individual Item
    b. Versioning should preserve the current state of *metadata*, *bitstreams* and *resource policies* attached to the item.
2. Access, Search and Discovery
    a. Only the most recent version of an item is available via the search interface
    b. Previous versions of Items should continue to be visible, citable and accessible
    c. The Bitstreams for previous versions are retained. If something was once retrievable, it should always be retrievable.
3. Identifiers
    a. Each version of an Item is represented by a separate "*versioned*" identifier
    b. A base "*versionhistory*" Identifier points to the most recent version of the Item.
    c. A revision identifier also exists that is unique to the specific version.
    d. When a new version of an Item is deposited, a new revision identifier will be created.
4. Presentation
    a. On the item page, there is a link to view previous/subsequent versions.
    b. By examining the metadata or identifiers, it is possible to determine whether an item is the most recent version, the original version, or an intermediate version.
5. Access Control and Rights
    a. Certain roles should be able to generate a new version of the item via submission.
    b. To submitters, collection manager, administrators will be given to option to create new version of an item.
    c. Rights to access a specific Item should transmute as well to previous versions
    d. Rights to access a specific Bitstream should also transmute to previous versions.
6. Data Integrity

274 https://jira.duraspace.org/browse/DS-1382
275 https://wiki.duraspace.org/display/DSPACE/Item+Versioning+Support

a. The relationships between versions should not be brittle and breakable by manipulating Item metadata records.
b. The relationships between versions should be preserved and predictable in various Metadata Exports (OAI, Packagers, ItemExport)
c. The relationships between versions should be maintained in SWORD and AIP packaging and be maintained in updates and restorations.

4.6.4.5 User Interface

General behaviour: Linear Versioning

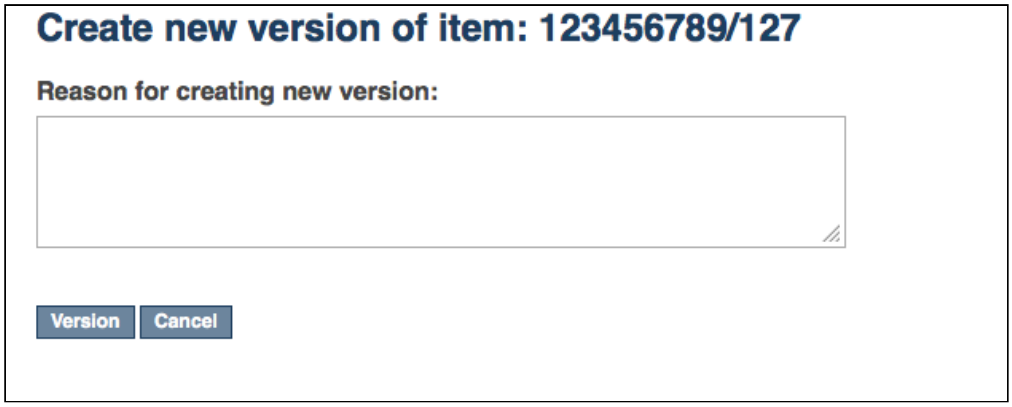
From the user interface, DSpace offers **linear** versioning. As opposed to hierarchical versioning, linear version has following properties:

- A new version can only be created started from the latest available version
- When new version has been created and still needs to pass certain steps of the workflow, it is temporarily impossible to create another new version until the workflow steps are finished and the new version has replaced the previous one.

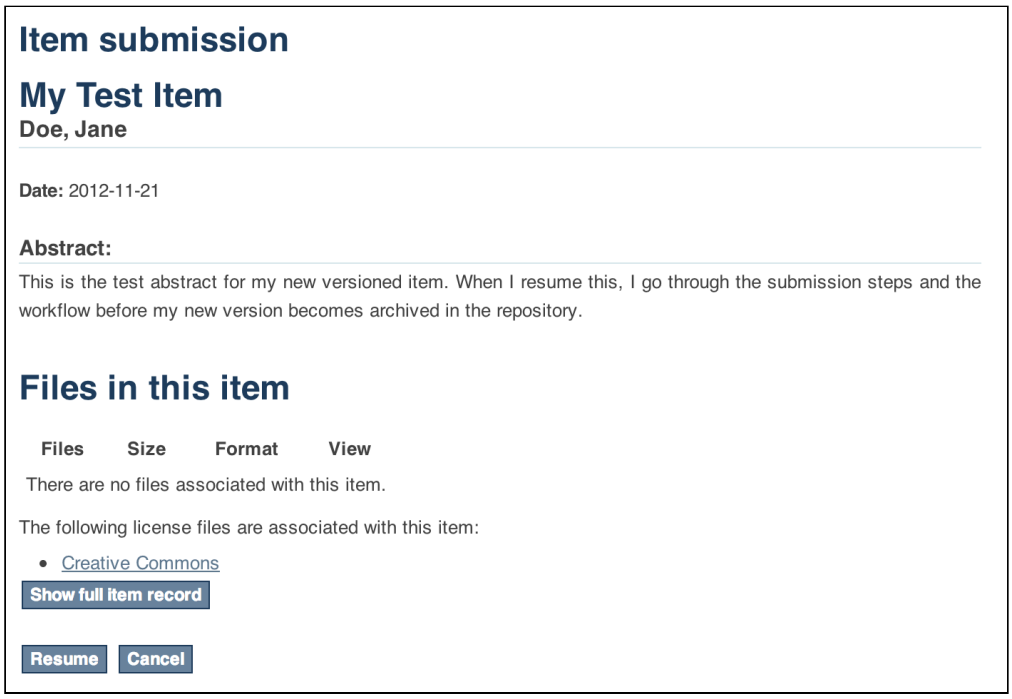
Creating a new version of an item

Administrators and collection/community administrators can create new versions of an item from the Item View page.

1. Click "Create a new version" from the Context Menu in the navigation bar.
2. Provide the reason for creating a new version that will lateron be stored and displayed in the version summary.



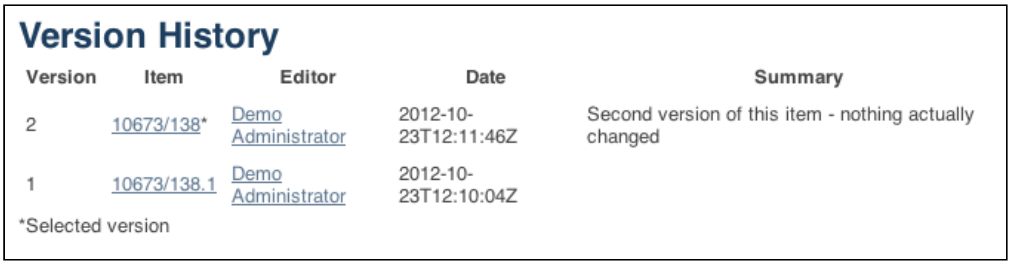
3. Your new version is now creates as a new Item in your Workspace. It requires you to go through the submission and workflow steps like you would do for a normal, new submission to the collection. The rationale behind this is that if you are adding new files or metadata, you will also need to accept the license for them. In addition to this, the versioning functionality does not bypass any quality control embedded in the workflow steps.



After the submission steps and the execution of subsequent workflow steps, the new version becomes available in the repository.

View the history and older versions of an item

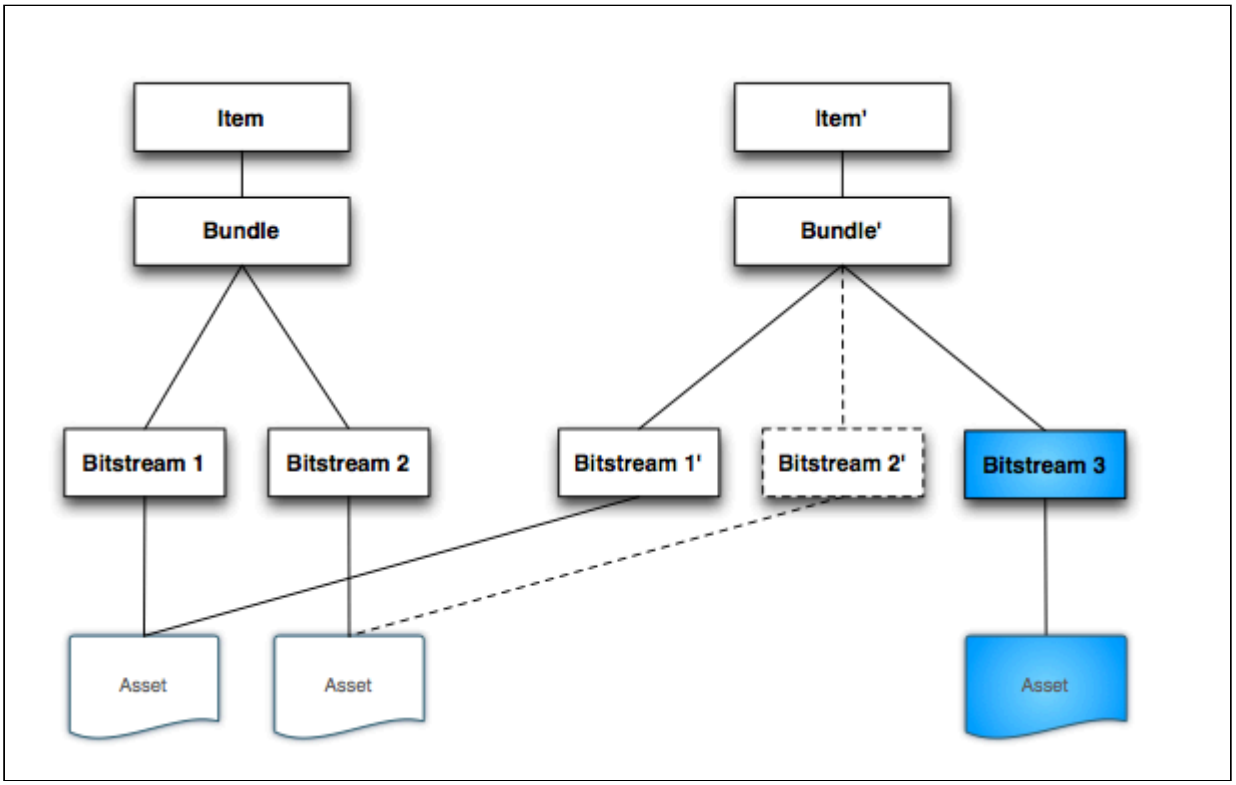
An overview of the version history, including links to older versions of an item, is available at the bottom of an Item View page. The repository administrator can decide whether the version history should be available to all users or restricted to administrators. Since DSpace 6 the repository administrator can decide whether all users should be able to see the version submitter/editor or if this information is restricted and can be seen by administrators only. As this may expose data that my be considered personal (name and email address of the submitter), we encourage everyone to leave the default setting and reveal those information to administrators only.



4.6.4.6 Architecture

Versioning model

For every new Version a separate DSpace Item will be created that replicates the metadata, bundle and bitstream records. The bitstream records will point to the same file on the disk.



The *Cleanup* method has been modified to retain the file if another Bitstream record point to it (the dotted lines in the diagram represent a bitstream deleted in the new version), in other words the file will be deleted only if the Bitstream record processed is the only one to point to the file *(count(INTERNAL_ID)=1).*

4.6.4.7 Configuration

Versioning Service Override

You can override the default behaviour of the Versioning Service using following XML configuration file, deployed under your dspace installation directory:

[dspace_installation_dir]/config/spring/api/versioning-service.xml[276]

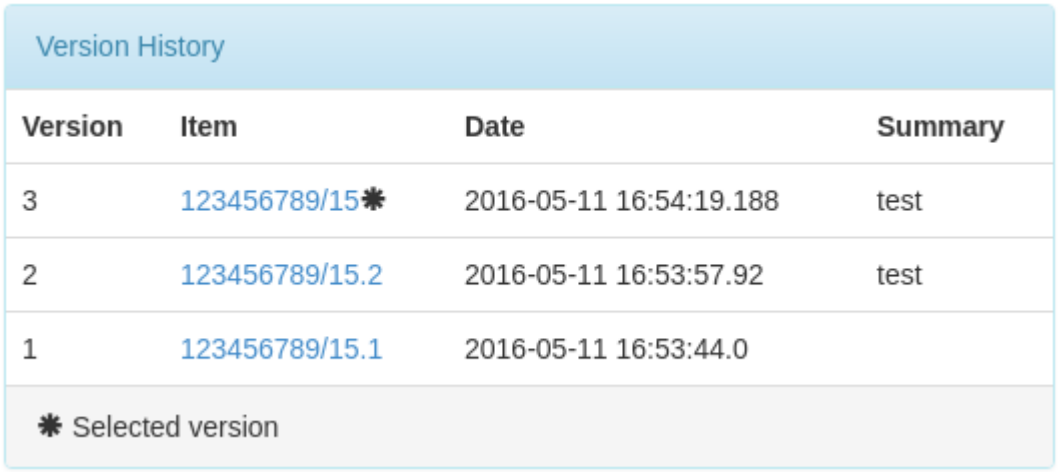
In this file, you can specify which metadata fields are automatically "reset" (i.e. cleared out) during the creation of a new item version. By default, all metadata values (and bitstreams) are copied over to the newly created version, with the exception of **dc.date.accessioned** and **dc.description.provenance.** You may specify additional metadata fields to reset by adding them to the "ignoredMetadataFields" property in the "versioning-service.xml" file:

276 https://github.com/DSpace/DSpace/blob/master/dspace/config/spring/api/versioning-service.xml

```
<!-- Default Item Versioning Provider, defines behavior for replicating
     Item, Metadata, Budles and Bitstreams. Autowired at this time. -->
<bean class="org.dspace.versioning.DefaultItemVersionProvider">
    <property name="ignoredMetadataFields">
        <set>
           <value>dc.date.accessioned</value>
           <value>dc.description.provenance</value>
        </set>
    </property>
</bean>
```

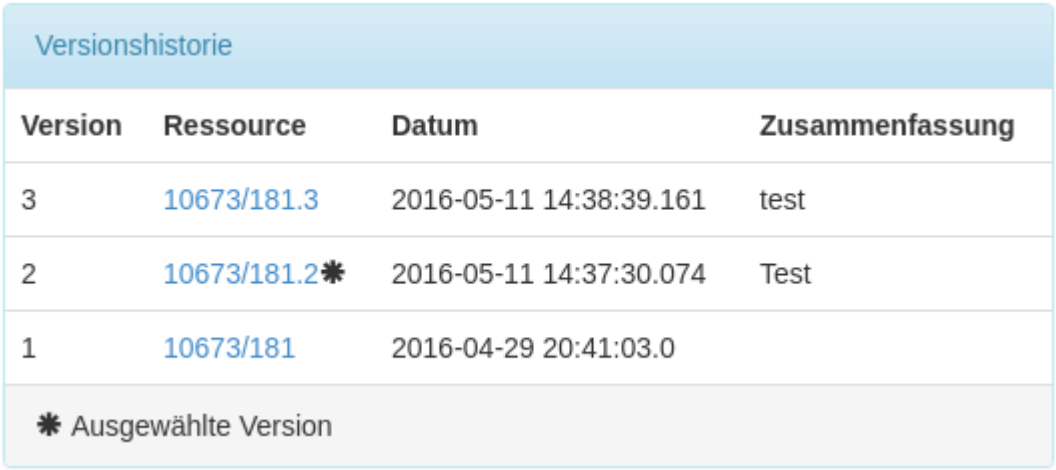
Identifier Service Override

Persistent Identifiers are used to address Items within DSpace. The handle system is deeply integrated within DSpace, but since version 4 DSpace can also mint DOIs. DSpace 4 and 5 supported one type of versioned handle: The initial version of an Item got a handle, e.g. 10673/100. This handle was called the canonical one. When a newer version was created, the canonical handle was moved to identify the newest version. The previously newest version got a new handle build out of the canonical handle extended by a dot and the version number. In the image below you see a version history of an item using this handle strategy.



The canonical handle will always point to the newest version of an Item. This makes sense if you hide the version history. Normal users won't be able to find older versions and will always see just the newest one. Please keep in mind, that older versions can be accessed by "guessing" the versioned Handle if you do not remove the read policies manually. The downside of this identifier strategy is that there is no permanent handle to cite the currently newest version, as it will get a new Handle when a newer version is created.

With DSpace 6 versioned DOIs (using DataCite as DOI registration agency) were added and the default versioned Handle strategy was changed. Starting with DSpace 6 the `VersionedHandleIdentifierProvider` creates a handle for the first version of an item. Every newer version gets the same handle extended by a dot and the version number. To stay by the example from above, the first version of an Item gets the Handle 10673/100, the second version 10673/100.2, the third version 10673.3 and so on. This strategy has the downside that there is no handle pointing always to the newest version. But each version gets an identifier that can be use to cite exactly this version. If page numbers changes in newer editions the old citations stay valid. This strategy makes sense especially if you present the version history to all users. In the image below you see a version history using this strategy.



Versionshistorie

Version	Ressource	Datum	Zusammenfassung
3	10673/181.3	2016-05-11 14:38:39.161	test
2	10673/181.2✱	2016-05-11 14:37:30.074	Test
1	10673/181	2016-04-29 20:41:03.0	

✱ Ausgewählte Version

In DSpace 4 and 5 only the strategy using canonical handles (one handle that always points to the newest version) were implemented. In DSpace 6 the strategy of creating a new handle for each version was implemented. With DSpace 6 this new strategy become the default. The strategy using canonical handle still exists in DSpace but you have to enable the `VersionedHandleIdentifierWithCanonicalHandles` in the file `[dspace]/config/spring/api/identifier-serice.xml`. With DSpace 6 versioned DOIs were introduced using the strategy that every new version gets a new DOI (extended by a dot and the version numbers for versions >= 2). To use versioned Handle you have to enable DOIs, you have to use DataCite as registration agency and you have to enable the `VersionedDOIIdentifierProvider` in the named configuration file.

You can configure which persistent identifiers should be used by editing following XML configuration file, deployed under your dspace installation directory:

[dspace_installation_dir]/config/spring/api/identifier-service.xml[277]

No changes to this file are required to enable Versioning. This file is currently only relevant if you want to keep the identifier strategy from DSpace 4 and 5 or if you want to enable DOIs(see page 296) or even versioned DOIs.

Version History Visibility

Version History

Version	Item	Editor	Date	Summary
2	10673/138*	Demo Administrator	2012-10-23T12:11:46Z	Second version of this item - nothing actually changed
1	10673/138.1	Demo Administrator	2012-10-23T12:10:04Z	

*Selected version

By default, **all** users will be able to see the version history. To ensure that only administrators can see the Version History, enable `versioning.item.history.view.admin` in the `[dspace]/config/modules/versioning.cfg` OR in your `local.cfg` file.

```
versioning.item.history.view.admin=false
```

277 https://github.com/DSpace/DSpace/blob/master/dspace/config/spring/api/identifier-service.xml

Allowing submitters to version their items

With DSpace 6.0 it became possible to allow submitters to create new versions of their own items. The new versions are going through the normal workflow process if the collection is configured this way. To allow submitters to create new versions of Item they originally submitted, you have to change the configuration property `versioning.submitterCanCreateNewVersion` and set it to `true`. It is part of the configuration file `[dspace]/config/modules/versioning.cfg` but can be overridden in your `local.cfg` too.

4.6.4.8 Identified Challenges & Known Issues

Item Level Versioning has a substantial conceptual impact on many DSpace features. Therefore it has been accepted into DSpace as an optional feature. Following challenges have been identified in the current implementation. As an early adopter of the Item Level Versioning feature, your input is greatly appreciated on any of these.

Only Administrators and Collection/Community Administrators can add new versions

There is currently no configuration option to allow submitters to create new versions of an item. This functionality is restricted to Administrators and Collection/Community Administrators. In a context where original submission of DSpace items is done by non-administrator users, it might also make sense to allow them to create new versions. Especially given the fact that new versions have to pass through the workflow anyway.

Conceptual compatibility with Embargo

Lifting an embargo currently does not interact with Item Level Versioning. Additional implementation would be required to ensure that lifting an embargo actually creates a new version of the item.

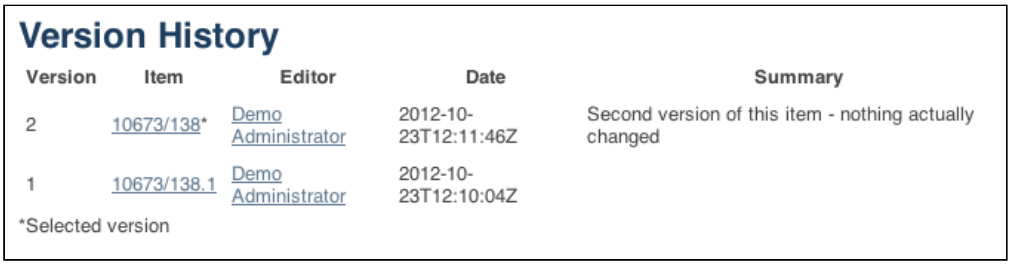
Conceptual compatibility with Item Level Statistics

Both on the level of pageviews and downloads, different versions of an item are treated as different items. As a result, an end user will have the impression that the stats are being "reset" once a new version is installed, because the previous downloads and pageviews are allocated to the previous version.

One possible solution would be to present an end user with aggregated statistics across all viewers, and give administrators the possibility to view statistics per version.

Exposing version history

The version history is added on the bottom of a versioned item page. A repository administrator can either decide to show this to all users, or restrict it to admins only. If it is shown to admins only, an end user will have no way to find the way to an older version. Since DSpace 6 you can also configure if the submitter's name and email address should be part of the version history or if they should be hidden. To show the submitter might actually be useful if the editor account is always a generic institutional email address, but may conflict with local privacy laws if any personal details are included.



Version History

Version	Item	Editor	Date	Summary
2	10673/138*	Demo Administrator	2012-10-23T12:11:46Z	Second version of this item - nothing actually changed
1	10673/138.1	Demo Administrator	2012-10-23T12:10:04Z	

*Selected version

Hide Submitter details in version table

In either the `[dspace]/config/modules/versioning.cfg` configuration file or your `local.cfg`, you can customize the configuration option `versioning.item.history.include.submitter`. By default this is set to false, which means that information about the submitter is only viewable by administrators. If you want to expose the submitters information to everyone (which be useful if all submitters uses generic institutional email addresses, but may conflict with local privacy laws if personal details are included) you can set this configuration property to true.

```
# The property item.history.include.submitter controls whether the name of
# the submitter of a version should be included in the version history of
# an item.
versioning.item.history.include.submitter=false
```

4.6.5 Mapping/Linking Items to multiple Collections

4.6.5.1 Introduction

The Item Mapper is a tool in the DSpace web user interface allowing repository managers to display the same item in multiple collections at once. Thanks to this feature, a repository manager is not forced to duplicate items to display them in different collections

4.6.5.2 Using the Item Mapper

In the User Interface, the item mapper can be accessed when editing an Item.

- Login as someone with Edit permissions
- Search/browse to the Item
- Click the "Edit this Item" button
- Click "Mapped" collections" button on the "Status" tab
  - You'll be shown a list of currently mapped collections (if any)
  - You can also map the item to a new collection by clicking on "Map new collections" tab, and searching for the Collection(s)

4.6.5.3 Implications

Mapping collection vs Owning collection

The relation between an item and the collection in which it is mapped is different from the relation that this item has with the collection to which it was originally submitted. This second collection is referred to as the "owning" collection. When an item is deleted from the owning collection, it automatically disappears from the mapping collection. From within the mapping collection, the only thing that can be deleted is the mapping relation. Removing this mapping relation does not affect the presence of the item in the owning collection.

Mapping an item does not modify access rights

When an item gets mapped into a collection, it does not receive new access rights. It retains the authorizations that it inherited from the collection that "owns" it. Collection admins who do not have read access to an item will not be able to map them to other collections.

4.6.6 Metadata Recommendations

4.6.6.1 Recommended Metadata Fields

DSpace provides a broad list of metadata fields out of the box (see: Metadata and Bitstream Format Registries[278]), and a variety of options for adding content to DSpace (both from the UI and from other services). No matter which Ingest option you use, DSpace recommends ensuring that the following metadata fields are specified:

- **Title** (`dc.title`)
  - When submitting an Item via the DSpace web user interface, this field is **required**.
  - If you add an Item to DSpace through another means (SWORD, etc), it is recommend to specify a title for an Item. Without a title, the Item will show up in DSpace a "Untitled".
- **Publication Date** (`dc.date.issued`)
- When submitting an Item via the DSpace web user interface, this field is **required** (by default).
  - However, your System Administrator can choose to enable the "Initial Questions" step within the Submission User Interface[279]. Enabling this step will cause the following to occur: If the item is said to be "published", then the Publication Date will be required. If the item is said to be "unpublished" then the Publication Date will be auto-set to today's date (date of submission). *WARNING:* Google Scholar has recommended against automatically assigning this "dc.date.issued" field to the date of submission as it often results in incorrect dates in Google Scholar results. See DS-1481[280] and DS-1745[281] for more details.
- If you add and Item to DSpace through another means (SWORD, etc), it is recommended to specify the date in which an Item was published, in ISO-8601[282] (e.g. 2007, 2008-01, or 2011-03-04). This ensures DSpace can accurately report the publication date to services like Google Scholar. If an item is unpublished, you can

278 https://wiki.lyrasis.org/display/DSDOC4x/Metadata+and+Bitstream+Format+Registries
279 https://wiki.lyrasis.org/display/DSDOC4x/Submission+User+Interface
280 https://jira.duraspace.org/browse/DS-1481
281 https://jira.duraspace.org/browse/DS-1745
282 http://en.wikipedia.org/wiki/ISO_8601

either chose to leave this blank, or pass in the literal string "today" (which will tell DSpace to automatically set it to the date of ingest)

> **DSpace will not auto-assign a "dc.date.issued"**
>
> As of DSpace 4.0, the system will not assign a "dc.date.issued" when unspecified. Previous versions of DSpace (3.0 or below) would set "dc.date.issued" to the date of accession (dc.date.accessioned), if it was unspecified during ingest.
>
> If you are adding content to DSpace without using the DSpace web user interface, there are two recommended options for assigning "dc.date.issued"
>
> - If the item is previously published before, please set "dc.date.issued" to the date of publication in ISO-8601[283](e.g. 2007, 2008-01, or 2011-03-04)
> - If the item has never been previously published, you may set "dc.date.issued='today'" (the literal string "today"). This will cause DSpace to automatically assign "dc.date.issued" to the date of accession (dc.date.accessioned), as it did previously
>   - You can also chose to leave "dc.date.issued" as unspecified, but then the new Item will have an empty date within DSpace.

Obviously, we recommend specifying as much metadata as you can about a new Item. For a full list of supported metadata fields, please see: Metadata and Bitstream Format Registries[284]

4.6.6.2 Local Fields

You may encounter situations in which you will require an appropriate place to store information that does not immediately fit with the description of a field in the default registry. The recommended practice in this situation is to create new fields in a separate schema. You can choose your own name and prefix for this schema such as *local.* or *myuni.*

It is generally discouraged to use any of the fields from the default schema as a place to store information that doesn't correspond with the fields description. This is especially true if you are ever considering the option to open up your repository metadata for external harvesting.

4.6.7 Moving Items

4.6.7.1 Moving Items via Web UI

It is possible for Administrators to move items one at a time from the User Interface.

- Login as an Administrator
- Browse/search for the item.
- Click "Edit this Item" on the item page
- When editing an item, on the 'Edit item' screen, click the "Move..." button.
- Search for the new Collection for the item to appear in. By default, when the item is moved, it will take its authorizations (who can READ / WRITE it) with it.

283 http://en.wikipedia.org/wiki/ISO_8601
284 https://wiki.lyrasis.org/display/DSDOC4x/Metadata+and+Bitstream+Format+Registries

- If you wish for the item to take on the default authorizations of the destination collection, tick the 'Inherit policies' checkbox. This is useful if you are moving an item from a private collection to a public collection, or from a public collection to a private collection.
- Note: When selecting the 'Inherit policies' option, ensure that this will not override system-managed authorizations such as those imposed by the embargo system.

4.6.7.2 Moving Items via the Batch Metadata Editor

Items may also be moved in bulk by using the CSV batch metadata editor (see Editing Collection Membership(see page 293) section under Batch Metadata Editing(see page 287)).

4.6.8 ORCID Integration

4.6.8.1 Introduction

The ORCID integration adds ORCID compatibility to the existing solutions for Authority control in DSpace[285]. String names of authors are still being stored in DSpace metadata. The authority key field is leveraged to store a uniquely generated internal ID that links the author to more extended metadata, including the ORCID ID and alternative author names.

This extended metadata is stored and managed in a dedicated SOLR index, the DSpace authority cache.

285 https://wiki.lyrasis.org/pages/viewpage.action?pageId=25468046

4.6.8.2 Use case and high level benefits

The vision behind this project consists of the following two aspects:

Lowering the threshold to adopt ORCID for the members of the DSpace community

ORCID's API has enabled developers across the globe to build points of integration between ORCID and third party applications. Up until today, this meant that members of the DSpace community were still required to implement front-end and back-end modifications to the DSpace source code in order to leverage these APIs. As DSpace aims to provide turnkey Institutional Repository functionality, the platform is expected to provide more functionality out of the box. Only an elite selection of members in the DSpace community has software development resources readily available to implement this kind of functionality. By contributing a solution directly to the core DSpace codebase, this threshold to adopt ORCID functionality in DSpace repositories is effectively lowered. The ultimate goal is to allow easy adoption of ORCID without customization of the DSpace software, by allowing repository administrators to enable or disable functionality by means of user friendly configuration.

Address generic use cases with appealing end user functionality

This proposal aims to provide user friendly features for both repository administrators as well as non- technical end users of the system. The addition of ORCID functionality to DSpace should not come at the cost of making the system more difficult for administrators and end users to use. Scope With this vision in mind, the project partners wanted to tackle the first phases for repository managers of existing DSpace repositories: ensuring that ORCIDs are properly associated with new works entering the system, as well as providing functionality to efficiently batch-update content already existing in the system, with unambiguous author identity information.

4.6.8.3 Enabling the ORCID authority control

If you wish to enable this feature, some changes are required to the `dspace.cfg` file. The first step is to activate the authority as a valid option for authority control, this is done by adding/setting an additional plugin in the `plugin.named.org`[286]`.dspace.content.authority.ChoiceAuthority` property. An example of this can be found below.

```
plugin.named.org.dspace.content.authority.ChoiceAuthority = \
    org.dspace.content.authority.SolrAuthority = SolrAuthorAuthority
```

The feature relies on the following configuration parameters in `dspace.cfg`. To activate the default settings it suffices to remove the comment hashes ("#") for the following lines. See the section at the bottom of this page what these parameters mean exactly and how you can tweak the configuration.

```
solr.authority.server=${solr.server}/authority
choices.plugin.dc.contributor.author = SolrAuthorAuthority
choices.presentation.dc.contributor.author = authorLookup
authority.controlled.dc.contributor.author = true
authority.author.indexer.field.1=dc.contributor.author
```

The final part of configuration is to add the authority consumer in front of the list of event consumers. Add "authority" in front of the list as displayed below.

286 http://plugin.named.org

```
event.dispatcher.default.consumers = authority, versioning, discovery, eperson, harvester
```

4.6.8.4 Importing existing authors & keeping the index up to date

When first enabled the authority index will be empty, to populate the authority index run the following script:

```
[dspace]/bin/dspace index-authority
```

This will iterate over every metadata under authority control and create records of them in the authority index. The metadata without an authority key will each be updated with an auto generated authority key. These will not be matched in any way with other existing records. The metadata with an authority key that does not already exist in the index will be indexed with those authority keys. The metadata with an authority key that already exist in the index will be re-indexed the same way. These records remain unchanged.

Different possible use cases for Index-authority script

Metadata value WITHOUT authority key in metadata

“Luyten, Bram” is present in the metadata without any authority key.
GOAL: “Luyten, Bram” gets added in the cache ONCE

All occurences of “Luyten, Bram” in the DSpace item metadata will become linked with the same generated uid.

Metadata that already has an authority key from an external source (NOT auto-generated by DSpace)

“Snyers, Antoine” is present with authority key “u12345”

The old authority key needs to be preserved in the item metadata and duplicated in the cache.
“u12345” will be copied to the authority cache and used as the authority key there.

Metadata that has already a new dspace generated uid authority key

Item metadata already contains an author with name “Haak, Danielle” and a uid in the authority field 3dda2571-6be8-4102-a47b-5748531ae286

This uid is preserved and no new record is being created in the authority index.

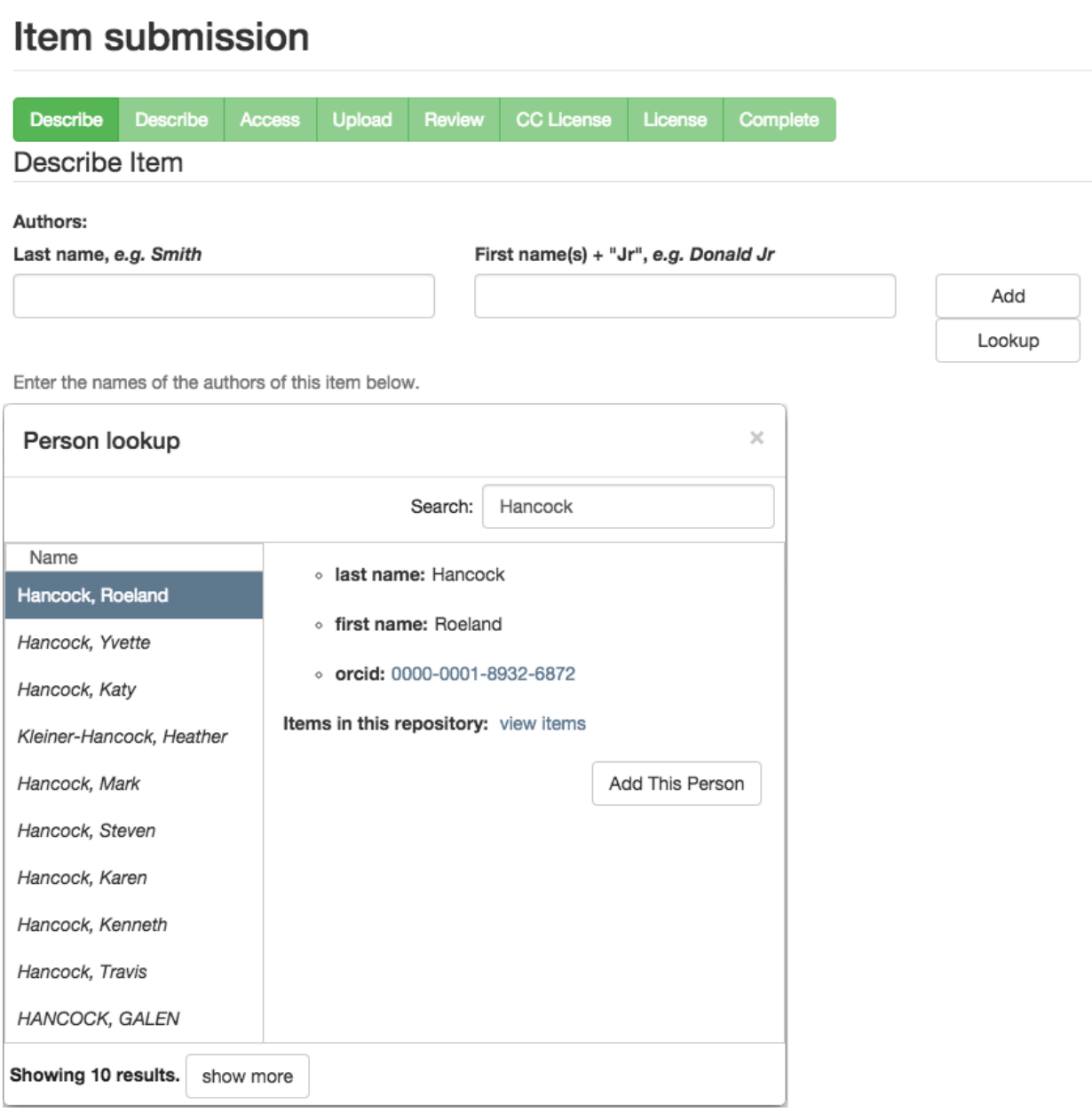
Processing on records in the authority cache

Running this script again will update the index and keep the index clean. For example if an author occurs in a single item and that item is deleted the script will need to be run again to remove it from the index. When run again it will remove all records that no longer have a link to existing authors in the database.

Submission of new DSpace items - Author lookup

The submissions forms have not changed much. The only thing you can notice is an extra button next to the input fields for the author names. Next to the Add button, which is common for all repeatable fields, there is the Lookup & Add button.

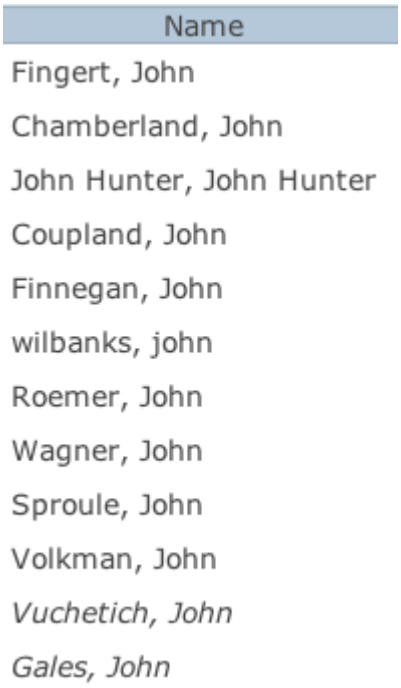
Note: the below screenshots are from DSpace 6.x. They have not yet been updated for 7.x or above.



It's by clicking on that button that the Look-up User Interface appears. If an author name was filled in but not added yet, the Lookup User Interface will immediately perform a search for that name. Otherwise the search field remains empty and a list of known authors is displayed. The list of authors is updated as you type in the search box.

Authors that already appear somewhere in the repository are differentiated from the authors that have been retrieved from ORCID.

The authors retrieved from ORCID have their name italicized and they're listed after the authors that are found in the repository.



Click on one of these names to see more information about them. The message "There's no one selected" will vanish, making room for the author's information. The available information can vary: Authors imported from ORCID have an orcid where the others do not. Authors that have been added without look-up only show their last name and first name.

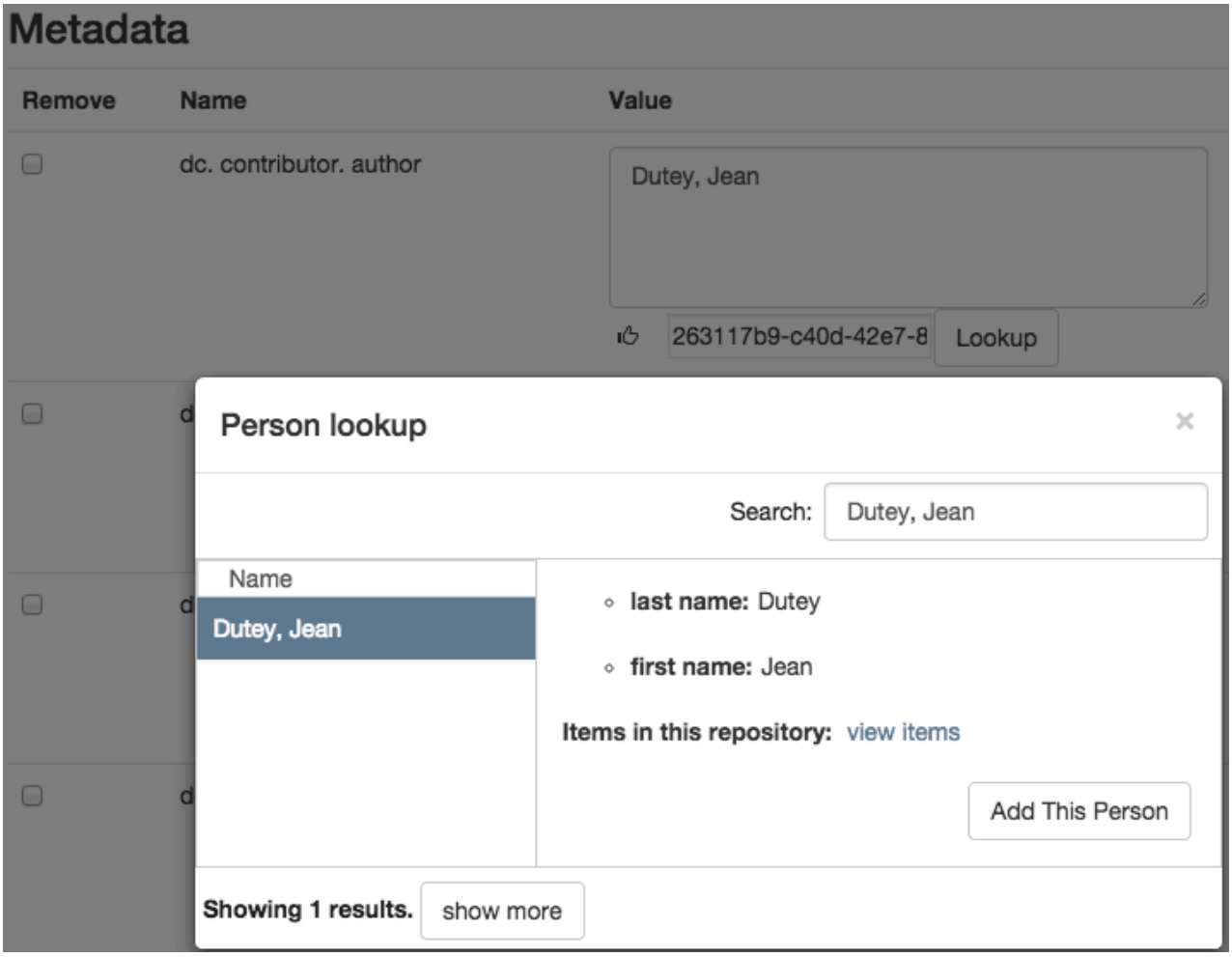
To add an author from the Look-up User Interface, you select the author in the list and then you click on the "Add This Person" button.

To add an author without look-up, you don't go through the Look-up User Interface. Instead you simply use the "Add" button in the submissions forms.

Admin Edit Item

In the edit metadata page, under the values for the dc.contributor.author fields, an extra line shows the author ID together with a lock icon and a Lookup button. The author ID cannot be changed manually. However the Lookup button will help you change the author name and ID at the same time.

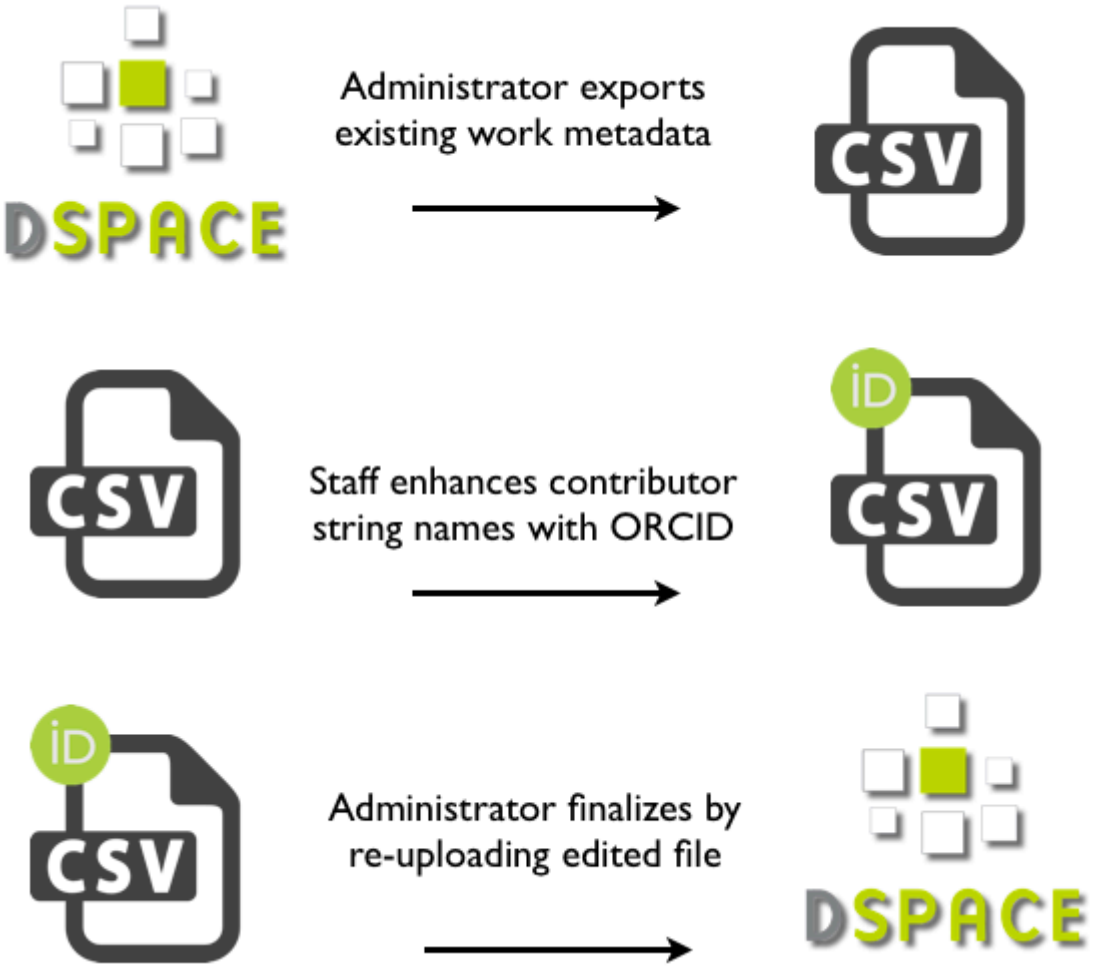
Clicking the Lookup button brings back the Lookup User Interface. This works just the same way as in the submission forms.



Editing existing items using Batch CSV Editing

Instructions on how to use the Batch CSV Editing are found on the Batch Metadata Editing documentation page[287].

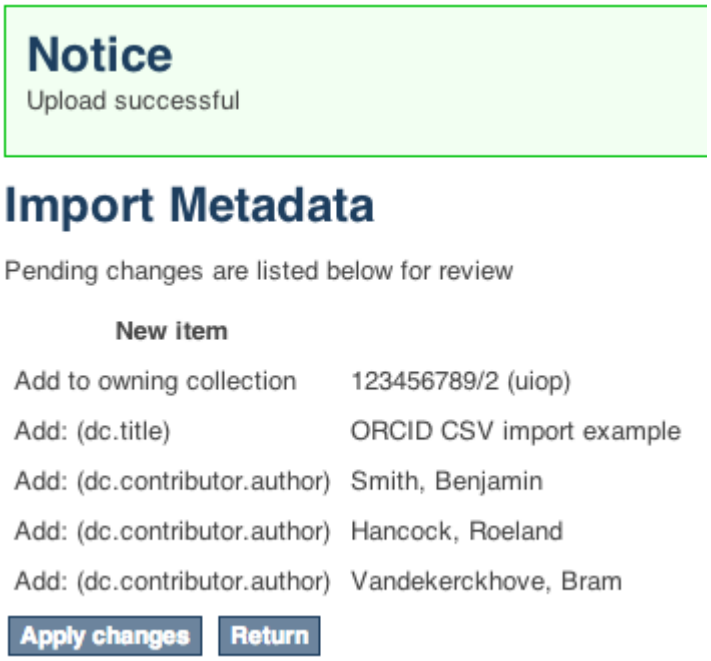
287 https://wiki.lyrasis.org/display/DSDOC5x/Batch+Metadata+Editing



ORCID Integration is provided through the Batch CSV Editing feature with an extra available headers "ORCID:dc.contributor.author". The usual CSV headers only contain the metadata fields: e.g. "dc.contributor.author". In addition to the traditional header, another dc.contributor.author header can be added with the "ORCID:" prefix. The values in this column are supposed to be ORCIDs.

id	collection	dc.title	ORCID:dc.contributor.author	dc.contributor.author
+	123456789/2	ORCID CSV import example	0000-0001-8932-6872||0000-0001-9152-849X	Smith, Benjamin

For each of the ORCID authors a lookup will be done and their names will be added to the metadata. All the non-ORCID authors will be added as well. The authority keys and solr records are added when the reported changes are applied.



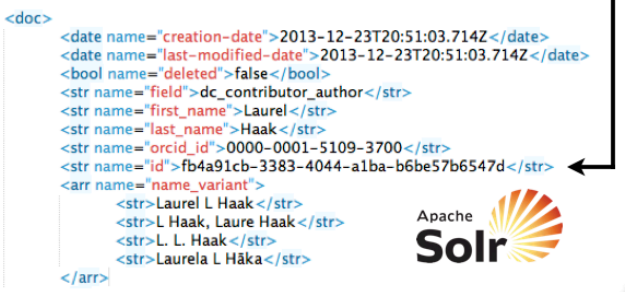
Storage of related metadata

ORCID authorities not only link a digital identifier to a name. It regroups a load of metadata going from alternative names and email addresses to keywords about their works and much more. The metadata is obtained by querying the ORCID web services. In order to avoid querying the ORCID web services every time, all these related metadata is gathered in a "metadata authority cache" that DSpace can access directly.

In practice the cache is provided by an apache solr server. When a look-up is made and an author is chosen that is not yet in the cache, a record is created from an ORCID profile and added to the cache with the list of related metadata. The value of the Dublin Core metadata is based on the first and last name as they are set in the ORCID profile. The authority key for this value links directly to the solr document's id. DSpace does not provide a way to edit these records manually.

String representation of the name in standard Dublin Core metadata enhanced with DSpace compliant authority id

Second level "metadata authority cache" stores extended contributor metadata, including ORCID ID and alternative names



The information in the authority cache can be updated by running the following command line operation:

Command used:	`[dspace]/bin/dspace dsrun org.dspace.authority.UpdateAuthorities`
Arguments	description
-i	update specific solr records with the given internal ids (comma-separated)
-h	prints this help message

This will iterate over every solr record currently in use (unless the -i argument is provided), query the ORCID web service for the latest data and update the information in the cache. If configured, the script will also update the metadata of the items in the repository where applicable.

The configuration property can be set in `config/modules/solrauthority.cfg`, or overridden in your `local.cfg` (see Configuration Reference(see page 552)).

```
solrauthority.auto-update-items = false | true
```

When set to true and this is script is run, if an authority record's information is updated the whole repository will be scanned for this authority. Every metadata field with this authority key will be updated with the value of the updated authority record.

4.6.8.5 Configuration

In the Enabling the ORCID authority control(see page 0) section, you have been told to add this block of configuration.

> **NOTE: for DSpace 6x you can use local.cfg for these**
>
> For all of the configuration options described below, you can use either dspace.cfg or local.cfg. Either will work. It is possible that, when you compile your code with Maven, and you have tests enabled, your build will fail. DSpace unit tests utilize parts of dspace.cfg, and the configuration options you will utilize below are known to cause unit test errors. The easiest way to avoid this situation is to use the local.cfg file.

```
solr.authority.server=${solr.server}/authority
choices.plugin.dc.contributor.author = SolrAuthorAuthority
choices.presentation.dc.contributor.author = authorLookup
authority.controlled.dc.contributor.author = true
authority.author.indexer.field.1=dc.contributor.author
```

The ORCID Integration feature is an extension on the authority control in DSpace. Most of these properties are extensively explained on the Authority Control of Metadata Values documentation page[288]. These will be revisited but first we cover the properties that have been newly added.

288 https://wiki.lyrasis.org/display/DSPACE/Authority_Control_of_Metadata_Values

- The **solr.authority.server** is the url to the solr core. Usually this would be on the **solr.server** next to the oai, search and statistics cores.
- **authority.author.indexer.field.1** and the subsequent increments configure which fields will be indexed in the authority cache. However before adding extra fields into the solr cache, please read the section about Adding additional fields under ORCID(see page 0).

That's it for the novelties. Moving on to the generic authority control properties:

- With the **authority.controlled** property every metadata field that needs to be authority controlled is configured. This involves every type of authority control, not only the fields for ORCID integration.
- The **choices.plugin** should be configured for each metadata field under authority control. Setting the value on SolrAuthorAuthority tells DSpace to use the solr authority cache for this metadatafield, cfr. Storage of related metadata(see page 0).
- The **choices.presention** should be configured for each metadata field as well. The traditional values for this property are `select`|`suggest`|`lookup`. A new value, `authorLookup`, has been added to be used in combination with the SolrAuthorAuthority choices plugin. While the other values can still be used, the authorLookup provides a richer user interface in the form of a popup on the submission page.
- The browse indexes need to point to the new authority-controlled index: `webui.browse.index.2 = author:metadata:dc.contributor.*,dc.creator:text` should become **webui.browse.index.2 = author:metadataAuthority:dc.contributor.author:authority**
- More existing configuration properties are available but their values are independent of this feature and their default values are usually fine: **choices.closed** , **authority.required, authority.minconfidence** .

For the cache update script, one property can be set in `config/modules/solrauthority.cfg`:

```
auto-update-items = false | true
```

The default value for when the property is missing is false.

The final part of configuration is to add the authority consumer in front of the list of event consumers. Add "authority" in front of the list as displayed below.

```
event.dispatcher.default.consumers = authority, versioning, discovery, eperson, harvester
```

Without the consumer there is no automatic indexing of the authority cache and the metadata will not even have authority keys.

Changes to the configuration always require a server restart before they're in effect.

4.6.8.6 Adding additional fields under ORCID

Other metadata fields besides "dc.contributor.author" can benefit from the ORCID authority control at the same time. Here is an example of how to get the same ORCID functionality for the "dc.contributor.editor" metadata field assuming that "dc.contributor.author" is already configured correctly. It can be achieved by modifying configuration files only.

First add the same configuration fields that have been added for the "dc.contributor.author"

```
choices.plugin.dc.contributor.editor = SolrAuthorAuthority
choices.presentation.dc.contributor.editor = authorLookup
authority.controlled.dc.contributor.editor = true

authority.author.indexer.field.1=dc.contributor.author
authority.author.indexer.field.2=dc.contributor.editor
```

This is enough to get the look-up interface on the submission page and on the edit metadata page. The authority keys will be added and indexed with the information from orcid just as it happens with the Authors.

But you're not completely done yet, There is one more configuration step. Because now when adding new editors in the metadata that are not retrieved through the external look-up, their first and last name will not be displayed in the look-up interface next time you look for them.

To fix this, open the file at `config/spring/api/orcid-authority-services.xml` and find this spring bean:

```
<bean name="AuthorityTypes" class="org.dspace.authority.AuthorityTypes">
    <property name="types">
        <list>
            <bean class="org.dspace.authority.orcid.OrcidAuthorityValue"/>
            <bean class="org.dspace.authority.PersonAuthorityValue"/>
        </list>
    </property>
    <property name="fieldDefaults">
        <map>
            <entry key="dc_contributor_author">
                <bean class="org.dspace.authority.PersonAuthorityValue"/>
            </entry>
        </map>
    </property>
</bean>
```

The map inside the "fieldDefaults" property needs an additional entry for the editor field:

```
<entry key="dc_contributor_editor">
    <bean class="org.dspace.authority.PersonAuthorityValue"/>
</entry>
```

With this last change everything is set up to work correctly. The rest of this configuration file is meant for JAVA developers that wish to provide integration with other systems beside ORCID(see page 0). Developers that wish to display other fields than first and last name can also have a look in that section.

Note: Each metadata field has a separate set of authority records. Authority keys are not shared between different metadata fields. E. g. multiple dc.contributor.author can have the same authority key and point to the same authority record in the cache. But when an ORCID is chosen for a dc.contributor.editor field, a separate record is made in the cache. Both records are updated from the same source and will contain the same information. The difference is that when performing a look-up of a person that has been introduced as an authority for an author field but not yet as an editor, it will show as record that is not yet present in the repository cache.

4.6.8.7 Integration with other systems beside ORCID

The authority cache and look-up functionality can be extended to use other sources than ORCID or to show more information in the look-up interface. However some JAVA development is necessary for this. Specific instructions can be found in the readme file of the org.dspace.authority package[289].

4.6.8.8 FAQ

Which information from ORCID is currently indexed in the authority cache?

Here is a breakdown of the fields stored in the solr cache.

The system/dspace related fields are: *id, field, value, deleted, creation_date, last_modified_date, authority_type*.

The fields with data coming directly from ORCID are: *first_name, last_name, name_variant, orcid_id, label_researcher_url, label_keyword, label_external_identifier, label_biography, label_country.* The field *all_labels* contains all the values from the other fields starting with "label_".

How can I index additional fields in the authority cache?

There is currently no configuration to control which fields are indexed. The only way to achieve this is to modify the source code.

List of the files at work for this job:
`config/spring/api/orcid-authority-services.xml`: OrcidSource contains the URL for orcid's REST API.
org.dspace.authority.orcid.Orcid makes the REST call
+ org.dspace.authority.orcid.xml.XMLtoBio converts the received XML to a java object (Bio).
+ org.dspace.authority.orcid.model.Bio
+ org.dspace.authority.orcid.OrcidAuthorityValue#create(org.dspace.authority.orcid.model.Bio) inserts all the values from Bio into the AuthorityValue subclass.
+ org.dspace.authority.orcid.OrcidAuthorityValue#getSolrInputDocument defines what's included in solr.

The files preceded with a '+' would be necessary to modify to add more info into the cache.

How can I use the information stored in the authority cache?

The look-up UI is currently the only place this information is sent to. However just a limited number of fields are sent. The place in the source code to modify to get more fields there is org.dspace.authority.orcid.OrcidAuthorityValue#choiceSelectMap. This is also documented in the readme of the org.dspace.authority package.[290]

How to add additional metadata fields in the authority cache that are not related to ORCID?

Make the same configuration step as for adding additional fields under ORCID(see page 0). Currently the ORCID suggestions cannot be turned off for specific fields, that would require custom code.

289 https://github.com/DSpace/DSpace/tree/master/dspace-api/src/main/java/org/dspace/authority
290 https://github.com/DSpace/DSpace/tree/master/dspace-api/src/main/java/org/dspace/authority

What happens to data if another authority control was already present?

As long as the metadata does not get indexed, there will be no changes. However every time any metadata of an item is modified, the metadata under authority control for that item will be re-indexed. When that happens a record will be inserted in the solr cache. That record's ID will be the authority key of the metadata. This can be done for all metadata at once with the index-authority script.

In short: authority keys that exist prior to enabling the solrauthority are kept. They just won't show in the look-up until they are indexed.

Where can I find the URL that is used to lookup ORCIDs?

It is found in the `config/spring/api/orcid-authority-services.xml` configuration file. Look for the `<bean name="OrcidSource">`, which is initialized with a URL of http://pub.orcid.org

4.6.9 PDF Citation Cover Page

Enabling PDF Cover Pages may affect your site's visibility in Google Scholar (and similar search engines)

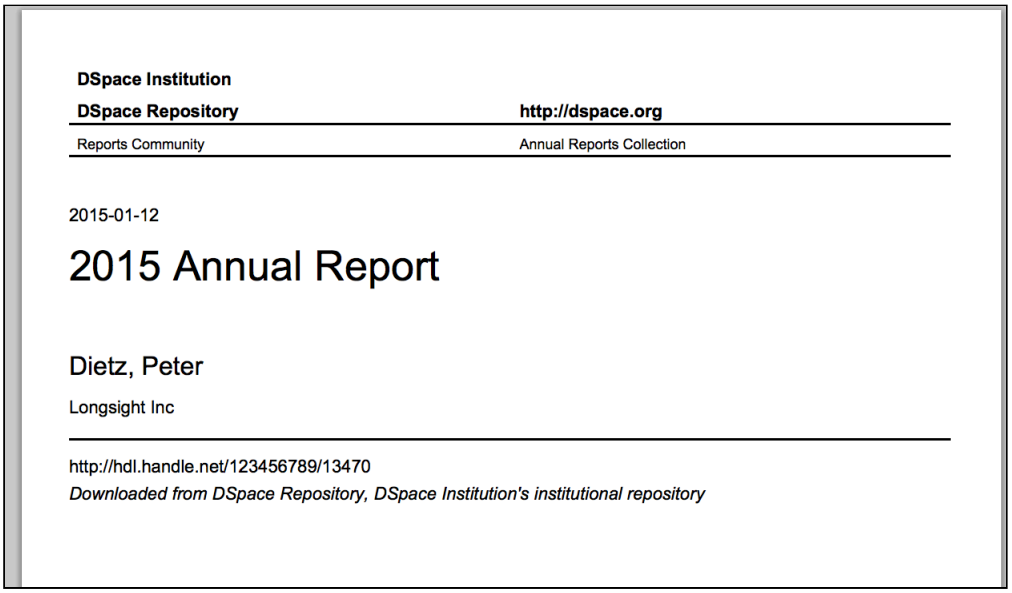
Google Scholar specifically warns against automatically generating PDF Cover Pages, as they can break the metadata extraction techniques used by their search engine. Be aware that enabling PDF Cover Pages may also cause those items to no longer be indexed by Google Scholar. For more information, please see the "Indexing Repositories: Pitfalls and Best Practices[291]" talk from Anurag Acharya (co-creator of Google Scholar) presented at the Open Repositories 2015 conference[292].

A known issue with the current implementation of the PDF Citation Cover Page is that primarily only English/Roman characters are supported. This is due to a limitation in the tool used to generate PDFs. See DS-2224[293] for more details on this issue

Adding a cover page to retrieved documents from DSpace that include additional citation information has been sought, as documents uploaded to the repository might have had their context stripped from them, when they are just a PDF. Context that might have surrounded the document would be the journal, publisher, edition, and more. Without that information, the document might just be a few pages of text, with no way to piece it together. Since repository policy might be to include this information as metadata to the Item, this metadata can be added to the citation cover page, so that the derivative PDF includes all of this information.

The citation cover page works by only storing the original PDF in DSpace, and then generating the citation-cover-page PDF on-the-fly. An alternative set up would be to run the PDF Citation Coverpage Curation Task on the DSpace repository contents, and then disseminate the pre-generated citation-version instead of generating it on the fly.

291 http://www.or2015.net/wp-content/uploads/2015/06/or-2015-anurag-google-scholar.pdf
292 http://www.or2015.net
293 https://jira.duraspace.org/browse/DS-2224



4.6.9.1 Configuration settings for Citation Cover Page

> **Configuration file renamed to citation-page.cfg and configurations names have changed**
>
> As of DSpace 6.0, the configuration file for this feature was **renamed** from `disseminate-citation.cfg` to `citation-page.cfg`. The renaming was to clarify the purpose of this configuration file, as its previous name was misleading / confusing to some users.
>
> In addition, all configurations below have now been prefixed with "citation-page" (e.g. the `enable_globally` configuration has been renamed to `citation-page.enable_globally`)

In the `{dspace.dir}/config/modules/citation-page.cfg` file review the following fields to make sure they are uncommented:

Property:	**citation-page.enable_globally**
Example Values:	citation-page.enable_globally = true
Informational Note:	Boolean to determine is citation-functionality is enabled globally for entire site. This will enable the citation cover page generator for all PDFs. Default: disabled
Property:	**citation-page.enabled_collections**

Example Values:	citation-page.enabled_collections = 1811/123, 1811/234
Informational Note:	List of collection handles to enable the cover page generator for bitstreams within. Default: blank
Property:	**citation-page.enabled_communities**
Example Values:	citation-page.enabled_communities = 1811/222, 1811/333
Informational Note:	List of community handles to enable the cover page generator for bitstreams within. Default: blank
Property:	**citation-page.citation_as_first_page**
Example Values:	citation-page.citation_as_first_page = true
Informational Note:	Should the citation page be the first page cover (true), or the last page (false). Default: true, (first page)
Property:	**citation-page.header1**
Example Values:	citation-page.header1 = University of Higher Education
Informational Note:	First row of header, perhaps for institution / university name. Commas separate multiple sections of the header (see screenshot above) Default Value: DSpace Institution
Property:	**citation-page.header2**
Example Values:	citation-page.header2 = Scholar Archive\, http://archive.example.com

Informational Note:	Second row of header, perhaps put your DSpace instance branded name, and url to your DSpace. A comma is used to separate instance name, and the URL Default Value: DSpace Repository, http://dspace.org
Property:	**citation-page.fields**
Example Values:	citation-page.fields = dc.date.issued, dc.title, dc.creator, dc.contributor.author, dc.publisher, _line_, dc.identifier.citation, dc.identifier.uri
Informational Note:	Metadata fields to display on the citation PDF. Specify in schema.element.qualifier form, and separate fields by a comma. If you want to have a horizontal line break, use _line_ Default Value: dc.date.issued,dc.title,dc.creator,dc.contributor.author,dc.publisher, _line_,dc.identifier.citation,dc.identifier.uri
Property:	**citation-page.footer**
Example Values:	citation-page.footer = Downloaded from Scholar Archive at University of Higher Education\, an open access institutional repository. All Rights Reserved.
Informational Note:	Footer text at the bottom of the citation page. It might be some type of license or copyright information, or just letting the recipient know where they downloaded the file from. Default Value: Downloaded from DSpace Repository\, DSpace Institution's institutional repository NOTE: any commas appearing in this text should be escaped as "\,". See example above.

4.6.10 Updating Items via Simple Archive Format

4.6.10.1 Item Update Tool

ItemUpdate is a batch-mode command-line tool for altering the metadata and bitstream content of existing items in a DSpace instance. It is a companion tool to ItemImport and uses the DSpace simple archive format to specify changes in metadata and bitstream contents. Those familiar with generating the source trees for ItemImport will find a similar environment in the use of this batch processing tool.

For metadata, ItemUpdate can perform 'add' and 'delete' actions on specified metadata elements. For bitstreams, 'add' and 'delete' are similarly available. All these actions can be combined in a single batch run.

ItemUpdate supports an undo feature for all actions except bitstream deletion. There is also a test mode, as with ItemImport. However, unlike ItemImport, there is no resume feature for incomplete processing. There is more extensive logging with a summary statement at the end with counts of successful and unsuccessful items processed.

One probable scenario for using this tool is where there is an external primary data source for which the DSpace instance is a secondary or down-stream system. Metadata and/or bitstream content changes in the primary system can be exported to the simple archive format to be used by ItemUpdate to synchronize the changes.

A note on terminology: **item** refers to a DSpace item. **metadata element** refers generally to a qualified or unqualified element in a schema in the form `[schema].[element].[qualifier]` or `[schema].[element]` and occasionally in a more specific way to the second part of that form. **metadata field** refers to a specific instance pairing a metadata element to a value.

DSpace Simple Archive Format

As with ItemImporter(see page 233), the idea behind the DSpace's simple archive format is to create an archive directory with a subdirectory per item. There are a few additional features added to this format specifically for ItemUpdate. Note that in the simple archive format, the item directories are merely local references and only used by ItemUpdate in the log output.

The user is referred to the previous section DSpace Simple Archive Format(see page 233).

Additionally, the use of a **delete_contents** is now available. This file lists the bitstreams to be deleted, one bitstream ID per line. Currently, no other identifiers for bitstreams are usable for this function. This file is an addition to the Archive format specifically for ItemUpdate.

The optional suppress_undo file is a flag to indicate that the 'undo archive' should not be written to disk. This file is usually written by the application in an undo archive to prevent a recursive undo. This file is an addition to the Archive format specifically for ItemUpdate.

ItemUpdate Commands

Command used:	`[dspace]/bin/dspace itemupdate`
Java class:	*org.dspace.app.itemupdate.ItemUpdate*
Arguments short and (long) forms:	Description

`-a` or `--addmetadata [metadata element]`	Repeatable for multiple elements. The metadata element should be in the form dc.x or dc.x.y. The mandatory argument indicates the metadata fields in the dublin_core.xml file to be added unless already present (multiple fields should be separated by a semicolon ';'). However, duplicate fields will not be added to the item metadata without warning or error.
`-d` or `--deletemetadata [metadata element]`	Repeatable for multiple elements. All metadata fields matching the element will be deleted.
`-A` or `--addbitstreams`	Adds bitstreams listed in the contents file with the bitstream metadata cited there.
`-D` or `--deletebitstreams [filter plug classname or alias]`	Not repeatable. With no argument, this operation deletes bitstreams listed in the `deletes_contents` file. Only bitstream IDs are recognized identifiers for this operation. The optional filter argument is the classname of an implementation of `org.dspace.app.itemdupate.BitstreamFilter` class to identify files for deletion or one of the aliases (e.g. ORIGINAL, ORIGINAL_AND_DERIVATIVES, TEXT, THUMBNAIL) which reference existing filters based on membership in a bundle of that name. In this case, the `delete_contents` file is not required for any item. The filter properties file will contains properties pertinent to the particular filer used. Multiple filters are not allowed.
`-h` or `--help`	Displays brief command line help.
`-e` or `--eperson`	Email address of the person or the user's database ID **(Required)**
`-s` or `--source`	Directory archive to process **(Required)**
`-i` or `--itemfield`	Specifies the metadata field that contains the item's identifier; Default value is "dc.identifier.uri" (Optional)
`-t` or `--test`	Runs the process in test mode with logging. But no changes applied to the DSpace instance. (Optional)

`-P` or `--provenance`	Prevents any changes to the provenance field to represent changes in the bitstream content resulting from an Add or Delete. In other words, when this flag is specified, no new provenance information is added to the DSpace Item when adding/deleting a bitstream. No provenance statements are written for thumbnails or text derivative bitstreams, in keeping with the practice of MediaFilterManager. (Optional)
`-F` or `--filter-properties`	The filter properties files to be used by the delete bitstreams action (Optional)
`-v` or `--verbose`	Turn on verbose logging.

CLI Examples

Adding Metadata:

```
[dspace]/bin/dspace itemupdate -e joe@user.com -s [path/to/archive] -a dc.description
```

This will update all DSpace Items listed in your archive directory, adding a new `dc.description` metadata field. Items will be located in DSpace based on the handle found in 'dc.identifier.uri' (since the `-i` argument wasn't used, the default metadata field, dc.identifier.uri, from the dublin_core.xml file in the archive folder, is used).

4.7 Managing Community Hierarchy

4.7.1 Sub-Community Management

DSpace provides an administrative tool, 'CommunityFiliator', for managing community sub-structure. It has two operations, either establishing a community to sub-community relationship, or dis-establishing an existing relationship.

The familiar parent/child metaphor can be used to explain how it works. Every community in DSpace can be either a 'parent' community, meaning it has at least one sub-community, or a 'child' community, meaning it is a sub-community of another community, or both or neither. In these terms, an 'orphan' is a community that lacks a parent (although it can be a parent); 'orphans' are referred to as 'top-level' communities in the DSpace user-interface, since there is no parent community 'above' them. The first operation, establishing a parent/child relationship - can take place between any community and an orphan. The second operation - removing a parent/child relationship, will make the child an orphan.

Command used:	`[dspace]/bin/dspace community-filiator`

Java class:	*org.dspace.administer.CommunityFiliator*
Arguments short and (long) forms:	Description
`-s` or `--set`	Set a parent/child relationship
`-r` or `--remove`	Remove a parent/child relationship
`-c` or `--child`	Child community (Handle or database ID)
`-p` or `--parent`	Parent community (Handle or database ID
`-h` or `--help`	Online help.

Set a parent/child relationship, issue the following at the CLI:

```
[dspace]/bin/dspace community-filiator --set --parent=parentID --child=childID
```

(or using the short form)

```
[dspace]/bin/dspace community-filiator -s -p parentID -c childID
```

where '-s' or '-set' means establish a relationship whereby the community identified by the '-p' parameter becomes the parent of the community identified by the '-c' parameter. Both the 'parentID' and 'childID' values may be handles or database IDs.

The reverse operation looks like this:

```
[dspace]/bin/dspace community-filiator --remove --parent=parentID --child=childID
```

(or using the short form)

```
[dspace]/bin/dspace community-filiator -r -p parentID -c childID
```

where '-r' or '-remove' means dis-establish the current relationship in which the community identified by 'parentID' is the parent of the community identified by 'childID'. The outcome will be that the 'childID' community will become an orphan, i.e. a top-level community.

If the required constraints of operation are violated, an error message will appear explaining the problem, and no change will be made. An example in a removal operation, where the stated child community does not have the stated parent community as its parent: "Error, child community not a child of parent community".

It is possible to effect arbitrary changes to the community hierarchy by chaining the basic operations together. For example, to move a child community from one parent to another, simply perform a 'remove' from its current parent (which will leave it an orphan), followed by a 'set' to its new parent.

It is important to understand that when any operation is performed, all the sub-structure of the child community follows it. Thus, if a child has itself children (sub-communities), or collections, they will all move with it to its new 'location' in the community tree.

4.8 Statistics and Metrics

4.8.1 SOLR Statistics

DSpace uses the Apache SOLR application underlying the statistics. SOLR enables performant searching and adding to vast amounts of (usage) data.
Unlike previous versions, enabling statistics in DSpace does not require additional installation or customization. All the necessary software is included.

4.8.1.1 What is exactly being logged ?

After the introduction of the SOLR Statistics logging, every pageview and file download is logged in a dedicated SOLR statistics core.

In addition to the already existing logging of pageviews and downloads, DSpace also logs search queries users enter in the DSpace search dialog and workflow events.

DSpace 7.0 does not yet support all features

In DSpace 7.0, only usage statistics (pageview, downloads) are logged. Search statistics and workflow reports (which were available in v6) are not yet supported, but are both scheduled to be restored in a later 7.x release (currently 7.1 for workflow reports, and 7.2 for search statistics), see DSpace Release 7.0 Status[294]

Workflow Events logging

Only workflow events, initiated and executed by a physical user are being logged. Automated workflow steps or ingest procedures are currently **not** being logged by the workflow events logger.

The logging happens at the server side, and doesn't require a javascript like Google Analytics does, to provide usage data. Definition of which fields are to be stored happens in the file **dspace/solr/statistics/conf/schema.xml**.

Although they are stored in the same index, the stored fields for views, search queries and workflow events are different. A new field, statistics_type determines which kind of a usage event you are dealing with. The three possible values for this field are **view, search** and **workflow.**

```
<field name="statistics_type" type="string" indexed="true" stored="true" required="true" />
```

294 https://wiki.lyrasis.org/display/DSPACE/DSpace+Release+7.0+Status

Common stored fields for all usage events

```
<field name="type" type="integer" indexed="true" stored="true" required="true" />
<field name="id" type="integer" indexed="true" stored="true" required="true" />
<field name="ip" type="string" indexed="true" stored="true" required="false" />
<field name="time" type="date" indexed="true" stored="true" required="true" />
<field name="epersonid" type="integer" indexed="true" stored="true" required="false" />
<field name="continent" type="string" indexed="true" stored="true" required="false"/>
<field name="country" type="string" indexed="true" stored="true" required="false"/>
<field name="countryCode" type="string" indexed="true" stored="true" required="false"/>
<field name="city" type="string" indexed="true" stored="true" required="false"/>
<field name="longitude" type="float" indexed="true" stored="true" required="false"/>
<field name="latitude" type="float" indexed="true" stored="true" required="false"/>
<field name="owningComm" type="integer" indexed="true" stored="true" required="false" multiValued="true"/>
<field name="owningColl" type="integer" indexed="true" stored="true" required="false" multiValued="true"/>
<field name="owningItem" type="integer" indexed="true" stored="true" required="false" multiValued="true"/>
<field name="dns" type="string" indexed="true" stored="true" required="false"/>
<field name="userAgent" type="string" indexed="true" stored="true" required="false"/>
<field name="isBot" type="boolean" indexed="true" stored="true" required="false"/>
<field name="referrer" type="string" indexed="true" stored="true" required="false"/>
<field name="uid" type="uuid" indexed="true" stored="true" default="NEW" />
<field name="statistics_type" type="string" indexed="true" stored="true" required="true" default="view" />
```

The combination of type(see page 656) and id determines which resource (either community, collection, item page or file download) has been requested.

Unique stored fields for bitstream downloads

```
<field name="bundleName" type="string" indexed="true" stored="true" required="false" multiValued="true" />
```

Unique stored fields for search queries

```
<field name="query" type="string" indexed="true" stored="true" required="false" multiValued="true"/>
<field name="scopeType" type="integer" indexed="true" stored="true" required="false" />
<field name="scopeId" type="integer" indexed="true" stored="true" required="false" />
<field name="rpp" type="integer" indexed="true" stored="true" required="false" />
<field name="sortBy" type="string" indexed="true" stored="true" required="false" />
<field name="sortOrder" type="string" indexed="true" stored="true" required="false" />
<field name="page" type="integer" indexed="true" stored="true" required="false" />
```

Unique stored fields for workflow events

```
<field name="workflowStep" type="string" indexed="true" stored="true" required="false" multiValued="true"/>
<field name="previousWorkflowStep" type="string" indexed="true" stored="true" required="false"
multiValued="true"/>
<field name="owner" type="string" indexed="true" stored="true" required="false" multiValued="true"/>
<field name="submitter" type="integer" indexed="true" stored="true" required="false" />
<field name="actor" type="integer" indexed="true" stored="true" required="false" />
<field name="workflowItemId" type="integer" indexed="true" stored="true" required="false" />
```

4.8.1.2 Web User Interface Elements

Pageview and Download statistics

In the UI, pageview and download statistics can be accessed from the "Statistics" navigation menu near the header. That statistics page is "context aware", so it will show the usage statistics for whatever page (site, Community, Collection) you are currently on.

If you are not seeing the menu, it's likely that they are only enabled for administrators in your installation. Change the configuration parameter "authorization.admin.usage" in usage-statistics.cfg to false in order to make statistics visible for all repository visitors.

Home page

Starting from the repository homepage, the statistics page displays the top 10 most popular items of the entire repository.

Community home page

The following statistics are available for the community home pages:

- Total visits of the current community home page
- Visits of the community home page over a timespan of the last 7 months
- Top 10 country from where the visits originate
- Top 10 cities from where the visits originate

Collection home page

The following statistics are available for the collection home pages:

- Total visits of the current collection home page
- Visits of the collection home over a timespan of the last 7 months
- Top 10 country from where the visits originate
- Top 10 cities from where the visits originate

Item home page

The following statistics are available for the item home pages:

- Total visits of the item
- Total visits for the bitstreams attached to the item

- Visits of the item over a timespan of the last 7 months
- Top 10 country views from where the visits originate
- Top 10 cities from where the visits originate

Search Query Statistics

DSpace 7.0 does not yet support

Search query statistics are not supported in 7.0, but are scheduled to be released in a later 7.x release (currently 7.2), see DSpace Release 7.0 Status[295].

The below screenshots and instructions are for 6.x and will need updating for 7.x once this feature is completed.

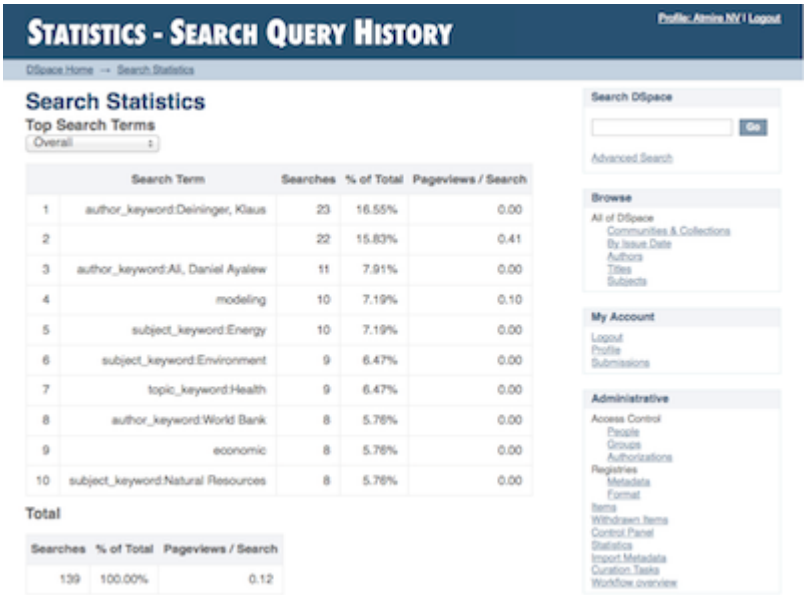
In the UI, search query statistics can be accessed from the lower end of the navigation menu.

If you are not seeing the link labelled "search statistics", it is likely that they are only enabled for administrators in your installation. Change the configuration parameter "authorization.admin.search" in usage-statistics.cfg to false in order to make statistics visible for all repository visitors.

The dropdown on top of the page allows you to modify the time frame for the displayed statistics.

The Pageviews/Search column tracks the amount of pages visited after a particular search term. Therefor a zero in this column means that after executing a search for a specific keyword, not a single user has clicked a single result in the list.

If you are using Discovery, note that clicking the facets[296] also counts as a search, because clicking a facet[297] sends a search query to the Discovery index.



295 https://wiki.lyrasis.org/display/DSPACE/DSpace+Release+7.0+Status
296 https://wiki.duraspace.org/display/DSDOC3x/Discovery#Discovery-WhatisaSidebarFacet
297 https://wiki.duraspace.org/display/DSDOC3x/Discovery#Discovery-WhatisaSidebarFacet

Workflow Event Statistics

DSpace 7.0 does not yet support

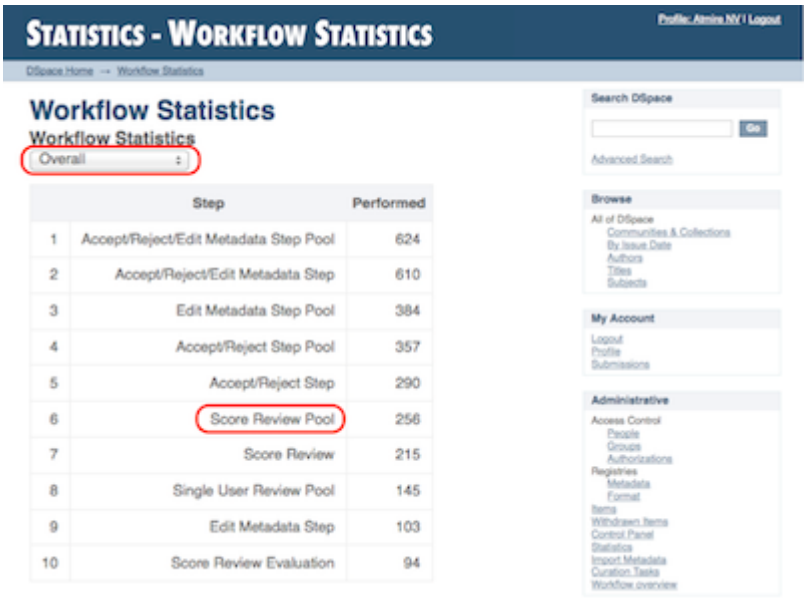
Workflow event statistics are not supported in 7.0, but are scheduled to be released in a later 7.x release (currently 7.1), see DSpace Release 7.0 Status[298].

The below screenshots and instructions are for 6.x and will need updating for 7.x once this feature is completed.

In the UI, search query statistics can be accessed from the lower end of the navigation menu.

If you are not seeing the link labelled "Workflow statistics", it is likely that they are only enabled for administrators in your installation. Change the configuration parameter "authorization.admin.workflow" in usage-statistics.cfg to false in order to make statistics visible for all repository visitors.

The dropdown on top of the page allows you to modify the time frame for the displayed statistics.



4.8.1.3 Architecture

The DSpace Statistics Implementation is a Client/Server architecture based on Solr for collecting usage events in the User Interface or REST API applications of DSpace. Solr must be installed separately from DSpace.

4.8.1.4 Configuration settings for Statistics

In the `{dspace.dir}/config/modules/solr-statistics.cfg` file review the following fields. These fields can be edited in place, or overridden in your own local.cfg config file (see Configuration Reference(see page 552)).

Property:	solr-statistics.server

298 https://wiki.lyrasis.org/display/DSPACE/DSpace+Release+7.0+Status

Example Values:	solr-statistics.server = http://127.0.0.1/solr/statistics solr-statistics.server = ${solr.server}/statistics
Informational Note:	Is used by the SolrLogger Client class to connect to the Solr server over http and perform updates and queries. In most cases, this can (and should) be set to localhost (or 127.0.0.1). To determine the correct path, you can use a tool like `wget` to see where Solr is responding on your server. For example, you'd want to send a query to Solr like the following: `wget http://127.0.0.1/solr/statistics/select?q=*:*` Assuming you get an HTTP 200 OK response, then you should set `solr.log.server` to the '/statistics' URL of 'http://127.0.0.1/solr/statistics' (essentially removing the "/select?q=**:**" query off the end of the responding URL.)
Property:	solr-statistics.query.filter.bundles
Example Value:	solr-statistics.query.filter.bundles=ORIGINAL
Informational Note:	A comma seperated list that contains the bundles for which the file statistics will be displayed.
Property:	solr-statistics.query.filter.spiderIp
Example Value:	solr-statistics.query.filter.spiderIp = false
Informational Note:	If true, statistics queries will filter out spider IPs -- use with caution, as this often results in extremely long query strings.
Property:	solr-statistics.query.filter.isBot

Example Value:	solr-statistics.query.filter.isBot = true
Informational Note:	If true, statistics queries will filter out events flagged with the "isBot" field. This is the recommended method of filtering spiders from statistics.
Property:	solr-statistics.autoCommit
Example Value:	solr-statistics.autoCommit = true
Informational Note:	If true (default), then all view statistics will be committed to Solr whenever the next autoCommit is triggered. This is recommended behavior. If false, then view statistics will be committed to Solr *immediately* (i.e. via an explicit commit call). This setting is untested in Production scenarios, and is primarily used by automated integration tests (to verify that the statistics engine is working properly).
Property:	solr-statistics.spiderips.urls
Example Value:	solr-statistics.spiderips.urls = `http://iplists.com/google.txt, \` `http://iplists.com/inktomi.txt, \` `http://iplists.com/lycos.txt, \` `http://iplists.com/infoseek.txt, \` `http://iplists.com/altavista.txt, \` `http://iplists.com/excite.txt, \` `http://iplists.com/misc.txt`

Informational Note:	List of URLs to download spiders files into [dspace]/config/spiders. These files contain lists of known spider IPs and are utilized by the SolrLogger to flag usage events with an "isBot" field, or ignore them entirely. The "stats-util" command can be used to force an update of spider files, regenerate "isBot" fields on indexed events, and delete spiders from the index. For usage, run: `dspace stats-util -h` from your [dspace]/bin directory

In the `{dspace.dir}/config/modules/`**`usage-statistics`**`.cfg` file review the following fields. These fields can be edited in place, or overridden in your own local.cfg config file (see Configuration Reference(see page 552)).

Property:	usage-statistics.dbfile
Example Value:	usage-statistics.dbfile = ${dspace.dir}/config/GeoLite2-City.mmdb
Informational Note:	References the location of the installed GeoLite or DB-IP City "mmdb" database file. This file is utilized by the LocationUtils to calculate the location of client requests based on IP address. NOTE: This database file MUST be downloaded, installed and updated using third-party tools. See the "Managing the City Database File(see page 350)" section below.
Property:	usage-statistics.resolver.timeout
Example Value:	usage-statistics.resolver.timeout = 200
Informational Note:	Timeout in milliseconds for DNS resolution of origin hosts/IPs. Setting this value too high may result in solr exhausting your connection pool.

Property:	useProxies (Set in dspace.cfg)
Example Value:	useProxies = true
Informational Note:	Will cause Statistics logging to look for X-Forward URI to detect clients IP that have accessed it through a Proxy service (e.g. the Apache mod_proxy). Allows detection of client IP when accessing DSpace. [Note: This setting is found in the DSpace Logging section of dspace.cfg]
Property:	usage-statistics.authorization.admin.usage
Example Value:	usage-statistics.authorization.admin.usage = true
Informational Note:	When set to true, only general administrators, collection and community administrators are able to access the pageview and download statistics from the web user interface. As a result, the links to access statistics are hidden for non logged-in admin users. Setting this property to "false" will display the links to access statistics to anyone, making them publicly available.
Property:	usage-statistics.authorization.admin.search
Example Value:	usage-statistics.authorization.admin.search = true
Informational Note:	When set to true, only system, collection or community administrators are able to access statistics on search queries.
Property:	usage-statistics.authorization.admin.workflow
Example Value:	usage-statistics.authorization.admin.workflow = true
Informational Note:	When set to true, only system, collection or community administrators are able to access statistics on workflow events.

Property:	usage-statistics.logBots
Example Value:	usage-statistics.logBots = true
Informational Note:	When this property is set to false, and IP is detected as a spider, the event is not logged. When this property is set to true, the event will be logged with the "isBot" field set to true. (see solr-statistics.query.filter.* for query filter options)
Property:	usage-statistics.shardedByYear
Example Value:	usage-statistics.shardedByYear = false
Informational Note:	When set to "true", the DSpace statistics engine will look for additional Solr Shards (per year) when compiling all usage statistics. Therefore, if you are regularly running "stats-utils -s" (as documented in the "Solr Sharding By Year"[299]section of the "SOLR Statistics Maintenance" page), then you should set this to "true". By default, it is "false", which tells the statistics engine to only compile usage statistics based on what is found in the current Solr core.

Pre-1.6 Statistics settings

DSpace 7.0 does not yet support

Log-based statistics not supported in 7.0. They are under discussion as this feature is not widely used. Tentatively they are scheduled for a possible release/replacement in 7.1, see DSpace Release 7.0 Status[300].

Older versions of DSpace featured static reports generated from the log files. They still persist in DSpace today but are completely independent from the SOLR based statistics.
The following configuration parameters applicable to these reports can be found in dspace.cfg.

```
###### Statistical Report Configuration Settings ######

 # should the stats be publicly available?  should be set to false if you only
 # want administrators to access the stats, or you do not intend to generate
 # any
 report.public = false

 # directory where live reports are stored
 report.dir = ${dspace.dir}/reports/
```

299 https://wiki.lyrasis.org/display/DSDOC6x/SOLR+Statistics+Maintenance#SOLRStatisticsMaintenance-SolrShardingByYear
300 https://wiki.lyrasis.org/display/DSPACE/DSpace+Release+7.0+Status

These fields are not used by the new 1.6 Statistics, but are only related to the Statistics from previous DSpace releases

4.8.1.5 Statistics Administration

Converting older DSpace logs into SOLR usage data

If you have upgraded from a previous version of DSpace, converting older log files ensures that you carry over older usage stats from before the upgrade.

Statistics Client Utility

The command line interface (CLI) scripts can be used to clean the usage database from additional spider traffic and other maintenance tasks. As of DSpace 3.0, a script has been added to split up the monolithic SOLR core into individual cores each containing a year of statistics.

4.8.1.6 Custom Reporting - Querying SOLR Directly

When the web user interface does not offer you the statistics you need, you can greatly expand the reports by querying the SOLR index directly.

Resources

- https://www.safaribooksonline.com/library/view/apache-solr-enterprise/9781782161363/
- https://lucidworks.com/blog/faceted-search-with-solr/

Examples

Top downloaded items by a specific user

Query:

```
http://localhost:8983/solr/statistics/select?
indent=on&version=2.2&start=0&rows=10&fl=*%2Cscore&qt=standard&wt=standard&explainOther=&hl.fl=&facet=true&
facet.field=epersonid&q=type:0
```

Explained:

facet.field=epersonid — You want to group by epersonid, which is the user id.
type:0 — Interested in bitstreams only

```
<lst name="facet_counts">
    <lst name="facet_fields">
        <lst name="epersonid">
            <int name="66">1167</int>

<int name="117">251</int>

<int name="52">42</int>

<int name="19">36</int>

<int name="88">20</int>

<int name="112">18</int>

<int name="110">9</int>

<int name="96">0</int>

</lst>
    </lst>
</lst>
```

4.8.1.7 Managing the City Database File

If you wish to record the geographic locations of clients in your DSpace statistics records (e.g. the City or Country where they are accessing your DSpace), you **must** install (and regularly update) one of the following IP to City Database Files (in MMDB format). We recommend installing a City-level database, as it provides more granular location information than a Country-level database (which can only provide the country where the access originated).

- Either install a copy of MaxMind's GeoLite City database[301] (in MMDB format)
  - Installing MaxMind GeoLite2 is *free*. However, you **must** sign up for a (free) MaxMind account in order to obtain a license key to use the GeoLite2 database.
  - You will need to arrange regular downloads of the GeoLite2 database. MaxMind offers an updater tool (geoipupdate)[302] to do the downloading/updating, and a number of Linux distributions package it (as `geoipupdate`). You will still need to configure your license key prior to usage. Use it before restarting DSpace, to get an up-to-date database.
  - Once the "GeoLite2-City.mmdb" database file is installed on your system, you will need to configure its location as the value of `usage-statistics.dbfile` in your `local.cfg` configuration file.
  - NOTE: This file is frequently updated by MaxMind.com[303], so you will need to refresh it regularly (ideally by scheduling the updater tool via a cron job or similar). As this is written, the database is updated monthly, and to be allowed to obtain it you need to agree to keep your copy updated.
- Or, you can alternatively use/install DB-IP's City Lite database[304] (in MMDB format)
  - This database is also free to use, but does **not** require an account to download.
  - You will need to arrange regular downloads of the City Lite database. DB-IP offers an updater tool (dbip-update)[305] to do the downloading/updating, but it requires PHP to run.

301 https://dev.maxmind.com/geoip/geoip2/geolite2/
302 https://dev.maxmind.com/geoip/geoipupdate/#For_Free_GeoLite2_Databases
303 http://MaxMind.com
304 https://db-ip.com/db/download/ip-to-city-lite
305 https://db-ip.com/tutorials/database-file-update

- Once the "dbip-city-lite.mmdb" database file is installed on your system, you will need to configure its location as the value of `usage-statistics.dbfile` in your `local.cfg` configuration file.
- NOTE: This file is frequently updated by DB-IP.com[306], so you will need to refresh it regularly (ideally by scheduling the updater tool via a cron job or similar). As this is written, the database is updated monthly with the latest available at https://db-ip.com/db/download/ip-to-city-lite

4.8.1.8 SOLR Statistics Maintenance

DSpace Log Converter

The use of Solr for statistics in DSpace makes it possible to have a database of statistics. With this in mind, there is the issue of the older log files and how a site can use them. The following command process is able to convert the existing log files and then import them for Solr use. The user will need to perform this conversion only once.

The Log Converter program converts log files from dspace.log into an intermediate format that can be inserted into Solr.

Command used:	`[dspace]/bin/dspace stats-log-converter`
Java class:	org.dspace.statistics.util.ClassicDSpaceLogConverter
Arguments short and long forms):	Description
`-i` or `--in`	Input file
`-o` or `--out`	Output file
`-m` or `--multiple`	Adds a wildcard at the end of input and output, so it would mean if `-i dspace.log -m` was specified, dspace.log* would be converted. (i.e. all of the following: *dspace.log, dspace.log.1, dspace.log.2, dspace.log.3,* etc.)

306 https://db-ip.com/

`-n` or `--newformat`	If the log files have been created with DSpace 1.6 or newer
`-v` or `--verbose`	Display verbose output (helpful for debugging)
`-h` or `--help`	Help

The command loads the intermediate log files that have been created by the aforementioned script into Solr.

Command used:	`[dspace]/bin/dspace stats-log-importer`
Java class:	org.dspace.statistics.util.StatisticsImporter
Arguments (short and long forms):	Description
`-i` or `--in`	input file
`-m` or `--multiple`	Adds a wildcard at the end of the input, so it would mean dspace.log* would be imported
`-s` or `--skipdns`	To skip the reverse DNS lookups that work out where a user is from. (The DNS lookup finds the information about the host from its IP address, such as geographical location, etc. This can be slow, and wouldn't work on a server not connected to the internet.)
`-v` or `--verbose`	Display verbose ouput (helpful for debugging)
`-l` or `--local`	For developers: allows you to import a log file from another system, so because the handles won't exist, it looks up random items in your local system to add hits to instead.
`-h` or `--help`	Help

Although the DSpace Log Convertor applies basic spider filtering (googlebot, yahoo slurp, msnbot), it is far from complete. Please refer to Filtering and Pruning Spiders(see page 352) for spider removal operations, after converting your old logs.

Filtering and Pruning Spiders

Command used:	`[dspace]/bin/dspace stats-util`

Java class:	org.dspace.statistics.util.StatisticsClient
Arguments (short and long forms):	Description
`-b` or `--reindex-bitstreams`	Reindex the bitstreams to ensure we have the bundle name
`-r` or `--remove-deleted-bitstreams`	While indexing the bundle names remove the statistics about deleted bitstreams
`-u` or `--update-spider-files`	Update Spider IP Files from internet into `[dspace]/config/spiders`. Downloads Spider files identified in `dspace.cfg` under property `solr.spiderips.urls`. See Configuration settings for Statistics[307]
`-f` or `--delete-spiders-by-flag`	Delete Spiders in Solr By isBot Flag. Will prune out all records that have `isBot:true`
`-i` or `--delete-spiders-by-ip`	Delete Spiders in Solr By IP Address, DNS name, or Agent name. Will prune out all records that match spider identification patterns.
`-m` or `--mark-spiders`	Update isBot Flag in Solr. Marks any records currently stored in statistics that have IP addresses matched in spiders files
`-h` or `--help`	Calls up this brief help table at command line.

Notes:

The usage of these options is open for the user to choose. If you want to keep spider entries in your repository, you can just mark them using "`-m`" and they will be excluded from statistics queries when "`solr.statistics.query.filter.isBot = true`" in the `dspace.cfg`. If you want to keep the spiders out of the solr repository, just use the "`-i`" option and they will be removed immediately.

Spider IPs are specified in files containing one pattern per line. A line may be a comment (starting with "#" in column 1), empty, or a single IP address or DNS name. If a name is given, it will be resolved to an address. Unresolvable names are discarded and will be noted in the log.

There are guards in place to control what can be defined as an IP range for a bot. In `[dspace]/config/spiders`, spider IP address ranges have to be at least 3 subnet sections in length 123.123.123 and IP Ranges can only be on the smallest subnet [123.123.123.0 - 123.123.123.255]. If not, loading that row will cause exceptions in the dspace logs and exclude that IP entry.

307 https://wiki.duraspace.org/display/DSDOC3x/DSpace+Statistics#DSpaceStatistics-ConfigurationsettingsforStatistics

Spiders may also be excluded by DNS name or Agent header value. Place one or more files of patterns in the directories `[dspace]/config/spiders/domains` and/or `[dspace]/config/spiders/agents`. Each line in a pattern file should be either empty, a comment starting with "#" in column 1, or a regular expression which matches some names to be recognized as spiders.

Export SOLR records to intermediate format for import into another tool/instance

Command used:	`[dspace]/bin/dspace stats-util`
Java class:	org.dspace.statistics.util.StatisticsClient
Arguments (short and long forms):	Description
`-e` or `--export`	Export SOLR view statistics data to usage statistics intermediate format

This exports the records to `[dspace]/temp/usagestats_0.csv`. This will chunk the files at 10,000 records to new files. This can be imported with `stats-log-importer` to SOLR Statistics(see page 338)

Export SOLR statistics, for backup and moving to another server

Command used:	`[dspace]/bin/dspace solr-export-statistics`
Java class:	org.dspace.util.SolrImportExport
Arguments (short and long forms):	Description
- i or - -index-name	optional, the name of the index to process. "statistics" is the default
`-l or --last` *integer*	optionally export only *integer* many days worth of statistics
`-d or --directory`	optional, directory to use for storing the exported files. By default, `[dspace]/solr-export` is used. If that is not appropriate (due to storage concerns), we recommend you use this option to specify a more appropriate location.
- f or - -force-overwrite	optional, overwrite export file if it exists (DSpace 6.1 and later)

Import SOLR statistics, for restoring lost data or moving to another server

Command used:	`[dspace]/bin/dspace solr-import-statistics`
Java class:	org.dspace.util.SolrImportExport
Arguments (short and long forms):	Description
- i or - -index-name	optional, the name of the index to process. "statistics" is the default
`-c or --clear`	optional, clears the contents of the existing stats core before importing
`-d or --directory`	optional, directory which contains the files for importing. By default, `[dspace]/solr-export` is used. If that is not appropriate (due to storage concerns), we recommend you use this option to specify a more appropriate location.

Reindex SOLR statistics, for upgrades or whenever the Solr schema for statistics is changed

Command used:	`[dspace]/bin/dspace solr-reindex-statistics`
Java class:	org.dspace.util.SolrImportExport
Arguments (short and long forms):	Description
- i or - -index-name	optional, the name of the index to process. "statistics" is the default
`-k or --keep`	optional, tells the script to keep the intermediate export files for possible later use (by default all exported files are removed at the end of the reindex process).
`-d or --directory`	optional, directory to use for storing the exported files (temporarily, unless you also specify `--keep`, see above). By default, `[dspace]/solr-export` is used. If that is not appropriate (due to storage concerns), we recommend you use this option to specify a more appropriate location. Not sure about your space requirements? You can estimate the space required by looking at the current size of `[dspace]/solr/statistics`

- f or - -force-overwrite	optional, overwrite export file if it exists (DSpace 6.1 and later)

NOTE: `solr-reindex-statistics` is safe to run on a live site. The script stores incoming usage data in a temporary SOLR core, and then merges that new data into the reindexed data when the reindex process completes.

Upgrade Legacy DSpace Object Identifiers (pre-6x statistics) to DSpace 6x UUID Identifiers

This feature has not yet been released.

This command will be introduced in the **DSpace 6.4** and **DSpace 7.0** releases.

It is recommended that all DSpace instances with legacy identifiers perform this one-time upgrade of legacy statistics records.

This action is safe to run on a live site. As a precaution, it is recommended that you backup you statistics shards before performing this action.

Note: a link to this section of the documentation should be added to the DSpace 6.4 and DSpace 7.0 Release Notes.

The DSpace 6x code base changed the primary key for all DSpace objects from an integer id to UUID identifiers. Statistics records that were created before upgrading to DSpace 6x contain the legacy identifiers.

While the DSpace user interfaces make some attempt to correlate legacy identifiers with uuid identifiers, it is recommended that users perform this one time upgrade of legacy statistics records.

If you have sharded your statistics repository, this action must be performed on each shard.

Command used:	`[dspace]/bin/dspace solr-upgrade-statistics-6x`
Java class:	org.dspace.util.SolrUpgradePre6xStatistics
Arguments (short and long forms):	Description
- i or - -index-name	Optional, the name of the index to process. "statistics" is the default
`-n or --num_rec`	Optional. Total number of records to update (defaut=100,000). To process all records, set -n to 10000000 or to 100000000 (10M or 100M) If possible, please allocate 2GB of memory to this process (e.g. -Xmx2000m)
`-b or --batch_size`	Number of records to batch update to SOLR at one time (default=10,000).

NOTE: This process will rewrite most solr statistics records and may temporarily double the size of your statistics repositories. Consider optimizing your solr repos when complete.

If a UUID value cannot be found for a legacy id, the legacy id will be converted to the form "xxxx-unmigrated" where xxxx is the legacy id.

Routine Solr Index Maintenance

Command used:	`[dspace]/bin/dspace stats-util`
Java class:	org.dspace.statistics.util.StatisticsClient
Arguments (short and long forms):	Description
`-o` or `--optimize`	Run maintenance on the SOLR index. No longer recommended

Notes:

The usage of this this option is strongly recommended, you should run this script daily (from crontab or your system's scheduler), to prevent your servlet container from running out of memory.

Solr Sharding By Year

Command used:	`[dspace]/bin/dspace stats-util`
Java class:	org.dspace.statistics.util.StatisticsClient
Arguments (short and long forms):	Description
`-s` or `--shard-solr-index`	Splits the data in the main Solr core up into a separate core for each year. This will upgrade the performance of Solr.

Notes:

Yearly Solr sharding is a routine that can drastically improve the performance of your DSpace SOLR statistics. It was introduced in DSpace 3.0 and is not backwards compatible. The routine decreases the load created by the logging of new usage events by reducing the size of the SOLR Core in which new usage data are being logged. By running the script, you effectively split your current SOLR core, containing all of your usage events, into different SOLR cores that each contain the data for one year. In case your DSpace has been logging usage events for less than one year, you will see no notable performance improvements until you run the script after the start of a new year. Both writing new usage events as well as read operations should be more performant over several smaller SOLR Shards instead of one monolithic one.

It is highly recommended that you execute this script once at the start of every year. To ensure this is not forgotten, you can include it in your crontab or other system scheduling software. Here's an example cron entry (just replace [dspace] with the full path of your DSpace installation):

```
# At 12:00AM on January 1, "shard" the DSpace Statistics Solr index.  Ensures each year has its own Solr
index - this improves performance.
0 0 1 1 * [dspace]/bin/dspace stats-util -s
```

You MUST restart Tomcat after sharding

After running the statistics shard process, the "View Usage Statistics" page(s) in DSpace will **not** automatically recognize the new shard.

Restart tomcat to ensure that the new shard is recognized & included in usage statistics queries.

Repair of Shards Created Before DSpace 5.7 or DSpace 6.1

If you ran the shard process before upgrading to DSpace 5.7 or DSpace 6.1, the multi-value fields such as owningComm and onwningColl are likely be corrupted. Previous versions of the shard process lost the multi-valued nature of these fields. Without the multi-valued nature of these fields, it is difficult to query for statistics records by community / collection / bundle.

You can verify this problem in the solr admin console by looking at the owningComm field on existing records and looking for the presence of "\\," within that field.

The following process may be used to repair these records.

1. Backup your solr statistics-xxxx directories while tomcat is down.
2. Backup and delete the contents of the dspace-install/solr-export directory
3. For each "statistics-xxxx" shard that exists, export the repository

```
dspace solr-export-statistics -i statistics-xxxx -f
```

4. Run the following to repair records in the dspace-install/solr-export directory

```
for file in *
do
sed -E -e "s/[\\]+,/,/g" -i $file
done
```

5. For each shard that was exported, run the following import

```
dspace solr-import-statistics -i statistics-xxxx -f
```

If you repeat the query that was run previously, the fields containing "\\," should now contain an array of owning community ids.

Shard Naming

Prior to the release of DSpace 6.1, the shard names created were off by one year in timezones with a positive offset from GMT.

Shards created subsequent to this release may appear to skip by one year.

See DS-3437[308] - When sharding statistics, the destination shard name is off by one year CLOSED

Technical implementation details

After sharding, the Solr data cores are located in the [dspace.dir]/solr directory. There is no need to define the location of each individual core in solr.xml because they are automatically retrieved at runtime. This retrieval happens in the *static* method located in the *org.dspace.statistics.SolrLogger* class. These cores are stored in the *statisticYearCores* list. Each time a query is made to Solr, these cores are added as shards by the *addAdditionalSolrYearCores* method. The cores share a common configuration copied from your original *statistics* core. Therefore, no issues should be resulting from subsequent `ant updates`.

The actual sharding of the of the original Solr core into individual cores by year is done in the *shardSolrIndex* method in the *org.dspace.statistics.SolrLogger* class. The sharding is done by first running a facet on the time to get the facets split by year. Once we have our years from our logs we query the main Solr data server for all information on each year & download these as CSVs. When we have all data for one year, we upload it to the newly created core of that year by using the update csv[309] handler. Once all data of one year have been uploaded, those data are removed from the main Solr (by doing it this way if our Solr crashes we do not need to start from scratch).

Multiple Shard Fix (DSpace 6.1)

A bug exists in the DSpace 6.0 release that prevents tomcat from starting when multiple shards are present.

To address this issue, the initialization of SOLR shards is deferred until the first SOLR related requests are processed.

See DS-3457[310] - Tomcat Restart Hangs after Sharding DSpace 6x Statistics CLOSED

Testing Solr Shards

Testing Solr Shards(see page 359)

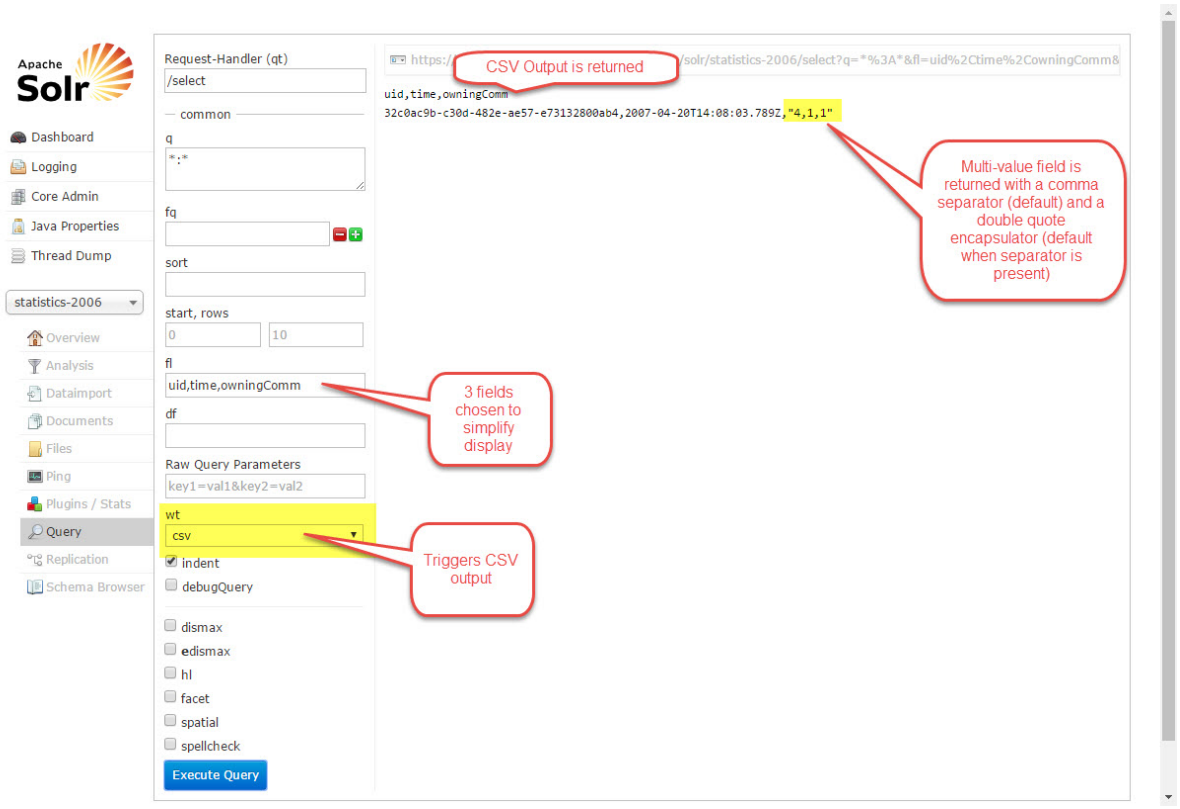
Testing Solr Shards

These notes detail how to test and manipulate SOLR statistics shards.

Testing CSV Export

The SOLR Admin Console provides a mechanism to test the CSV Export Process and Parameters

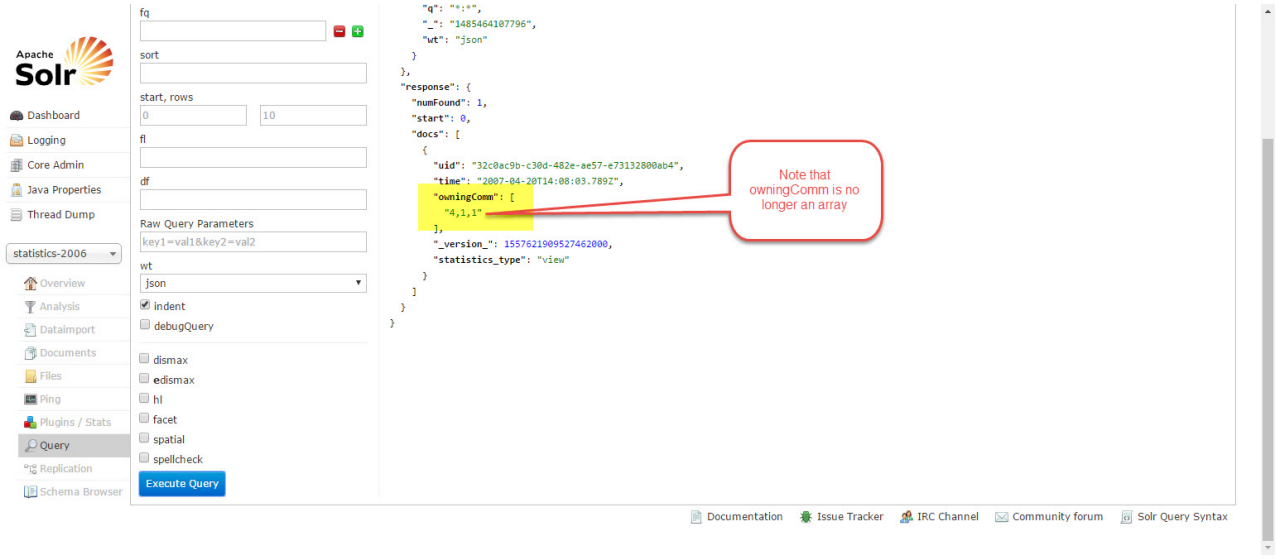
308 https://jira.lyrasis.org/browse/DS-3437?src=confmacro
309 http://wiki.apache.org/solr/UpdateCSV
310 https://jira.lyrasis.org/browse/DS-3457?src=confmacro



Testing CSV Import

The SOLR Admin Console provides a mechanism to access the CSV Upload process. Unfortunately, it does not all parameters to be provided.

Note that the multi-value field is corrupted if you import by this manner.



It is possible to csv import parameters using curl.

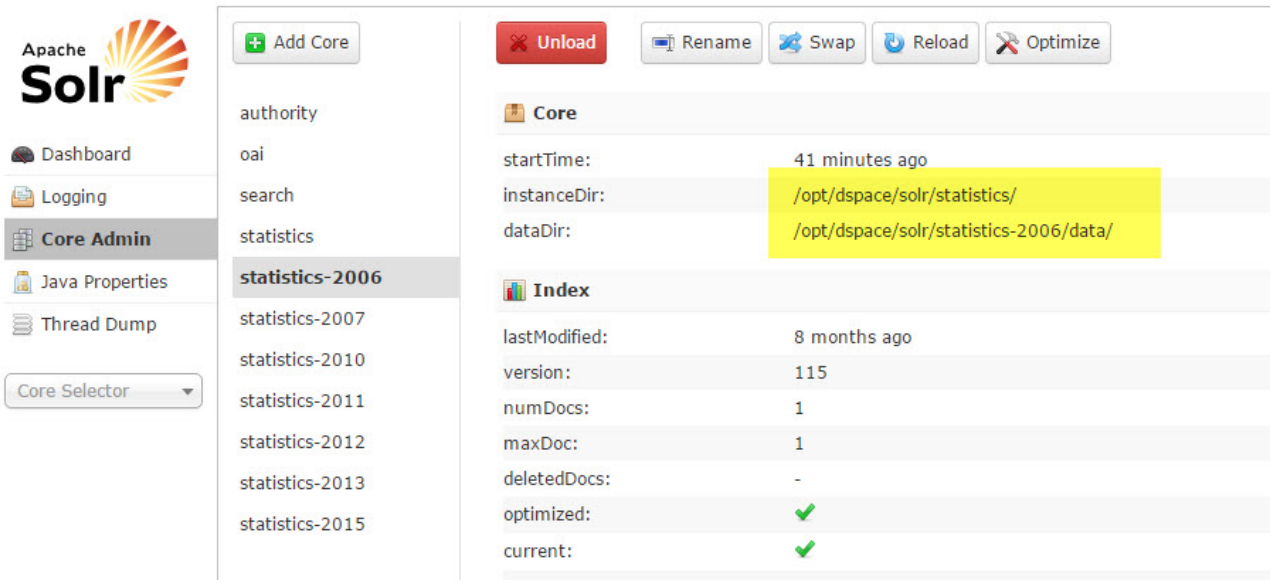
Running CSV Upload with curl

```
curl -F "data=@statistics-2006_export_2007-04.csv" "http://localhost/solr/
statistics-2006/update/csv?
skip=_version_&csv.mv.escape=%5C&f.owningColl.split=true&f.owningColl.separator=%7C&f.
owningComm.split=true&f.owningComm.separator=,&f.owningItem.split=true&f.owningItem.se
parator=%7C&f.bundleName.split=true&f.bundleName.separator=%7C&stream.contentType=text
%2Fcsv%3Bcharset%3Dutf-8&commit=true&softCommit=false&waitSearcher=true&wt=javabin&ver
sion=2"
```

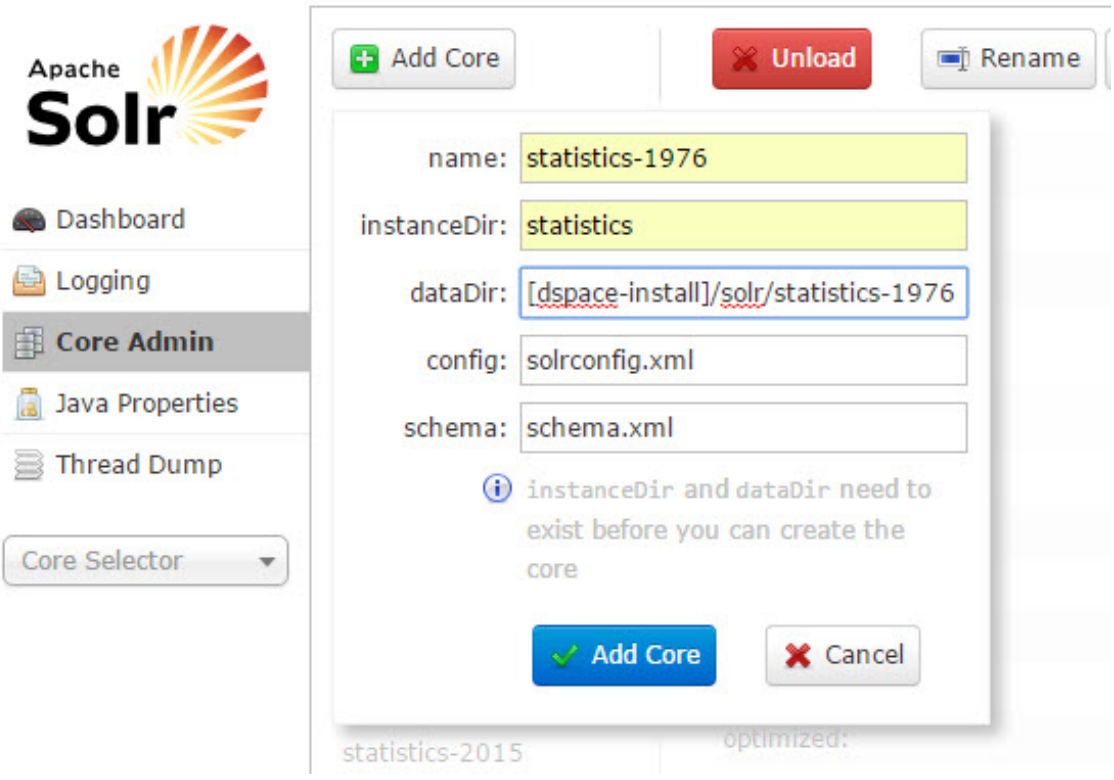
Creating a Shard in the Admin Console

While this is probably not necessary, it is possible to create an empty shard in the Solr Admin console.

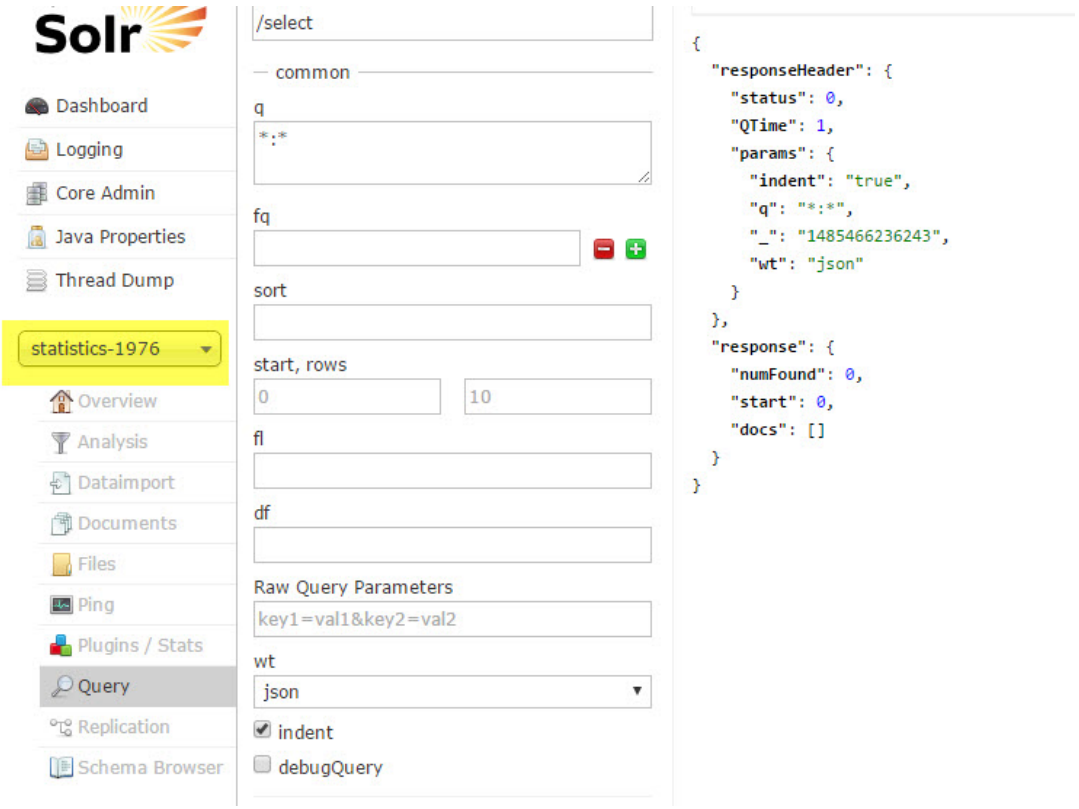
Note that existing shards use the statistics directory as an "instance" directory.



Manually create a new shard



The new shard can be queried like the other ones



4.8.2 DSpace Google Analytics Statistics

4.8.2.1 Google Analytics Recording

It is possible to record User Interface traffic by enabling the recording of Google Analytics data within DSpace using the `google.analytics.key` in the DSpace configuration file dspace.cfg. Until DSpace version 5.0 only User Interface activity could be recorded, that is to say that downloads initiated straight from a Google search (or any other search engine) were not recorded. As of DSpace version 5.0 downloads are now recorded as Google 'Events', so that all item page views and bitstream downloads are now recorded.

4.8.2.2 Google Analytics Reporting

DSpace 7.0 does not yet support

Google Analytics Reporting is not available in DSpace 7.0. It is under discussion as it's unclear how many sites use it. At this time it is tentatively scheduled for discussion as part of 7.2, see DSpace Release 7.0 Status[311]

311 https://wiki.lyrasis.org/display/DSPACE/DSpace+Release+7.0+Status

As of DSpace version 5.0 it has also become possible to expose that recorded Google Analytics data within DSpace. The data is retrieved from Google using the Google Analytics Reporting API v3. This feature is disabled by default, to enable it please follow the instructions below.

Please read the documentation found at https://developers.google.com/analytics/devguides/reporting/core/v3/ and https://developers.google.com/accounts/docs/OAuth2ServiceAccount. It is the definitive documentation, however, it is over detailed for our purposes so the critical steps are summarised below. The theory is that as a developer you would create a Google project, write your application and store the code in the Google code repository, then create a Google Service Account which your application could use to retrieve data from the Google Analytics API. In our case we already have our application, DSpace, but we still have to go through the motions of creating a project in order to be able to be able to generate the Service Account which we need to allow DSpace to talk to the Google Analytics API.

1. Logon to the Google Developers Console https://console.developers.google.com/project with whatever email address you use to access/manage your existing Google Analytics account(s).
2. Create a new Google Project. The assumption is that you are developing some new software and will make use of the Google code repository. This is not the case but you need to create the skeleton project before you can proceed to the next step.
3. Enable the Analytics API for the project. In the sidebar on the left, expand **APIs & auth**. Next, click **APIs**. In the list of APIs, make sure the status is **ON** for the Analytics API.
4. In the sidebar on the left, select **Credentials**.
5. Select **OAuth / Create new Client ID**, then in the subsequent popup screen select **Service account**. This will automatically generate the required Service Account email address and certificate.
6. Go to your Google Analytics dashboard http://www.google.com/analytics/. Create an account for the newly generated Service Account email address and give it permission to 'Read and Analyze' at account level. See *Note below.
7. The generated certificate needs to be placed somewhere that your DSpace application can access and be referenced as described below in the configuration section..

*Note:- The Google documentation specifies that the Service Account email address should only require 'Read and Analyze' permission. However, it would appear this may not be the case and it may be necessary to grant greater permissions, at least initially.

4.8.2.3 Configuration settings for Google Analytics Statistics

In the `[dspace.dir]/config/modules/google-analytics.cfg` file review the following fields. These should be either edited directly or overridden in your local.cfg config file (see Configuration Reference(see page 552)).

Property:	google-analytics.application.name
Value:	Dummy Project
Informational Note:	Not sure if this property is required but it was in the example code provided by Google. Please do not delete.
Property:	google-analytics.table.id
Example Value:	ga:12345678

Informational Note:	Logon to the Google Analytics Dashboard and select the Property (or website in plain English) that you wish to target. Then select the Admin section for the property. You should then be able to select the 'view settings' for the view you are interested in. The View ID should replace 12345678 below (note - confusingly the Reporting API documentation refers to the View ID as Table ID).
Property:	google-analytics.account.email
Example Value:	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx@developer.gserviceaccount.com
Informational Note:	The email address automatically generated when you created the Service Account.
Property:	google-analytics.certificate.location
Example Value:	/home/example/dslweb--privatekey.p12
Informational Note:	The certificate file automatically generated when you created the Service Account.
Property:	google-analytics.authorization.admin.usage
Example Value:	true
Informational Note:	Control if the statistics pages should be only shown to authorized users. If enabled, only the administrators for the DSpaceObject will be able to view the statistics. If disabled, anyone with READ permissions on the DSpaceObject will be able to view the statistics.

4.8.3 Exchange usage statistics with IRUS

4.8.3.1 Introduction

IRUS (Institutional Repository Usage Statistics)[312] enables Institutional Repositories to share and expose statistics based on the COUNTER standard.
It offers opportunities for benchmarking and acts as an intermediary between repositories and other agencies.

IRUS is currently available in the following areas/countries:

- United Kingdom: https://irus.jisc.ac.uk[313]/[314]
- Australia and New-Zealand: https://irus.jisc.ac.uk/irus-anz/
- United States: https://irus.jisc.ac.uk/irus-us/

4.8.3.2 Prerequisite

The DSpace server should be able to access the tracker's base production and test URL's.
The tracker's base production URL will depend on the area/country where your repository is located:

- United Kingdom: http://irus.jisc.ac.uk/counter/
- Australia and New-Zealand: https://irus.jisc.ac.uk/counter/anz/
- United States: https://irus.jisc.ac.uk/counter/us/

The tracker's base test URL is common to all areas/countries:

> https://irus.jisc.ac.uk/counter/test/

Access to the tracker's base URLs can easily be verified using a *wget* command with the applicable URL, e.g.:

```
wget https://irus.jisc.ac.uk/counter/test/
```

The above command should return a HTTP 200.

4.8.3.3 Configuration

The IRUS statistics tracker can be configured in the irus-statistics.cfg file which can be found [dspace-src]/dspace/config/modules.

Property	Description	Default
`irus.statistics.tracker.enabled`	Configuration used to enable the IRUS statistics tracker. Set to true to enable.	`false`
`irus.statistics.tracker.type-field`	Metadata field to check if certain items should be excluded from tracking. If empty or commented out, all items are tracked.	
`irus.statistics.tracker.type-value`	The values in the above metadata field that will be considered to be tracked.	

312 http://www.irus.mimas.ac.uk/
313 https://irus.jisc.ac.uk/
314 https://irus.jisc.ac.uk/

Property	Description	Default
`irus.statistics.tracker.entity-types`	The entity types to be included in the tracking. If left empty, only publication hits will be tracked.	`Publication`
`irus.statistics.tracker.environment`	The tracker environment determines to which url the statistics are exported (test or prod).	`test`
`irus.statistics.tracker.testurl`	The url to which the trackings are exported when testing. (In theory, this should be https://irus.jisc.ac.uk/counter/test/)	
`irus.statistics.tracker.produrl`	The url to which the trackings are exported in production. (this will depend on your area/country, refer to the Prerequisite section)	
`irus.statistics.tracker.urlversion`	Tracker version	
`irus.statistics.spider.agentregex.url`	External URL to the COUNTER user agents file (e.g. https://github.com/atmire/COUNTER-Robots/blob/master/generated/COUNTER_Robots_list.txt)	
`irus.statistics.spider.agentregex.regexfile`	Location where the user agents file should be downloaded to.	

4.9 User Interface

4.9.1 User Interface Configuration

4.9.1.1 Overview

As the DSpace 7 User Interface is built on Angular.io[315], it aligns with many of the best practices of that platform & the surrounding community. One example is that Configuration files (along with much of the UI code) use the TypeScript language[316]. That said, you do NOT need to be deeply familiar with TypeScript to edit the configuration files. Much of the syntax is very similar to JSON, which is a common configuration format.

> **You must rebuild after any configuration change**
>
> At this time, you must rebuild the UI anytime you modify a configuration setting. This is because the currently active configuration is in your `./src/environments/environment.ts` which is the compiled/ deduped version of your configuration after applying all overrides.

4.9.1.2 Configuration Override

The UI configuration files reside in the `./src/environments/` folder in the Angular UI source code[317]. The default configuration are in `environment.common.ts` in that directory.

To change the default configuration values, you simply create (one or more) local files that override the parameters you need to modify. You can use `environment.template.ts` as a starting point.

- For example, create a new `environment.dev.ts` file in `src/environments/` for a `development` environment;
- For example, create a new `environment.prod.ts` file in `src/environments/` for a `production` environment;

The "ui" and "rest" sections of the configuration may also be overridden separately via one of the following

- By setting any of the following environment variables in your system:

315 https://angular.io/
316 https://www.typescriptlang.org/
317 https://github.com/DSpace/dspace-angular/

```
# "ui" settings environment variables
DSPACE_HOST # The host name of the angular application
DSPACE_PORT # The port number of the angular application
DSPACE_NAMESPACE # The namespace of the angular application
DSPACE_SSL # Whether the angular application uses SSL [true/false]

# "rest" settings environment variables
DSPACE_REST_HOST # The host name of the REST application
DSPACE_REST_PORT # The port number of the REST application
DSPACE_REST_NAMESPACE # The namespace of the REST application
DSPACE_REST_SSL # Whether the angular REST uses SSL [true/false]
```

- Or, by creating a `.env` (environment) file in the project root directory and setting the environment variables in that location.

The override priority ordering is as follows (with items listed at the top overriding all other settings)

1. Environment variables
2. The ".env" file
3. The "environment.prod.ts", "environment.dev.ts" or "environment.test.ts"
4. The "environment.common.ts"

4.9.1.3 Configuration Reference

The following configurations are available in `./src/environments/environment.common.ts` These settings may be overridden as described above

Production Mode

When Production mode is enabled, this enables Angular's runtime production mode[318] and compresses the built application. This should always be enabled in Production scenarios.

```
production: true
```

UI Core Settings

The "ui" (user interface) section defines where you want Node.js to run/respond. It may correspond to your primary/public URL, but it also may not (if you are running behind a proxy). In this example, we are setting up our UI to just use localhost, port 4000. This is a common setup for when you want to use Apache or Nginx to handle HTTPS and proxy requests to Node.js running on port 4000.

318 https://angular.io/guide/deployment#enable-runtime-production-mode

```
ui: {
  ssl: false,
  host: 'localhost',
  port: 4000,
  // NOTE: Space is capitalized because 'namespace' is a reserved string in TypeScript
  nameSpace: '/',
  // The rateLimiter settings limit each IP to a "max" of 500 requests per "windowMs" (1 minute).
  rateLimiter: {
    windowMs: 1 * 60 * 1000,   // 1 minute
    max: 500 // limit each IP to 500 requests per windowMs
  }
},
```

The "rateLimiter" sub-section can be used to protect against a DOS (denial of service) attack when the UI is processed on the server side (i.e. server-side rendering). Default settings are usually OK. In Angular, server-side rendering occurs to support better Search Engine Optimization(see page 485) (SEO), as well as to support clients which cannot use Javascript. See also Angular's docs on Server-side rendering[319].

REST API Settings

The "rest" (REST API) section defines which REST API the UI will use. The REST settings MUST correspond to the primary URL of the backend. Usually, this means they must be kept in sync
with the value of `dspace.server.url` in the backend's `local.cfg`

This example is valid if your Backend is publicly available at https://api.mydspace.edu/server/ . Keep in mind that the "port" must always be specified even if it's a standard port (i.e. port 80 for HTTP and port 443 for HTTPS).

```
rest: {
  ssl: true,
  host: 'api.mydspace.edu',
  port: 443,
  // NOTE: Space is capitalized because 'namespace' is a reserved string in TypeScript
  nameSpace: '/server'
},
```

Cache Settings

The "cache" section controls how long objects/responses will remain in the UI cache. The defaults should be OK for most sites.

319 https://angular.io/guide/universal

```
cache: {
  // NOTE: how long should objects be cached for by default
  msToLive: {
    default: 15 * 60 * 1000, // 15 minutes
  },
  control: 'max-age=60', // revalidate browser
  autoSync: {
    defaultTime: 0,
    maxBufferSize: 100,
    timePerMethod: {[RestRequestMethod.PATCH]: 3} as any // time in seconds
  }
},
```

Authentication Settings

The "auth" section provides some basic authentication-related settings. Currently, it's primarily settings related to when a session timeout warning will be showed to your users, etc.

```
auth: {
  // Authentication UI settings
  ui: {
    // the amount of time before the idle warning is shown
    timeUntilIdle: 15 * 60 * 1000, // 15 minutes
    // the amount of time the user has to react after the idle warning is shown before they are logged out.
    idleGracePeriod: 5 * 60 * 1000, // 5 minutes
  },
  // Authentication REST settings
  rest: {
    // If the rest token expires in less than this amount of time, it will be refreshed automatically.
    // This is independent from the idle warning.
    timeLeftBeforeTokenRefresh: 2 * 60 * 1000, // 2 minutes
  },
},
```

Form Settings

The "form" section provides basic settings for any forms displayed in the UI. At this time, these settings only include a validatorMap, which is not necessary to modify for most sites

```
form: {
  // NOTE: Map server-side validators to comparative Angular form validators
  validatorMap: {
    required: 'required',
    regex: 'pattern'
  }
},
```

Notification Settings

The "notifications" section provides options related to where user notifications will appear in your UI. By default, they appear in the top right corner, and timeout after 5 seconds.

```
notifications: {
  rtl: false,
  position: ['top', 'right'],
  maxStack: 8,
  // NOTE: after how many seconds notification is closed automatically. If set to zero notifications are
not closed automatically
  timeOut: 5000, // 5 second
  clickToClose: true,
  // NOTE: 'fade' | 'fromTop' | 'fromRight' | 'fromBottom' | 'fromLeft' | 'rotate' | 'scale'
  animate: NotificationAnimationsType.Scale
},
```

The set of valid animations can be found in the NotificationAnimationsType[320], and are implemented in `./src/shared/animations/`

Submission Settings

The "submission" section provides some basic Submission/Deposit UI options. These allow you to optionally enable an autosave (disabled by default), and custom styles/icons for metadata fields or authority confidence values.

320 https://github.com/DSpace/dspace-angular/blob/main/src/app/shared/notifications/models/notification-animations-type.ts

```
submission: {
    autosave: {
      // NOTE: which metadata trigger an autosave
      metadata: [],
      /**
       * NOTE: after how many time (milliseconds) submission is saved automatically
       * eg. timer: 5 * (1000 * 60); // 5 minutes
       */
      timer: 0
    },
    icons: {
      metadata: [
        /**
         * NOTE: example of configuration
         * {
         *    // NOTE: metadata name
         *    name: 'dc.author',
         *    // NOTE: fontawesome (v5.x) icon classes and bootstrap utility classes can be used
         *    style: 'fa-user'
         * }
         */
        {
          name: 'dc.author',
          style: 'fas fa-user'
        },
        // default configuration
        {
          name: 'default',
          style: ''
        }
      ],
      authority: {
        confidence: [
          /**
           * NOTE: example of configuration
           * {
           *    // NOTE: confidence value
           *    value: 'dc.author',
           *    // NOTE: fontawesome (v4.x) icon classes and bootstrap utility classes can be used
           *    style: 'fa-user'
           * }
           */
          {
            value: 600,
            style: 'text-success'
          },
          {
            value: 500,
            style: 'text-info'
          },
          {
            value: 400,
            style: 'text-warning'
          },
          // default configuration
```

```
          {
            value: 'default',
            style: 'text-muted'
          },

        ]
      }
    }
  },
```

Universal (Server-side Rendering) Settings

The "universal" section pertains to enabling/disabling Angular Universal for Server-side rendering[321]. DSpace requires server-side rendering to support Search Engine Optimization(see page 485). When it's turned off, your site may not be able to be indexed in Google, Google Scholar and other search engines.

```
// Angular Universal settings
universal: {
  preboot: true,
  async: true,
  time: false
},
```

Debug Settings

The "debug" property allows you to turn on debugging in the Angular UI. When enabled, your environment and all Redux[322] actions/transfers are logged to the console. This is only ever needed if you are debugging a tricky issue.

```
// NOTE: will log all redux actions and transfers in console
debug: false
```

Language Settings

The "defaultLanguage" and "languages" sections allow you to customize which languages to support in your User Interface. See also Multilingual Support(see page 407).

321 https://angular.io/guide/universal
322 https://redux.js.org/

```
// Default Language in which the UI will be rendered if the user's browser language is not an active
language
defaultLanguage: 'en',
// Languages. DSpace Angular holds a message catalog for each of the following languages.
// When set to active, users will be able to switch to the use of this language in the user interface.
languages: [{
  code: 'en',
  label: 'English',
  active: true,
}, {
  code: 'de',
  label: 'Deutsch',
  active: true,
},
...
],
```

The DSpace UI requires that a corresponding language pack file (named with the language code and ending in ".json5") be placed in `./src/assets/i18n/`. See also DSpace 7 Translation - Internationalization (i18n) - Localization (l10n)[323] for information about how to create and contribute these files.

Browse By Settings

The "browseBy" section allows you to customize which "Browse by" options appear in the "All of DSpace" header menu at the top of your DSpace site. The "id" MUST correspond to the *name* of a valid Browse index available from your REST API (see documentation on the REST API /api/discover/browses endpoint[324]). It is possible to configure additional indexes on the Backend using Discovery(see page 386), and any configured index appears in your REST API.

323 https://wiki.lyrasis.org/pages/viewpage.action?pageId=117735441
324 https://github.com/DSpace/RestContract/blob/main/browses.md

```
browseBy: {
  // Amount of years to display using jumps of one year (current year - oneYearLimit)
  oneYearLimit: 10,
  // Limit for years to display using jumps of five years (current year - fiveYearLimit)
  fiveYearLimit: 30,
  // The absolute lowest year to display in the dropdown (only used when no lowest date can be found for
all items)
  defaultLowerLimit: 1900,
  // List of all the active Browse-By types
  // Adding a type will activate their Browse-By page and add them to the global navigation menu,
  // as well as community and collection pages
  // Allowed fields and their purpose:
  //    id:             The browse id to use for fetching info from the rest api
  //    type:           The type of Browse-By page to display
  //    metadataField:  The metadata-field used to create starts-with options (only necessary when the type
is set to 'date')
  types: [
    {
      id: 'title',
      type: BrowseByType.Title,
    },
    {
      id: 'dateissued',
      type: BrowseByType.Date,
      metadataField: 'dc.date.issued'
    },
    {
      id: 'author',
      type: BrowseByType.Metadata
    },
    {
      id: 'subject',
      type: BrowseByType.Metadata
    }
  ]
},
```

Undo Settings

Both the "item" edit and "collection" edit screens allow you to undo changes within a specific time. This is controlled by these settings:

```
item: {
  edit: {
    undoTimeout: 10000 // 10 seconds
  }
},
collection: {
  edit: {
    undoTimeout: 10000 // 10 seconds
  }
},
```

Theme Settings

The "themes" section allows you to configure which theme(s) are enabled for your DSpace site (with the default theme being the "dspace" one). You can enable a single theme across all pages, and/or enable specific alternative themes based on a specific Community, Collection or Item (by UUID or Handle), or based on a Regex match of a URL pattern. This allows you fine grained control over how your site looks, including the ability to customize it per Community or Collection or even per specific pages in the site. *See* User Interface Customization(see page 378) *for details of how to create a new, custom theme.*

```
themes: [
  // Add additional themes here. In the case where multiple themes match a route, the first one
  // in this list will get priority. It is advisable to always have a theme that matches
  // every route as the last one

  // {
  //   // A theme with a handle property will match the community, collection or item with the given
  //   // handle, and all collections and/or items within it
  //   name: 'custom',
  //   handle: '10673/1233'
  // },
  // {
  //   // A theme with a regex property will match the route using a regular expression. If it
  //   // matches the route for a community or collection it will also apply to all collections
  //   // and/or items within it
  //   name: 'custom',
  //   regex: 'collections\/e8043bc2.*'
  // },
  // {
  //   // A theme with a uuid property will match the community, collection or item with the given
  //   // ID, and all collections and/or items within it
  //   name: 'custom',
  //   uuid: '0958c910-2037-42a9-81c7-dca80e3892b4'
  // },
  // {
  //   // A theme with only a name will match every route
  //   name: 'custom'
  // },
  // {
  //   // This theme will use the default bootstrap styling for DSpace components
  //   name: BASE_THEME_NAME
  // },
  {
    // The default dspace theme
    name: 'dspace'
  },
],
```

Media Viewer Settings

The DSpace UI comes with a basic, out-of-the-box Media Viewer (disabled by default). You can choose to enable it for "image/*" or "video/*" MIME types, or both.

```
// Whether to enable media viewer for image and/or video Bitstreams (i.e. Bitstreams whose MIME type starts
with "image" or "video").
// For images, this enables a gallery viewer where you can zoom or page through images.
// For videos, this enables embedded video streaming
mediaViewer: {
  image: false,
  video: false,
},
```

4.9.2 User Interface Customization

4.9.2.1 Angular Overview

The DSpace User Interface (UI) is built on the Angular.io[325] framework. All data comes from the REST API(see page 502) (DSpace Backend), and the final HTML pages are generated via TypeScript[326].

Before getting started in customizing or branding the UI, there are some basic Angular concepts to be aware of. *You do not need to know Angular or TypeScript to theme or brand the UI.* But, understanding a few basic concepts will allow you to better understand the folder structure / layout of the codebase.

Angular Components: In Angular, every webpage consists of a number of "components" which define the structure of a page. They are the main "building block" of any Angular application. Components are reusable across many pages. So, for example, there's only one "header" and "footer" component, even though they appear across all pages.

Each Component has:

- A `*.component.ts` (TypeScript[327]) file which contains the logic & name ("selector") of the component
- A `*.component.html` (HTML) file which contains the HTML markup for the component (and possibly references to other embedded components). This is also called the "template".
  - In HTML files, components are named/referenced as HTML-like tags (e.g. `<ds-header>`, `<ds-footer>`). In DSpace's UI, every component starts with "ds-" in order to distinguish it as an out-of-the-box DSpace component.
- A `*.component.scss` (Sass[328] / CSS) file which contains the style for the component.

325 https://angular.io/
326 https://www.typescriptlang.org/
327 https://www.typescriptlang.org/
328 https://sass-lang.com/

If you want a deeper dive into Angular concepts of Components and Templates, see https://angular.io/guide/architecture-components

4.9.2.2 Theme Technologies

The DSpace UI uses:

- Bootstrap[329] (v4.x) website framework for general layout & webpage components (buttons, alerts, etc)
- Sass[330], a CSS preprocessor, for stylesheets. Sass is very similar to CSS (an in fact, any CSS is valid Sass). But, Sass allows you to nest CSS rules & have variables and functions. For a brief overview on Sass, see https://sass-lang.com/guide
- HTML5[331], the latest specification of the HTML language

Familiarity with these technologies (or even just CSS + HTML) is all you need to do basic theming of the DSpace UI.

4.9.2.3 Creating a Custom Theme

Theme Directories & Design Principles

Out of the box, there are three theming layers/directories to be aware of:

- **Base Theme** (`src/app/` directories): The primary look & feel of DSpace (e.g. HTML layout, header/footer, etc) is defined by the HTML5 templates under this directory. Each HTML5 template is stored in a subdirectory named for the Angular component where that template is used. The base theme includes very limited styling (CSS, etc), based heavily on default Bootstrap (4.x) styling[332], and only allowing for minor tweaks to improve WCAG 2.1 AA accessibility.
- **Custom Theme** (`src/themes/custom` directories): This directory acts as the scaffolding or template for creating a new custom theme. It provides (empty) Angular components/templates which allow you to change the theme of individual components. Since all files are empty by default, if you enable this theme (without modifying it), it will look *identical* to the Base Theme.
- **DSpace Theme** (`src/themes/dspace` directories): This is the default theme for DSpace 7. It's a very simple example theme providing a custom color scheme & homepage on top of the Base Theme. It's important to note that this theme ONLY provides custom CSS/images to override our Base Theme. All HTML5 templates are included at the Base Theme level, as this ensures those HTML5 templates are also available to the Custom Theme.

The DSpace UI design principles & technologies are described in more detail at DSpace UI Design principles and guidelines[333]

Getting Started

1. *Start with the "custom" theme:* The best place to start with a new theme is the "custom" theme folder (`src/themes/custom`). This folder contains the boilerplate code for all theme-able components. It's a scaffolding for a new theme which doesn't modify any of the "base theme" (`src/app/` directories). This means that by default it's a plain Bootstrap look and feel, with a few tweaks for better accessibility.
2. *Create your own theme folder OR edit the "custom" theme:* Either edit the "custom" theme directory, or copy the "custom" theme folder (and all its contents) into a new folder under `src/themes/` (choose whatever folder name you want)

329 https://getbootstrap.com/
330 https://sass-lang.com/
331 https://html.spec.whatwg.org/dev/
332 https://getbootstrap.com/docs/4.6/getting-started/introduction/
333 https://wiki.lyrasis.org/display/DSPACE/DSpace+UI+Design+principles+and+guidelines

3. *Register your theme folder (only necessary if you create a new folder in previous step):* Now, we need to make the UI aware of this new theme folder, before it can be used in configuration.
    a. Modify `angular.json` (in the root folder), adding your theme folder's main "theme.scss" file to the "styles" list. The below example is for a new theme folder named `src/themes/mydspacesite/`

```
"styles": [
  "src/styles/startup.scss",
  {
     "input": "src/styles/base-theme.scss",
     "inject": false,
     "bundleName": "base-theme"
  },
  ...
  {
     "input": "src/themes/mydspacesite/styles/theme.scss",
     "inject": false,
     "bundleName": "mydspacesite-theme"
  },
]
```

    NOTE: the "bundleName" for your custom them MUST use the format "${folder-name}-theme". E.g. if the folder is named "arc/themes/amazingtheme", then the "bundleName" MUST be "amazingtheme-theme"
4. *Enable your theme:* Modify your `src/environments/environment.*.ts` configuration file, adding your new theme to the "themes" array in that file. Pay close attention to modify the correct environment file (e.g. modify environment.dev.ts if running in dev mode, or environment.prod.ts if running in prod mode). We recommend starting in "dev mode" (environment.dev.ts) as this mode lets you see your changes immediately in a browser without a full rebuild of the UI – see next step.

```
// In this example, we only show one theme enabled. It's possible to enable multiple (see below
note)
themes: [
 {
    name: 'mydspacesite'
 },
]
```

    NOTE: The "name" used is the name of the theme's folder, so the example is for enabling a theme at `src/themes/mydspacesite/` globally. You should also comment out the default "dspace" theme, if you intend to replace it entirely.
    NOTE #2: You may also choose to enable multiple themes for your site, and even specify a different theme for different Communities, Collections, Items or URL paths. See User Interface Configuration(see page 367) for more details on "Theme Settings" in your environment.*.ts
5. *Verify your settings by starting the UI* (ideally in Dev mode): At this point, you should verify the basic settings you've made all "work". We recommend doing your theme work while running the UI in "dev mode", as the UI will auto-restart anytime you save a new change. This will allow you to quickly see the impact of each change in your browser.

```
# Start in dev mode (which uses environment.dev.ts)
yarn start:dev
```

6. At this point, you can start making changes to your theme. See the following sections for examples of how to make common changes.

Global style/font/color customizations

Changes to the global Bootstrap variables or styles will apply to all pages / Angular components across the entire site.

1. *Global style changes:* All global style changes can be made in your theme's `styles` folder (e.g. `src/themes/mydspacesite/styles`). There are four main files in that folder:
   a. `_theme_sass_variable_overrides.scss` - May be used to override Bootstrap's default Sass variables. This is the file you may wish to use for **most** style changes. There are a large number of Bootstrap variables available which control everything from fonts, colors and the base style for all Bootstrap web components. For a full list of Bootstrap variables you can override in this file, see the `node_modules/bootstrap/scss/_variables.scss` file (which is installed in your source directory when you run "yarn install"). More information may also be found in the Bootstrap Sass documentation at https://getbootstrap.com/docs/4.0/getting-started/theming/#sass
   b. `_theme_css_variable_overrides.scss` - May be used to override DSpace's default CSS variables. DSpace's UI uses CSS variables for all its components. These variables all start with "--ds-*", and are listed in `src/styles/_custom_variables.scss`. You can also use this file to add your own, custom CSS variables which you want to use for your theme. If you create custom variables, avoid naming them with a "--ds-*" or a "--bs-*" prefix, as those are reserved for DSpace and Bootstrap variables.
   c. `_global-styles.scss` - May be used to modify the global CSS/SCSS for the site. This file may be used to override the default global style contained in `src/styles/_global-styles.scss`. Keep in mind, many styles can be more quickly changed by simply updating a variable in one of the above "*_variable_overrides.scss" files. So, it's often easier to use those first, where possible.
   d. `theme.scss` - This just imports all the necessary Sass files to create the theme. It's unnecessary to modify directly, unless you with to add new Sass files to your theme.
2. *Modifying the default font:* By default, DSpace uses Bootstrap's "native font stack"[334], which just uses system UI fonts. However, the font used in your site can be quickly updated via Bootstrap variables in your theme's `_theme_sass_variable_overrides.scss` file.
   a. One option is to add a new import statement and modify the "$font-family-sans-serif" variable:

```
// Import the font (from a URL)
@import url('https://fonts.googleapis.com/css?family=Source+Sans+Pro');

// Configure Bootstrap to use this font (and list a number of backup fonts to use on various systems)
$font-family-sans-serif: 'Source Sans Pro', -apple-system, BlinkMacSystemFont, "Segoe UI", "Roboto", "Helvetica Neue", Arial, sans-serif, "Apple Color Emoji", "Segoe UI Emoji", "Segoe UI Symbol" !default;
```

   b. If your font requires installing local files, you can do the following
      i. Copy your font file(s) in your theme's `assets/fonts/` folder
      ii. Create a new ".scss" file specific to your font in that folder, e.g. `assets/fonts/open-sans.scss`, and use the "@font-face" CSS rule[335] to load that font:

334 https://getbootstrap.com/docs/4.0/content/reboot/#native-font-stack
335 https://developer.mozilla.org/en-US/docs/Web/CSS/@font-face

open-sans.scss

```
@font-face {
  font-family: "Open Sans";
  src: url("/assets/fonts/OpenSans-Regular-webfont.woff2") format("woff2"),
       url("/assets/fonts/OpenSans-Regular-webfont.woff") format("woff");
}
```

iii. Then, import that new "open-sans.scss" file and use it in the "$font-family-sans-serif" variable

```
// Import the font via the custom SCSS file
@import '../assets/fonts/open-sans';

// Configure Bootstrap to use this font (and list a number of backup fonts to use on
various systems)
$font-family-sans-serif: 'Open Sans', -apple-system, BlinkMacSystemFont, "Segoe UI",
"Roboto", "Helvetica Neue", Arial, sans-serif, "Apple Color Emoji", "Segoe UI Emoji",
"Segoe UI Symbol" !default;
```

c. Keep in mind, as changing the font just requires adjusting Bootstrap Sass variables, there are a lot of Bootstrap guides out there that can help you make more advanced changes

3. *Modifying default color scheme:* The colors used in your site can be quickly updated via Bootstrap variables in your theme's `_theme_sass_variable_overrides.scss` file.
   a. Again, you can use entirely Bootstrap variables to adjust color schemes. See the Bootstrap Theming Colors documentation[336]
   b. A list of all Bootstrap color variables can be found in the `node_modules/bootstrap/scss/_variables.scss` file
   c. Additional examples can be found in the out-of-the-box "dspace" theme, which adjusts the default Bootstrap colors slightly for both accessibility & to match the DSpace logo.
4. Any changes require rebuilding your UI. If you are running in "dev mode" (yarn start:dev), then the UI will restart automatically whenever changes are detected.

Customize Logo in Header

1. Copy your logo to your theme's `assets/image/` folder. Anything in this theme folder will be deployed to `/assets/[theme-name]/images/` URL path.
2. Edit your theme's `app/header/header.component.ts` file. Swap the "templateUrl" property that your theme is using the local copy of "header.component.html"

336 https://getbootstrap.com/docs/4.1/getting-started/theming/#color

header.component.ts

```
@Component({
  selector: 'ds-header',
  // styleUrls: ['header.component.scss'],
  styleUrls: ['../../../../app/header/header.component.scss'],
  // Uncomment the templateUrl which references the "header.component.html" file in your theme
directory
  templateUrl: 'header.component.html',
  // Comment out the templateUrl which references the default "src/app/header/header.component.html"
file.
  //templateUrl: '../../../../app/header/header.component.html',
})
```

3. Your theme's version of the `header.component.html` file will be empty by default. Copy over the default HTML code from `src/app/header/header.component.html` into your version of this file.
4. Then, modify your copy of header.component.html to use your logo. In this example, we're assuming your theme name is "mytheme" and the logo file is named "my-logo.svg"

```
<header>
  <div class="container">
    <div class="d-flex flex-row justify-content-between">
      <a class="navbar-brand my-2" routerLink="/home">
        <!-- Modify the logo on the next line -->
        <img src="/assets/mytheme/images/my-logo.svg" [attr.alt]="'menu.header.image.logo' |
translate"/>
      </a>
      ...
</header>
```

5. Obviously, you can also make additional modifications to the HTML of the header in this file! You'll also see that the header references several other DSpace UI components (e.g. `<ds-search-navbar>` is the search icon in the header). You can easily comment out these tags to disable them, or move them around to change where that component appears in the header.
6. Any changes require rebuilding your UI. If you are running in "dev mode" (yarn start:dev), then the UI will restart automatically whenever changes are detected.

Customize Footer

1. First, you'll want to decide if you want to modify just the footer's HTML, or the footer's styles (CSS/Sass), or both.
    a. If you want to modify the HTML, you'll need to create a copy of "footer.component.html" in your theme, where you place your changes.
    b. If you want to modify the styles, you'll need to create a copy of "footer.component.scss" in your theme, where you place your changes.
2. Edit your theme's `app/footer/footer.component.ts` file. Swap the "templateUrl" and "styleUrls" properties, based on which you want to modify in your theme.

footer.component.ts

```
@Component({
  selector: 'ds-footer',
  // If you want to modify styles, then...
  // Uncomment the styleUrls which references the "footer.component.scss" file in your theme's
directory
  // and comment out the one that references the default "src/app/footer/footer.component.scss"
  styleUrls: ['footer.component.scss'],
  //styleUrls: ['../../../../app/footer/footer.component.scss'],
  // If you want to modify HTML, then...
  // Uncomment the templateUrl which references the "footer.component.html" file in your theme's
directory
  // and comment out the one that references the default "src/app/footer/footer.component.html"
  templateUrl: 'footer.component.html'
  //templateUrl: '../../../../app/footer/footer.component.html'
})
```

3. Now, based on what you want to modify, you will need to either update your theme's copy of `footer.component.html` or `footer.component.scss` or both.
    a. To change footer HTML: Your theme's version of the `footer.component.html` file will be empty by default. Copy over the default HTML code from `src/app/footer/footer.component.html` into your version of this file.
    b. To change footer Styles: Your theme's version of the `footer.component.scss` file will be empty by default. Copy over the default Sass code from `src/app/footer/footer.component.scss` into your version of this file.
4. Modify the HTML or Sass as you see fit.
    a. If you want to add images, add them to your theme's `assets/images` folder. Then reference them at the `/assets/[theme-name]/images/` URL path.
    b. Keep in mind, all Bootstrap variables, utility classes & styles can be used in these files. Take advantage of Bootstrap when you can do so.
5. DSpace also has a option to display a two-level footer, which is becoming more common these days. By default. DSpace just displays a small, bottom footer. But, you can enable a top footer (above that default footer) by add this line into your theme's `footer.component.ts`

    footer.component.ts

    ```
    export class FooterComponent extends BaseComponent {
       // This line will enable the top footer in your theme
       showTopFooter = true;
    }
    ```

    This top footer appears in the `footer.component.html` within a div. Notice the "`*ngIf='showTopFooter'`", which only shows that div when that variable is set to "true"

footer.component.html

```
<footer class="text-lg-start">
  <!-- This div and everything within it only displays if showTopFooter=true -->
  <div *ngIf="showTopFooter" class="top-footer">
    ...
  </div>
  <!-- The bottom footer always displays -->
  <div class="bottom-footer ...">
    ...
  </div>
</footer>
```

6. Any changes require rebuilding your UI. If you are running in "dev mode" (yarn start:dev), then the UI will restart automatically whenever changes are detected.

Customize Home Page News

1. First, you'll want to decide if you want to modify just the Home Page News HTML, or styles (CSS/Sass), or both.
    a. If you want to modify the HTML, you'll need to create a copy of "home-news.component.html" in your theme, where you place your changes.
    b. If you want to modify the styles, you'll need to create a copy of "home-news.component.scss" in your theme, where you place your changes.
2. Edit your theme's `app/home-page/home-news/home-news.component.ts` file. Swap the "templateUrl" and "styleUrls" properties, based on which you want to modify in your theme.

footer.component.ts

```
@Component({
  selector: 'ds-home-news',
  // If you want to modify styles, then...
  // Uncomment the styleUrls which references the "home-news.component.scss" file in your theme's
directory
  // and comment out the one that references the default "src/app/home-page/home-news/home-
news.component.scss"
  styleUrls: ['./home-news.component.scss'],
  //styleUrls: ['../../../../../app/home-page/home-news/home-news.component.scss'],
  // If you want to modify HTML, then...
  // Uncomment the templateUrl which references the "home-news.component.html" file in your theme's
directory
  // and comment out the one that references the default "src/app/home-page/home-news/home-
news.component.html"
  templateUrl: './home-news.component.html'
  //templateUrl: '../../../../../app/home-page/home-news/home-news.component.html'
})
```

3. Now, based on what you want to modify, you will need to either update your theme's copy of `home-news.component.html` or `home-news.component.scss` or both.
    a. To change HTML: Your theme's version of the `home-news.component.html` file will be empty by default. Copy over the default HTML code from `src/app/home-page/home-news/home-news.component.html` into your version of this file.

   b. To change Styles: Your theme's version of the `home-news.component.scss` file will be empty by default. Copy over the default Sass code from `src/app/home-page/home-news/home-news.component.scss` into your version of this file.
4. Modify the HTML or Sass as you see fit.
   a. If you want to add images, add them to your theme's `assets/images` folder. Then reference them at the `/assets/[theme-name]/images/` URL path.
   b. Keep in mind, all Bootstrap variables, utility classes & styles can be used in these files. Take advantage of Bootstrap when you can do so.
5. Any changes require rebuilding your UI. If you are running in "dev mode" (yarn start:dev), then the UI will restart automatically whenever changes are detected.

Customize other UI Components

By now, if you've followed this entire guide, you'll notice a pattern! Customizing specific DSpace UI components requires just three steps:

1. *Configure your theme to use its copies of files:* Modify the corresponding `*.component.ts` in your theme.
   a. If you want to modify component style, replace the "styleUrls" in that file to point at the copy of `*.component.scss` in your theme.
   b. If you want to modify component HTML, replace the "template" in that file to point at the copy of `*.component.html` in your theme.
2. *Copy the default UI code into your theme file(s)*
   a. If you want to modify component style, copy the default `*.component.scss` code (from `src/app/`) into your theme's `component.scss` file.
   b. If you want to modify component HTML, copy the default `*.component.html` code (from `src/app/`) into your theme's `component.html` file.
3. *Modify those theme-specific files*
   a. If you want to add images, add them to your theme's `assets/images` folder. Then reference them at the `/assets/[theme-name]/images/` URL path.
   b. Keep in mind, all Bootstrap variables, utility classes & styles can be used in these files. Take advantage of Bootstrap when you can do so.
4. Remember to either rebuild the UI after each change, or run in dev mode (yarn start:dev) while you are doing theme work.

4.9.2.4 Additional Theming Resources

- "Getting Started with DSpace 7.0" Basic Workshop at OR2021 Conference[337]
- Bootstrap Documentation[338] - DSpace's UI strives to be compliant with "out-of-the-box" Bootstrap as much as possible. Therefore, Bootstrap knowledge is very beneficial in customizing DSpace.
- Sass Documentation[339] - Bootstrap and DSpace both use Sass to enhance your ability to customize styles quickly via variables, etc. Some familiarity with Sass is recommended, though you need not be an expert.

4.9.3 Discovery

337 https://wiki.lyrasis.org/display/DSPACE/DSpace+7+at+OR2021
338 https://getbootstrap.com/docs/4.1/getting-started/introduction/
339 https://sass-lang.com/documentation

4.9.3.1 What is DSpace Discovery

The Discovery Module enables faceted searching & browsing for your repository.

Although these techniques are new in DSpace, they might feel familiar from other platforms like Aquabrowser or Amazon, where facets help you to select the right product according to facets like price and brand. DSpace Discovery offers very powerful browse and search configurations that were only possible with code customization in the past.

Watch the DSpace Discovery introduction video[340]

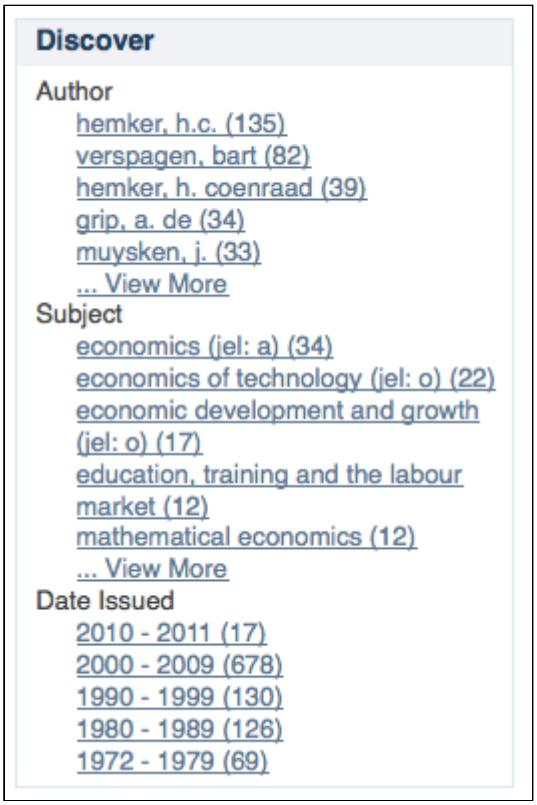
> ⓘ Since 6.0, Discovery is the only out-of-the-box Search and Browse infrastructure provided in DSpace.

What is a Sidebar Facet

From the user perspective, faceted search (also called faceted navigation, guided navigation, or parametric search) breaks up search results into multiple categories, typically showing counts for each, and allows the user to "drill down" or further restrict their search results based on those facets.

When you have successfully enabled Discovery in your DSpace, you will notice that the different enabled facets are visualized in a "Discover" section in your sidebar, by default, right below the Browse options.

340 http://www.youtube.com/v/abRSXTUEwws



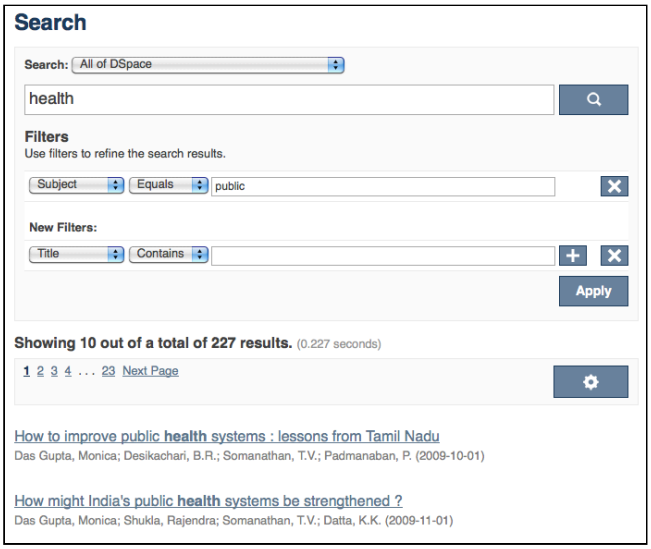
In this example, there are 3 Sidebar Facets, Author, Subject and Date Issued. It's important to know that multiple metadata fields can be included in one facet. For example, the Author facet above includes values from both dc.contributor.author as well as dc.creator.

Another important property of Sidebar Facets is that their contents are automatically updated to the context of the page. On collection homepages or community homepages it will include information about the items included in that particular collection or community.

What is a Search Filter

In a standard search operation, a user specifies his complete query prior to launching the operation. If the results are not satisfactory, the user starts over again with a (slightly) altered query.

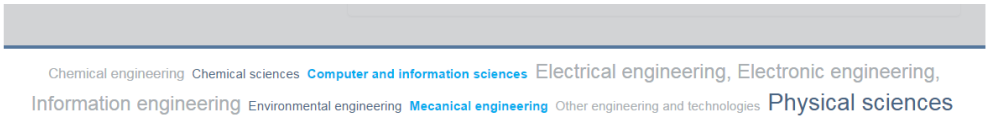
In a faceted search, a user can modify the list of displayed search results by specifying additional "filters" that will be applied on the list of search results. In DSpace, a filter is a contain condition applied to specific facets. In the example below, a user started with the search term "health", which yielded 500 results. After applying the filter "public" on the facet "Subject", only 227 results remain. Each time a user selects a sidebar facet it will be added as a filter. Active filters can be altered or removed in the 'filters' section of the search interface.



Another example: Using the standard search, a user would search for something like [**wetland + "dc.author=Mitsch, William J" + dc.subject="water quality"**]. With filtered search, they can start by searching for [**wetland**], and then filter the results by the other attributes, author and subject.

What is a tag cloud facet

Tag cloud facet is another way to display facets of your repository in a "tag cloud" form in which the importance of each tag is show with font size or color. This format is useful for quickly perceiving the most prominent terms.



This is a classic "tag cloud" facet in a DSpace repository.

4.9.3.2 Configuration files

The configuration for discovery is located in 2 separate files.

- General settings: The `discovery.cfg` file located in the `[dspace-install-dir]/config/modules directory`.
- User Interface Configuration: The `discovery.xml` file is located in `[dspace-install-dir]/config/spring/api/` directory.

4.9.3.3 General Discovery settings (`config/modules/discovery.cfg`)

The `discovery.cfg` file is located in the `[dspace]/config/modules` directory and contains following properties. Any of these properties may be overridden in your `local.cfg` (see Configuration Reference(see page 552)):

Property:	**discovery.search.server**

Example Value:	`discovery.search.server=[http://localhost:8080/solr/search]`
Informational Note:	Discovery relies on a Solr index for storage and retrieval of its information. This parameter determines the location of the Solr index. If you are uncertain whether this property is set correctly, you can use a commandline tool like "wget" to perform a query against the Solr index (and ensure Solr responds). For example, the below query searches the Solr index for "test" and returns the response on standard out: `wget -O - http://localhost:8080/solr/search/select?q=test`
Property:	**discovery.index.authority.ignore[.field]**
Example Value:	`discovery.index.authority.ignore=true` `discovery.index.authority.ignore.dc.contributor.author=false`
Informational Note:	By default, Discovery will use the authority information in the metadata to disambiguate homonyms. Setting this property to false will make the indexing process the same as the metadata doesn't include authority information. The configuration can be different on a field (<schema>.<element>.<qualifier>) basis, the property without field set the default value.
Property:	**discovery.index.authority.ignore-prefered[.field]**
Example Value:	`discovery.index.authority.ignore-prefered=true` `discovery.index.authority.ignore-prefered.dc.contributor.author=false`
Informational Note:	By default, Discovery will use the authority information in the metadata to query the authority for the preferred label. Setting this property to false will make the indexing process the same as the metadata doesn't include authority information (i.e. the preferred form is the one recorded in the metadata value). The configuration can be different on a field (<schema>.<element>.<qualifier>) basis, the property without field set the default value. **If the authority is a remote service, disabling this feature can greatly improve performance.**

Property:	**discovery.index.authority.ignore-variants[.field]**
Example Value:	`discovery.index.authority.ignore-variants=true` `discovery.index.authority.ignore-variants.dc.contributor.author=false`
Informational Note:	By default, Discovery will use the authority information in the metadata to query the authority for variants. Setting this property to false will make the indexing process the same, as the metadata doesn't include authority information. The configuration can be different on a per-field (<schema>.<element>.<qualifier>) basis, the property without field set the default value. **If authority is a remote service, disabling this feature can greatly improve performance.**

4.9.3.4 Modifying the Discovery User Interface (`config/spring/api/discovery.xml`)

The `discovery.xml` file is located in the `[dspace]/config/spring/api` directory.

Structure Summary

This file is in XML format, you should be familiar with XML before editing this file. The configurations are organized together in beans, depending on the purpose these properties are used for.
This purpose can be derived from the class of the beans. Here's a short summary of classes you will encounter throughout the file and what the corresponding properties in the bean are used for.

Download the configuration file and review it together with the following parameters[341]

Class:	**DiscoveryConfigurationService**
Purpose:	Defines the mapping between separate Discovery configurations and individual collections/communities
Default:	All communities, collections and the homepage (key=default) are mapped to defaultConfiguration, also controls the metadata fields that should not be indexed in the search core (item provenance for example).
Class:	**DiscoveryConfiguration**

341 https://wiki.lyrasis.org/download/attachments/104566682/discovery.xml?api=v2&modificationDate=1540322317159&version=1

Purpose:	Groups configurations for sidebar facets, search filters, search sort options and recent submissions
Default:	There is one configuration by default called defaultConfiguration
Class:	**DiscoverySearchFilter**
Purpose:	Defines that specific metadata fields should be enabled as a search filter
Default:	dc.title, dc.contributor.author, dc.creator, dc.subject.* and dc.date.issued are defined as search filters
Class:	**DiscoverySearchFilterFacet**
Purpose:	Defines which metadata fields should be offered as a contextual sidebar browse options, each of these facets has also got to be a search filter
Default:	dc.contributor.author, dc.creator, dc.subject.* and dc.date.issued
Class:	**HierarchicalSidebarFacetConfiguration**
Purpose:	Defines which metadata fields contain hierarchical data and should be offered as a contextual sidebar option
Class:	**DiscoverySortConfiguration**
Purpose:	Further specifies the sort options to which a DiscoveryConfiguration refers
Default:	dc.title and dc.date.issued are defined as alternatives for sorting, other than Relevance (hard-coded)
Class:	**DiscoveryHitHighlightingConfiguration**
Purpose:	Defines which metadata fields can contain hit highlighting & search snippets

Default:	dc.title, dc.contributor.author, dc.subject, dc.description.abstract & full text from text files.
Class:	**TagCloudFacetConfiguration**
Purpose:	Defines the tag cloud appearance configuration bean and the search filter facets to appear in the tag cloud form. You can have different "**TagCloudFacetConfiguration**" per community or collection or the home page

Default settings

In addition to the summarized descriptions of the default values, following details help you to better understand these defaults. If you haven't already done so, download the configuration file and review it together with the following parameters[342].
The file contains one default configuration that defines following sidebar facets, search filters, sort fields and recent submissions display:

- Sidebar facets
  - **searchFilterAuthor:** groups the metadata fields dc.contributor.author & dc.creator with a facet limit of 10, sorted by occurrence count
  - **searchFilterSubject:** groups all subject metadata fields (dc.subject.*) with a facet limit of 10, sorted by occurrence count
  - **searchFilterIssued:** contains the dc.date.issued metadata field, which is identified with the type "date" and sorted by specific date values
- Search filters
  - **searchFilterTitle:** contains the dc.title metadata field
  - **searchFilterAuthor:** contains the dc.contributor.author & dc.creator metadata fields
  - **searchFilterSubject:** contains the dc.subject.* metadata fields
  - **searchFilterIssued:** contains the dc.date.issued metadata field with the type "date"
- Sort fields
  - **sortTitle**: contains the dc.title metadata field
  - **sortDateIssued:** contains the dc.date.issued metadata field, this sort has the type date configured.
- defaultFilterQueries
  - The default configuration contains no defaultFilterQueries
  - The default filter queries are disabled by default but there is an example in the default configuration in comments which allows discovery to only return items (as opposed to also communities/ collections).
- Recent Submissions
  - The recent submissions are sorted by dc.date. accessioned which is a date and a maximum number of 5 recent submissions are displayed.
- Hit highlighting
  - The fields dc.title, dc.contributor.author & dc.subject can contain hit highlighting.
  - The dc.description.abstract & full text field are used to render search snippets.
- Non indexed metadata fields
  - **Community/Collections:** dc.rights (copyright text)
  - **Items:** dc.description.provenance

342 https://wiki.lyrasis.org/download/attachments/104566682/discovery.xml?api=v2&modificationDate=1540322317159&version=1

Many of the properties contain lists that use references to point to the configuration elements. This way a certain configuration type can be used in multiple discovery configurations so there is no need to duplicate them.

Non indexed metadata fields

The discovery.xml file has configuration to not index certain metadata fields for communities/collections/items. The configuration is handled in the "toIgnoreMetadataFields" property located in the "org.dspace.discovery.configuration.DiscoveryConfigurationService" bean. Below is an example configuration that excludes dc.description.provenance for items & dc.rights for communities/collections:

```
<property name="toIgnoreMetadataFields">
    <map>
        <entry>
            <key><util:constant static-field="org.dspace.core.Constants.COMMUNITY"/></key>
            <list>
                <!--Introduction text-->
                <!--<value>dc.description</value>-->
                <!--Short description-->
                <!--<value>dc.description.abstract</value>-->
                <!--News-->
                <!--<value>dc.description.tableofcontents</value>-->
                <!--Copyright text-->
                <value>dc.rights</value>
                <!--Community name-->
                <!--<value>dc.title</value>-->
            </list>
        </entry>
        <entry>
            <key><util:constant static-field="org.dspace.core.Constants.COLLECTION"/></key>
            <list>
                <!--Introduction text-->
                <!--<value>dc.description</value>-->
                <!--Short description-->
                <!--<value>dc.description.abstract</value>-->
                <!--News-->
                <!--<value>dc.description.tableofcontents</value>-->
                <!--Copyright text-->
                <value>dc.rights</value>
                <!--Collection name-->
                <!--<value>dc.title</value>-->
            </list>
        </entry>
        <entry>
            <key><util:constant static-field="org.dspace.core.Constants.ITEM"/></key>
            <list>
                <value>dc.description.provenance</value>
            </list>
        </entry>
    </map>
</property>
```

By adding additional values to the appropriate lists additional metadata can be excluded from the search core, a reindex is required after altering this file to ensure that the values are removed from the index.

Search filters & sidebar facets Customization

This section explains the properties for search filters & sidebar facets. Each sidebar facet must occur in the reference list of the search filters. Below is an example configuration of a search filter that is not used as a sidebar facet.

```
<bean id="searchFilterTitle" class="org.dspace.discovery.configuration.DiscoverySearchFilter">
    <property name="indexFieldName" value="title"/>
    <property name="metadataFields">
        <list>
            <value>dc.title</value>
        </list>
    </property>
</bean>
```

The id & class attributes are mandatory for this type of bean. The properties that it contains are discussed below.

- **indexFieldName** (Required)**:** A unique search filter name, the metadata will be indexed in Solr under this field name.
- **metadataFields** (Required): A list of the metadata fields that need to be included in the facet.

Sidebar facets extend the search filter and add some extra properties to it, below is an example of a search filter that is also used as a sidebar facet.

```
<bean id="searchFilterAuthor" class="org.dspace.discovery.configuration.SidebarFacetConfiguration">
        <property name="indexFieldName" value="author"/>
        <property name="metadataFields">
            <list>
                <value>dc.contributor.author</value>
                <value>dc.creator</value>
            </list>
        </property>
        <property name="facetLimit" value="10"/>
        <property name="sortOrder" value="COUNT"/>
        <property name="type" value="text"/>
    </bean>
```

Note that the class has changed from **DiscoverySearchFilter** to **SidebarFacetConfiguration** this is needed to support the extra properties.

- **facetLimit** (optional)**:** The maximum number of values to be shown. This property is optional, if none is specified the default value "10" will be used. If the filter has the type **date**, this property will not be used since dates are automatically grouped together.
- **sortOrder** (optional)**:**The sort order for the sidebar facets, it can either be COUNT or VALUE. The default value is COUNT.
    - **COUNT** Facets will be sorted by the amount of times they appear in the repository
    - **VALUE** Facets will be sorted alphabetically
- **type**(optional): the type of the sidebar facet it can either be "date" or "text", "text" is the default value.
    - **text:** The facets will be treated as is
    - **date**: Only the year will be stored in the Solr index. These years are automatically displayed in ranges that get smaller when you select one.

Hierarchical (taxonomies based) sidebar facets

Discovery supports specialized drill down in hierarchically structured metadata fields. For this drill down to work, the metadata in the field for which you enable this must be composed out of terms, divided by a splitter. For example, you could have a dc.subject.taxonomy field in which you keep metadata like "CARTOGRAPHY::PHOTOGRAMMETRY", in which Cartography and Photogrammetry are both terms, divided by the splitter "::". The sidebar will only display the top level facets, when clicking on view more all the facet options will be displayed.

```
<bean id="searchFilterSubject" class="org.dspace.discovery.configuration.HierarchicalSidebarFacetConfigurat
ion">
    <property name="indexFieldName" value="subject"/>
    <property name="metadataFields">
        <list>
            <value>dc.subject</value>
        </list>
    </property>
    <property name="sortOrder" value="COUNT"/>
    <property name="splitter" value="::"/>
    <property name="skipFirstNodeLevel" value="false"/>
</bean>
```

Note that the class has changed from **SidebarFacetConfiguration** to **HierarchicalSidebarFacetConfiguration** this is needed to support the extra properties.

- **splitter** (required)**:** The splitter used to split up the separate nodes
- **skipFirstNodeLevel** (optional)**:** Whether or not to show the root node level. For some hierarchical data there is a single root node. In most cases it doesn't need to be shown since it isn't relevant. **This property is true by default.**

Sort option customization for search results

This section explains the properties of an individual SortConfiguration, like sortTitle and sortDateIssued from the default configuration. In order to create custom sort options, you can either modify specific properties of those that already exist or create a totally new one from scratch.

Here's what the sortTitle SortConfiguration looks like:

```
<bean id="sortTitle" class="org.dspace.discovery.configuration.DiscoverySortFieldConfiguration">
        <property name="metadataField" value="dc.title"/>
        <property name="type" value="text"/>
 </bean>
```

The id & class attributes are mandatory for this type of bean. The properties that it contains are discussed below.

- **metadataField** (Required): The metadata field indicating the sort values
- **type** (optional): the type of the sort option can either be date or text, if none is defined text will be used.

DiscoveryConfiguration

The DiscoveryConfiguration Groups configurations for sidebar facets, search filters, search sort options and recent submissions. If you want to show the same sidebar facets, use the same search filters, search options and recent

submissions everywhere in your repository, you will only need one DiscoveryConfiguration and you might as well just edit the defaultConfiguration.

The DiscoveryConfiguration makes it very easy to use custom sidebar facets, search filters, ... on specific communities or collection homepage. This is particularly useful if your collections are heterogeneous. For example, in a collection with conference papers, you might want to offer a sidebar facet for conference date, which might be more relevant than the actual issued date of the proceedings. In a collection with papers, you might want to offer a facet for funding bodies or publisher, while these fields are irrelevant for items like learning objects.

A DiscoveryConfiguration consists out of five parts

- The list of applicable sidebarFacets
- The list of applicable searchFilters
- The list of applicable searchSortFields
- Any default filter queries (optional)
- The configuration for the Recent submissions display
- The configuration of the tag cloud facet

Configuring lists of sidebarFacets and searchFilters

After modifying sidebarFacets and searchFilters, don't forget to reindex existing items by running `[dspace]/bin/dspace index-discovery -b`, otherwise the changes will not appear.

Below is an example of how one of these lists can be configured. It's important that each of the bean references corresponds to the exact name of the earlier defined facets, filters or sort options.

Each sidebar facet must also occur in the list of the search filters.

```
<property name="sidebarFacets">
    <list>
        <ref bean="sidebarFacetAuthor" />
        <ref bean="sidebarFacetSubject" />
        <ref bean="sidebarFacetDateIssued" />
    </list>
</property>
```

Configuring and customizing search sort fields

The search sort field configuration block contains the available sort fields and the possibility to configure a default sort field and sort order.
Below is an example of the sort configuration.

```
<property name="searchSortConfiguration">
    <bean class="org.dspace.discovery.configuration.DiscoverySortConfiguration">
        <!--<property name="defaultSort" ref="sortDateIssued"/>-->
        <!--DefaultSortOrder can either be desc or asc (desc is default)-->
        <property name="defaultSortOrder" value="desc"/>
        <property name="sortFields">
            <list>
                <ref bean="sortTitle" />
                <ref bean="sortDateIssued" />
            </list>
        </property>
    </bean>
</property>
```

The property name & the bean class are mandatory. The property field names are discusses below.

- **defaultSort** (optional): The default field on which the search results will be sorted, this must be a reference to an existing search sort field bean. If none is given relevance will be the default. Sorting according to the internal relevance algorithm is always available, even though it's not explicitly mentioned in the sortFields section.
- **defaultSortOrder** (optional): The default sort order can either be asc or desc.
- **sortFields** (mandatory): The list of available sort options, each element in this list must link to an existing sort field configuration bean.

Adding default filter queries (OPTIONAL)

Default filter queries are applied on all search operations & sidebarfacet clicks. One useful application of default filter queries is ensuring that all returned results are items. As a result, subcommunities and collections that are returned as results of the search operation, are filtered out.
Similar to the lists above, the default filter queries are defined as a list. They are optional.

```
<property name="defaultFilterQueries">
    <list>
        <value>query1</value>
        <value>query2</value>
    </list>
</property>
```

This property contains a simple list which in turn contains the queries. Some examples of possible queries:

- search.resourcetype:2
- dc.subject:test
- dc.contributor.author: "Van de Velde, Kevin"
- ...

Access Rights Awareness

By default, when searching and browsing using Discovery, you will only see items that you have access to. So, your search/browse results may differ if you are logged into DSpace. This Access Rights Awareness feature ensures that anonymous users (and search engines) are not able to access information (both files and metadata) about embargoed or private items. It also provides you with more direct control over who can see individual items within your DSpace.

How does Access Rights Awareness work?

Access Rights Awareness checks the "READ" access on the Item.

If the "Anonymous" group has "READ" access on the Item, then anonymous/public users will be able to view that Item's metadata and locate that Item via DSpace's search/browse system. In addition, search engines will also be able to index that Item's metadata. However, even with Anonymous READ set at the Item-level, you may still choose to access-restrict the downloading/viewing of *files* within the Item. To do so, you would restrict "READ" access on individual Bitstream(s) attached to the Item.

If the "Anonymous" group does NOT have "READ" access on the Item, then anonymous users will never see that Item appear within their search/browse results (essentially the Item is "invisible" to them). In addition, that Item will be invisible to search engines, so it will never be indexed by them. However, any users who have been given READ access will be able to find/locate the item after logging into DSpace. For example, if a "Staff" group was provided "READ" access on the Item, then members of that "Staff" group would be able to locate the item via search/browse after logging into DSpace.

How can I disable Access Rights Awareness?

If you prefer to allow all access-restricted or embargoed Items to be findable within your DSpace, you can choose to turn off Access Rights Awareness. However, please be aware that this means that restricting "READ" access on an Item will not really do anything – the Item metadata will be available to the public no matter what group(s) were given READ access on that Item.

This feature can be switched off by going to the `[dspace.dir]/config/spring/api/discovery.xml` file & commenting out the bean & the alias shown below.

```
<bean class="org.dspace.discovery.SolrServiceResourceRestrictionPlugin" id="solrServiceResourceIndexPlugin"
/>

<alias name="solrServiceResourceIndexPlugin"
alias="org.dspace.discovery.SolrServiceResourceRestrictionPlugin"/>
```

The Browse Engine only supports the "Access Rights Awareness" if the Solr/Discovery backend is enabled (see Defining the Storage of the Browse Data(see page 607)). However, it is enabled by default for DSpace 3.x and above.

Access Rights Awareness - technical details
The *DSpaceObject* class has an *updateLastModified()* method which will be triggered each time an authorization policy changes. This method is only implemented in the item class where the last_modified timestamp will be updated and a modify event will be fired. By doing this we ensure that the discovery consumer is called and the item is reindexed. Since this feature can be switched off a separate plugin has been created:
the *SolrServiceResourceRestrictionPlugin.* Whenever we reindex a DSpace object all the read rights will be stored in the read field. We make a distinction between groups and users by adding a '*g*' prefix for groups and the '*e*' prefix for epersons.

When searching in discovery all the groups the user belongs to will be added as a filter query as well as the users identifier. If the user is an admin all items will be returned since an admin has read rights on everything.

Customizing the Recent Submissions display

The recent submissions configuration element contains all the configuration settings to display the list of recently submitted items on the home page or community/collection page. Because the recent submission configuration is

in the discovery configuration block, it is possible to show 10 recently submitted items on the home page but 5 on the community/collection pages.

Below is an example configuration of the recent submissions.

```
<property name="recentSubmissionConfiguration">
    <bean class="org.dspace.discovery.configuration.DiscoveryRecentSubmissionsConfiguration">
        <property name="metadataSortField" value="dc.date.accessioned"/>
        <property name="type" value="date"/>
        <property name="max" value="5"/>
    </bean>
</property>
```

The property name & the bean class are mandatory. The property field names are discusses below.

- **metadataSortField** (mandatory): The metadata field to sort on to retrieve the recent submissions
- **max** (mandatory): The maximum number of results to be displayed as recent submissions
- **type** (optional): the type of the search filter. It can either be date or text, if none is defined text will be used.

Customizing hit highlighting & search snippets

The hit highlighting configuration element contains all settings necessary to display search snippets & enable hit highlighting.

Disabling hit highlighting / search snippets

You can disable hit highlighting / search snippets by commenting out the entire `<property name="hitHighlightingConfiguration">` Configuration in the `[dspace]/config/spring/api/discovery.xml` configuration file.

PLEASE BE AWARE there are two sections where this <property> definition exists. You should comment out both. One is under the `<bean id="defaultConfiguration">` and one is under the `<bean id="homepageConfiguration">`

Alternatively, you may also choose to tweak which fields are shown in hit highlighting, or modify the number of matching words shown (snippets) and/or number of characters shown around the matching word (maxSize).

For this change to take effect in the User Interface, you will need to restart Tomcat.

Changes made to the configuration will not automatically be displayed in the user interface. By default, only the following fields are displayed: dc.title, dc.contributor.author, dc.creator, dc.contributor, dc.date.issued, dc.publisher, dc.description.abstract and fulltext.

If additional fields are required, look for the "itemSummaryList" template.

Below is an example configuration of hit highlighting.

```
<property name="hitHighlightingConfiguration">
    <bean class="org.dspace.discovery.configuration.DiscoveryHitHighlightingConfiguration">
        <property name="metadataFields">
            <list>
                <bean class="org.dspace.discovery.configuration.DiscoveryHitHighlightFieldConfiguration">
                    <property name="field" value="dc.title"/>
                    <property name="snippets" value="5"/>
                </bean>
                <bean class="org.dspace.discovery.configuration.DiscoveryHitHighlightFieldConfiguration">
                    <property name="field" value="dc.contributor.author"/>
                    <property name="snippets" value="5"/>
                </bean>
                <bean class="org.dspace.discovery.configuration.DiscoveryHitHighlightFieldConfiguration">
                    <property name="field" value="dc.subject"/>
                    <property name="snippets" value="5"/>
                </bean>
                <bean class="org.dspace.discovery.configuration.DiscoveryHitHighlightFieldConfiguration">
                    <property name="field" value="dc.description.abstract"/>
                    <!-- Max number of characters to display around the matching word (Warning setting to 0
 returns entire field) -->
                    <property name="maxSize" value="250"/>
                    <!-- Max number of snippets (matching words) to show -->
                    <property name="snippets" value="2"/>
                </bean>
                <bean class="org.dspace.discovery.configuration.DiscoveryHitHighlightFieldConfiguration">
                    <!-- Displays snippets from indexed full text of document (for supported formats) -->
                    <property name="field" value="fulltext"/>
                    <!-- Max number of characters to display around the matching word (Warning setting to 0
 returns entire field) -->
                    <property name="maxSize" value="250"/>
                    <!-- Max number of snippets (matching words) to show -->
                    <property name="snippets" value="2"/>
                </bean>
            </list>
        </property>
    </bean>
</property>
```

The property name & the bean class are mandatory. The property field names are:

- **field** (mandatory)**:** The metadata field to be highlighted (can also be * if all the metadata fields should be highlighted).
- **maxSize** (optional): Limit the number of characters displayed to only the relevant part (use metadata field as search snippet).
- **snippets** (optional)**:** The maximum number of snippets that can be found in one metadata field.

Hit highlighting technical details
The *org.dspace.discovery.DiscoveryQuery* object has a setter & getter for the hit highlighting configuration set in Discovery configuration. If this configuration is given the *resolveToSolrQuery* method located in the *org.dspace.discovery.SolrServiceImpl* class will use the standard Solr highlighting feature (http://wiki.apache.org/solr/HighlightingParameters). The *org.dspace.discovery.DiscoverResult* class has a method to set the highlighted fields for each object & field.

The rendering of search results is no longer handled by the METS format but uses a special type of list named "TYPE_DSO_LIST". Each metadata field (& fulltext if configured) is added in the DRI and IF the field contains hit

higlighting the Java code will split up the string & add *DRI highlights* to the list. The XSL for the themes also contains special rendering XSL for the DRI; for Mirage, the changes are located in the *discovery.xsl* file. For themes using the old themes based on structural.xsl, look for the template matching "*dri:list[@type='dsolist']*".

"More like this" configuration

The "more like this"-configuration element contains all the settings for displaying related items on an item display page.
Below is an example of the "more like this" configuration.

```
<property name="moreLikeThisConfiguration">
    <bean class="org.dspace.discovery.configuration.DiscoveryMoreLikeThisConfiguration">
        <property name="similarityMetadataFields">
            <list>
                <value>dc.title</value>
                <value>dc.contributor.author</value>
                <value>dc.creator</value>
                <value>dc.subject</value>
            </list>
        </property>
        <!--The minimum number of matching terms across the metadata fields above before an item is found
as related -->
        <property name="minTermFrequency" value="5"/>
        <!--The maximum number of related items displayed-->
        <property name="max" value="3"/>
        <!--The minimum word length below which words will be ignored-->
        <property name="minWordLength" value="5"/>
    </bean>
</property>
```

The property name & the bean class are mandatory. The property field names are discussed below.

- similarityMetadataFields: the metadata fields checked for similarity
- minTermFrequency: The minimum number of matching terms accross the metadata fields above before an item is found as related
- max: The maximum number of related items displayed
- minWordLength: The minimum word length below which words will be ignored

"More like this" technical details
The *org.dspace.discovery.SearchService* object has received a *getRelatedItems()* method. This method requires an item & the more-like-this configuration bean from above. This method is implemented in the *org.dspace.discovery.SolrServiceImpl* which uses the item as a query & uses the default Solr parameters for more-like-this to pass the bean configuration to solr (https://cwiki.apache.org/confluence/display/solr/MoreLikeThis). The result will be a list of items or if none found an empty list.

"Did you mean" spellcheck aid for search configuration

DSpace 4 introduces the use of SOLR's SpellCheckComponent as an aid for search. When a user's search does not return any hits, the user is presented with a suggestion for an alternative search query.

The feature currently only one line of configuration to discovery.xml. Changing the value from true to false will disable the feature.

```
<property name="spellCheckEnabled" value="true" />
```

"Did you mean" spellcheck aid for search technical details
Similar to the More like this configuration, SOLR's spell check component is used with default configuration values. Any of these values can be overridden in the solrconfig.xml file located in dspace/solr/search/conf/. Following links provide more information about the SOLR SpellCheckComponent:

http://wiki.apache.org/solr/SpellCheckComponent

https://cwiki.apache.org/confluence/display/solr/Spell+Checking

Customizing the "Tag Cloud" facet

Not yet supported in DSpace 7.0

```
<!-- Set TagCloud configuration per discovery configuration -->
<property name="tagCloudFacetConfiguration" ref="defaultTagCloudFacetConfiguration"/>
```

*Declare the bean (of class: **TagCloudFacetConfiguration**) that holds the configuration for the tag cloud facet.*

```
<!--TagCloud configuration bean for homepage discovery configuration-->
<bean id="homepageTagCloudFacetConfiguration" class="org.dspace.discovery.configuration.TagCloudFacetConfig
uration">
        <!-- Actual configuration of the tagcloud (colors, sorting, etc.) -->
        <property name="tagCloudConfiguration" ref="tagCloudConfiguration"/>
        <!-- List of tagclouds to appear, one for every search filter, one after the other -->
        <property name="tagCloudFacets">
            <list>
                <ref bean="searchFilterSubject" />
            </list>
        </property>
</bean>
```

This bean has two properties:

- **tagCloudConfiguration**: is the bean which describes the actual appearance parameters
- **tagCloudFacets**: the search filter facets which will be used for the tag cloud. If you leave the list empty, no tag cloud will appear. If you declare more than one, such number of tag clouds will appear for each search filter, one after the other.

The appearance configuration can have the following properties, as shown in the following bean:

```
<bean id="tagCloudConfiguration" class="org.dspace.discovery.configuration.TagCloudConfiguration">
              <!-- Should display the score of each tag next to it? Default: false -->
              <property name="displayScore" value="true"/>
              <!-- Should display the tag as center aligned in the page or left aligned? Possible values:
true | false. Default: true  -->
              <property name="shouldCenter" value="true"/>
              <!-- How many tags will be shown. Value -1 means all of them. Default: -1 -->
              <property name="totalTags" value="-1"/>
              <!-- The letter case of the tags.
                      Possible values: Case.LOWER | Case.UPPER | Case.CAPITALIZATION | Case.PRESERVE_CASE |
Case.CASE_SENSITIVE
                      Default: Case.PRESERVE_CASE -->
              <property name="cloudCase" value="Case.PRESERVE_CASE"/>
              <!-- If the 3 CSS classes of the tag cloud should be independent of score (random=yes) or
based on the score. Possible values: true | false . Default: true-->
              <property name="randomColors" value="true"/>
              <!-- The font size (in em) for the tag with the lowest score. Possible values: any decimal.
Default: 1.1 -->
              <property name="fontFrom" value="1.1"/>
              <!-- The font size (in em) for the tag with the lowest score. Possible values: any decimal.
Default: 3.2 -->
              <property name="fontTo" value="3.2"/>
              <!-- The score that tags with lower than that will not appear in the rag cloud. Possible
values: any integer from 1 to infinity. Default: 0 -->
              <property name="cuttingLevel" value="0"/>
              <!-- The distance (in px) between the tags. Default: 5 -->
              <property name="marginRight" value="5"/>
              <!-- The ordering of the tags (based either on the name or the score of the tag)
                      Possible values: Tag.NameComparatorAsc | Tag.NameComparatorDesc |
Tag.ScoreComparatorAsc | Tag.ScoreComparatorDesc
                      Default: Tag.NameComparatorAsc  -->
              <property name="ordering" value="Tag.NameComparatorAsc"/>
    </bean>
```

When tagCloud is rendered there are some CSS classes that you can change in order to change the appearance of the tag cloud.

Class	Note
tagcloud	General class for the whole tagcloud
tagcloud_1	Specific tag class for tag of type 1 (based on score)
tagcloud_2	Specific tag class for tag of type 2 (based on score)

Class	Note
tagcloud_3	Specific tag class for tag of type 3 (based on score)

Disabling the "Has file(s)" facet

Since DSpace 6, a new "Has file(s)" facet has been enabled by default. This facet shows whether items have or do not have any bitstreams in the "ORIGINAL" bundle.
Should you want to turn this off, you can edit `[dspace]/config/spring/api/discovery.xml` to remove the following line from the `defaultConfiguration` and `homepageConfiguration` beans (in the `sidebarFacets` property):

```
<ref bean="searchFilterContentInOriginalBundle"/>
```

Then restart your servlet container.

4.9.3.5 Discovery Solr Index Maintenance

Command used:	`[dspace]/bin/dspace index-discovery [-cbhf[r <item handle>]]`
Java class:	org.dspace.discovery.IndexClient
Arguments (short and long forms):	Description
	called without any options, will update/clean an existing index
`-b`	(re)build index, wiping out current one if it exists
`-c`	clean existing index removing any documents that no longer exist in the db
`-f`	if updating existing index, force each handle to be reindexed even if uptodate
`-h`	print this help message

`-i <object handle>`	Reindex an individual object (and any child objects). When run on an Item, it just reindexes that single Item. When run on a Collection, it reindexes the Collection itself and all Items in that Collection. When run on a Community, it reindexes the Community itself and all sub-Communities, contained Collections and contained Items.
`-o`	optimize search core
`-r <item handle>`	remove an Item, Collection or Community from index based on its handle
`-s`	Rebuild the spellchecker, can be combined with -b and -f.

It is recommended to run maintenance on the Discovery Solr index occasionally (from crontab or your system's scheduler), to prevent your servlet container from running out of memory:

```
[dspace]/bin/dspace index-discovery -o
```

(Since Solr 4, the underlying optimize operation has been discouraged as mostly unnecessary and renamed. See https://issues.apache.org/jira/browse/SOLR-3141).

4.9.3.6 Advanced Solr Configuration

Discovery is built as an application layer on top of the Solr open source enterprise search server. Therefore, Solr configuration can be applied to the Solr cores that are shipped with DSpace.
The DSpace Solr instance currently runs several cores (which means indexes in Solr parlance). The "statistics" core is for collection of DSpace usage events for statistical purposes (if you have been collecting statistics for multiple years, you may have chosen to use sharding(see page 351) and you will see one core per each year collected).
The "search" core is used by Discovery for for search and faceting, for displaying the collection/community hierarchy and item counts. The "authority" core is used by SolrAuthority(see page 318) to store information about authors, including their data imported from the ORCID registry.

```
solr
├── search
|   └── conf
|       ├── protwords.txt
|       ├── schema.xml
|       ├── solrconfig.xml
|       ├── stopwords.txt
|       ├── synonyms.txt
|
├── ...
└── statistics
    └── conf
        ├── protwords.txt
        ├── schema.xml
        ├── solrconfig.xml
        ├── stopwords.txt
        ├── synonyms.txt
```

4.9.4 Multilingual Support

DSpace supports a number of languages & you can even add your own translation. This may also be referred to as Localization (l10n) or Internationalization (i18n).

4.9.4.1 Multilingual Support on the Backend (REST API)

In order to deploy a multilingual version of DSpace you have to configure two parameters in *[dspace-source]/dspace/config/local.cfg*:

- *default.locale*, e.g. `default.locale = en`
- *webui.supported locales*, e.g. `webui.supported.locales = en, de`

The Locales might have the form country, country_language, country_language_variant.

According to the languages you wish to support, you have to make sure that all the i18n related files are available.

Where to find the message catalog

The latest **English** message catalog is part of the main DSpace distribution and can be found at: `[dspace-source]/dspace-api/src/main/resources/Messages.properties`

The **different translations** for this message catalog are being managed separately from the DSpace core project, in order to release updates for these files more frequently than the DSpace software itself. Visit the dspace-api-lang project on Github[343].

Where to edit the message catalog

In some cases you may want to add additional keys to the message catalog or changing the particular wording of DSpace concepts. For example, you may want to change "Communities" into "Departments". These kind of changes may get automatically overwritten again when you upgrade to the newest version of DSpace. It is therefore advised to keep such changes isolated in the following location: `[dspace-source]/dspace/modules/server/src/main/resources/Messages.properties`

After rebuilding DSpace, any messages files placed in this directory will be automatically included in the Server web application. Files of the same name will override any default files. By default, this full directory path may not exist or may be empty. If it does not exist, you can simply create it. You can place any number of translation catalogues in this directory. To add additional translations, just add another copy of the *Messages.properties* file translated into the specific language and country variant you need.

343 https://github.com/DSpace/dspace-api-lang

For more information about the `[dspace-source]/dspace/modules/` directory, and how it may be used to "overlay" (or customize) the default Server Webapp, classes and files, please see: Advanced Customisation(see page 499)

Localization of email messages

All email templates used by DSpace can be found in `[dspace]/config/emails/`

The contents of the emails can be edited and translated.

Metadata localization

DSpace associates each metadata field value with a language code (though it may be left empty, e.g. for numeric values).

Localization of input-forms.xml

The display labels for input-forms.xml are currently not managed in the messages catalogs. To localize this file, you can create versions of this file in the same folders, appending _COUNTRY at the end of the filename, before the extension. For example, input-forms_de.xml can be used to translate the submission form labels in German.

Localization of license.default

The text in the default submission license (license.default) is currently not managed in the messages catalogs. It is translatable by appending _COUNTRY at the end of the filename, before the extension like for the localization of the input-forms.xml.

4.9.4.2 Multilingual Support on the Frontend (UI)

By default, DSpace will look at the user's browser language. If it has a language file in the user's language, it will render the interface in that language. If not, it will default to English or another default that you have configured.

The User Interface translations can be found in the `/src/assets/i18n/` folder of your UI's codebase. You can add additional translations & contribute them back to the project. For details see DSpace 7 Translation - Internationalization (i18n) - Localization (l10n)[344]

All translations of the UI are provided in JSON5[345] format, which includes support for inline comments.

You can choose which languages you wish to enable/support in your UI by modifying this section of your `environment.prod.ts` configuration file:

344 https://wiki.lyrasis.org/pages/viewpage.action?pageId=117735441
345 https://json5.org/

```
  // Default Language in which the UI will be rendered if the user's browser language is not an active
language
  defaultLanguage: 'en',
  // Languages. DSpace Angular holds a message catalog for each of the following languages.
  // When set to active, users will be able to switch to the use of this language in the user interface.
  languages: [{
    code: 'en',
    label: 'English',
    active: true,
  }, {
    code: 'de',
    label: 'Deutsch',
    active: true,
  }, {
    code: 'cs',
    label: 'Čeština',
    active: true,
  }, {
    code: 'nl',
    label: 'Nederlands',
    active: true,
  }],
```

As shown above, the "defaultLanguage" is the language that your UI will use *by default* , if the user's browser has not specified a preferred language

The array of "languages" are all of the additional languages you wish to support.

- The "code" must match the prefix of a `*.json5` language file located in your `/src/assets/i18n/` folder
- The "label" is the text you want to display in the UI language selector (the globe in the header)
- The "active" setting allows you to decide whether that language appears in the UI language selector or not.

Any changes to the language settings require rebuilding & redeploying your UI.

5 System Administration

This top level node intends to hold all system administration aspects of DSpace including but not limited to:

- Installation
- Upgrading
- Troubleshooting system errors
- Managing Dependencies

In this context System administration is defined as all technical tasks required to get DSpace in a state in which it operates properly so its behaviour is predictable and can be used according to all the guidelines under "Using DSpace".

5.1 Introduction to DSpace System Administration

DSpace operates on several levels: as a Java servlet (in a servlet container like Tomcat), cron jobs, and on-demand operations. This section explains many of the on-demand operations. Some of the command operations may be also set up as cron jobs. Many of these operations are performed at the Command Line Interface (CLI) also known as the Unix prompt ($). Future references will use the term CLI when a command needs to be run at the command line.

Below is the "Command Help Table". This table explains what data is contained in the individual command/help tables in the sections that follow.

Command used:	*The directory and where the command is to be found.*
Java class:	*The actual java program doing the work.*
Arguments:	*The required/mandatory or optional arguments available to the user.*

DSpace Command Launcher

Many/most commands and scripts have a simple `[dspace]/bin/dspace <command>` command. See the Application Layer chapter for the details of the DSpace Command Launcher(see page 0), and the Command Line Operations(see page 460) guide for common commands.

5.2 AIP Backup and Restore

5.2.1 Background & Overview

AIP Backup & Restore functionality only works with the Latest Version of Items

If you are using the Item Level Versioning(see page 307) functionality (disabled by default), you must be aware that this "Item Level Versioning" feature is **not yet compatible** with AIP Backup & Restore. **Using them together may result in accidental data loss.** Currently the AIPs that DSpace generates only store the *latest version* of an Item. Therefore, past versions of Items will always be lost when you perform a restore / replace using AIP tools.

Additional background information available in the Open Repositories 2010 Presentation entitled Improving DSpace Backups, Restores & Migrations[346]

DSpace can backup and restore all of its contents as a set of AIP Files(see page 439). This includes all Communities, Collections, Items, Groups and People in the system.

This feature came out of a requirement for DSpace to better integrate with DuraCloud[347], and other backup storage systems. One of these requirements is to be able to essentially "backup" local DSpace contents into the cloud (as a type of offsite backup), and "restore" those contents at a later time.

Essentially, this means DSpace can export the entire hierarchy (i.e. bitstreams, metadata and relationships between Communities/Collections/Items) into a relatively standard format (a METS-based, AIP format(see page 439)). This entire hierarchy can also be re-imported into DSpace in the same format (essentially a restore of that content in the same or different DSpace installation).

Benefits for the DSpace community:

- Allows one to more easily move entire Communities or Collections between DSpace instances.
- Allows for a potentially more consistent backup of this hierarchy (e.g. to DuraCloud, or just to your own local backup system), rather than relying on synchronizing a backup of your Database (stores metadata/ relationships) and assetstore (stores files/bitstreams).
- Provides a way for people to more easily get their data out of DSpace (whatever the purpose may be).
- Provides a relatively standard format for people to migrate entire hierarchies (Communities/Collections) from one DSpace to another (or from another system into DSpace).

5.2.1.1 How does this differ from traditional DSpace Backups? Which Backup route is better?

Traditionally, it has always been recommended to backup and restore DSpace's database and files (also known as the "assetstore") separately. This is described in more detail in the Storage Layer(see page 686) section of the DSpace System Documentation. The traditional backup and restore route is still a recommended and supported option.

However, the new AIP Backup & Restore option seeks to try and resolve many of the complexities of a traditional backup and restore. The below table details some of the differences between these two valid Backup and Restore options.

	Traditional Backup & Restore (Database and Files)	AIP Backup & Restore

346 http://www.slideshare.net/tdonohue/improving-dspace-backups-restores-migrations
347 http://www.duracloud.org

Supported Backup/Restore Types		
Can Backup & Restore all DSpace Content easily	Yes (Requires two backups/restores – one for Database and one for Files)	Yes (Though, will not backup/restore items which are not officially "in archive")
Can Backup & Restore a Single Community/Collection/Item easily	No (It is possible, but requires a strong understanding of DSpace database structure & folder organization in order to only backup & restore metadata/files belonging to that single object)	Yes
Backups can be used to move one or more Community/ Collection/Items to another DSpace system easily.	No (Again, it is possible, but requires a strong understanding of DSpace database structure & folder organization in order to only move metadata/files belonging to that object)	Yes
Can Backup & Restore Item Versions(see page 307)	Yes (Requires two backups/restores – one for Database and one for Files)	**No** (Currently Item Level Versioning(see page 307) is not fully compatible with AIP Backup & Restore. AIP Backup & Restore can only backup/restore the *latest version* of an Item)
Supported Object Types During Backup & Restore		
Supports backup/restore of all Communities/Collections/Items (including metadata, files, logos, etc.)	Yes	Yes
Supports backup/restore of all People/Groups/Permissions	Yes	Yes
Supports backup/restore of all Collection-specific Item Templates	Yes	Yes

Supports backup/restore of all Collection Harvesting settings (only for Collections which pull in all Items via OAI-PMH or OAI-ORE)	Yes	No (This is a known issue. All previously harvested Items will be restored, but the OAI-PMH/OAI-ORE harvesting settings will be lost during the restore process.)
Supports backup/restore of all Withdrawn (but not deleted) Items	Yes	Yes
Supports backup/restore of Item Mappings between Collections	Yes	Yes (During restore, the AIP Ingester may throw a false "Could not find a parent DSpaceObject" error (see Common Issues or Error Messages(see page 438)), if it tries to restore an Item Mapping to a Collection that it hasn't yet restored. But this error can be safely bypassed using the 'skipIfParentMissing' flag (see Additional Packager Options(see page 429) for more details).
Supports backup/restore of all in-process, uncompleted Submissions (or those currently in an approval workflow)	Yes	No (AIPs are only generated for objects which are completed and considered "in archive")
Supports backup/restore of Items using custom Metadata Schemas & Fields	Yes	Yes (Custom Metadata Fields will be automatically recreated. Custom Metadata Schemas must be manually created first, in order for DSpace to be able to recreate custom fields belonging to that schema. See Common Issues or Error Messages(see page 438) for more details.)
Supports backup/restore of all local DSpace Configurations and Customizations	Yes (if you backup your **entire** DSpace directory as part of backing up your files)	Not by default (unless you also backup parts of your DSpace directory – note, you wouldn't need to backup the '[dspace]/assetstore' folder again, as those files are already included in AIPs)

Based on your local institutions needs, you will want to choose the backup & restore process which is most appropriate to you. You may also find it beneficial to use both types of backups on different time schedules, in order to keep to a minimum the likelihood of losing your DSpace installation settings or its contents. For example, you

may choose to perform a Traditional Backup once per week (to backup your local system configurations and customizations) and an AIP Backup on a daily basis. Alternatively, you may choose to perform daily Traditional Backups and only use the AIP Backup as a "permanent archives" option (perhaps performed on a weekly or monthly basis).

Don't Forget to Backup your Configurations and Customizations

If you choose to use the AIP Backup and Restore option, do not forget to also backup your local DSpace configurations and customizations. Depending on how you manage your own local DSpace, these configurations and customizations are likely in one or more of the following locations:

- `[dspace]` - The DSpace installation directory (Please note, if you also use the AIP Backup & Restore option, you do **not** need to backup your `[dspace]/assetstore` directory, as those files already exist in your AIPs).
- `[dspace-source]` - The DSpace source directory

5.2.1.2 How does this help backup your DSpace to remote storage or cloud services (like DuraCloud)?

While AIP Backup and Restore is primarily a way to export your DSpace content objects to a local filesystem (or mounted drive), it can also be used as the basis for ensuring your content is safely backed up in a remote location (e.g. DuraCloud[348] or other cloud backup services).

Simply put, these AIPs can be generated and then replicated off to remote storage or a cloud backup service for safe keeping. You can then pull them down either as an entire set, or individually, in order to restore one or more objects into your DSpace instance. While you could simply backup your entire DSpace database and "assetstore" to a cloud service, you'd have to download the **entire** database backup again in order to restore any content. With AIPs, you can instead just download the individual AIP files you need (which can decrease your I/O costs, if any exist) for that restoration.

This upload/download of your AIPs to a backup location can be managed in a manual fashion (e.g. via your own custom code or shell scripts), or you can use a DSpace Replication Task Suite[349] add-on to help ease this process

The Replication Task Suite add-on for DSpace allows you the ability to backup and restore DSpace contents to/from AIPs via the DSpace Administrative Web Interface. It also includes "connectors" to the DuraCloud[350] API, so you can configure it to automatically backup/retrieve your AIPs to/from DuraCloud. Installing this add-on means you can now easily backup and restore DSpace to DuraCloud (or other systems) simply via the DSpace Administrative Web Interface. More information on installing and configuring this add-on can be found on the Replication Task Suite[351] page.

Makeup and Definition of AIPs

5.2.1.3 AIPs are Archival Information Packages

- AIP is a package describing **one archival object** in DSpace.

348 http://www.duracloud.org/
349 https://wiki.lyrasis.org/display/DSPACE/ReplicationTaskSuite
350 http://www.duracloud.org/
351 https://wiki.lyrasis.org/display/DSPACE/ReplicationTaskSuite

- The **archival object** may be a single **Item**, **Collection**, **Community**, or **Site** (Site AIPs contain site-wide information). Bitstreams are included in an Item's AIP.
- Each AIP is logically self-contained, can be restored without rest of the archive. (So you could restore a single Item, Collection or Community)
- Collection or Community AIPs do **not** include all child objects (e.g. Items in those Collections or Communities), as each AIP only describes **one** object. However, these container AIPs do contain references (links) to all child objects. These references can be used by DSpace to automatically restore all referenced AIPs when restoring a Collection or Community.
- AIPs are only generated for objects which are currently in the "in archive" state in DSpace. This means that in-progress, uncompleted submissions are not described in AIPs and cannot be restored after a disaster. Permanently removed objects will also no longer be exported as AIPs after their removal. However, withdrawn objects will continue to be exported as AIPs, since they are still considered under the "in archive" status.
- AIPs with identical contents will always have identical checksums[352]. This provides a basic means of validating whether the contents within an AIP have changed. For example, if a Collection's AIP has the same checksum at two different points in time, it means that Collection has not changed during that time period.
- AIP profile favors completeness and accuracy rather than presenting the semantics of an object in a standard format. It conforms to the quirks of DSpace's internal object model rather than attempting to produce a universally understandable representation of the object. When possible, an AIP tries to use common standards to express objects.
- An AIP *can* serve as a DIP (Dissemination Information Package) or SIP (Submission Information Package), especially when transferring custody of objects to another DSpace implementation.
- In contrast to SIP or DIP, the AIP should include all available DSpace structural and administrative metadata, and basic provenance information. AIPs also describe some basic system level information (e.g. Groups and People).

5.2.1.4 AIP Structure / Format

Generally speaking, an AIP is an Zip file containing a METS manifest and all related content bitstreams.

For more specific details of AIP format / structure, along with examples, please see DSpace AIP Format(see page 439).

5.2.2 Running the Code

5.2.2.1 Exporting AIPs

Export Modes & Options

All AIP Exports are done by using the Dissemination Mode (`-d` option) of the `packager` command.

There are two types of AIP Dissemination you can perform:

- Single AIP(see page 417) (default, using `-d` option) - Exports just an AIP describing a single DSpace object. So, if you ran it in this default mode for a Collection, you'd just end up with a single Collection AIP (which would not include AIPs for all its child Items)
- Hierarchy of AIPs(see page 417) (using the `-d --all` or `-d -a`option) - Exports the requested AIP describing an object, plus the AIP for all child objects. Some examples follow:

352 http://en.wikipedia.org/wiki/Checksum

- For a Site - this would export **all** Communities, Collections & Items within the site into AIP files (in a provided directory)
- For a Community - this would export that Community and all SubCommunities, Collections and Items into AIP files (in a provided directory)
- For a Collection - this would export that Collection and all contained Items into AIP files (in a provided directory)
- For an Item – this just exports the Item into an AIP as normal (as it already contains its Bitstreams/ Bundles by default)

Exporting just a single AIP

To export in single AIP mode (default), use this "packager" command template:

```
[dspace]/bin/dspace packager -d -t AIP -e <eperson> -i <handle> <file-path>
```

for example:

```
[dspace]/bin/dspace packager -d -t AIP -e admin@myu.edu -i 4321/4567 aip4567.zip
```

The above code will export the object of the given handle (4321/4567) into an AIP file named "aip4567.zip". This will **not** include any child objects for Communities or Collections.

Exporting AIP Hierarchy

To export an AIP hierarchy, use the `-a` (or `--all`) package parameter.

For example, use this 'packager' command template:

```
[dspace]/bin/dspace packager -d -a -t AIP -e <eperson> -i <handle> <file-path>
```

for example:

```
[dspace]/bin/dspace packager -d -a -t AIP -e admin@myu.edu -i 4321/4567 aip4567.zip
```

The above code will export the object of the given handle (4321/4567) into an AIP file named "aip4567.zip". In addition it would export all children objects to the same directory as the "aip4567.zip" file. The child AIP files are all named using the following format:

- File Name Format: `<Obj-Type>@<Handle-with-dashes>.zip`
  - e.g. COMMUNITY@123456789-1.zip, COLLECTION@123456789-2.zip, ITEM@123456789-200.zip
  - This general file naming convention ensures that you can easily locate an object to restore by its name (assuming you know its Object Type and Handle).
- Alternatively, if object doesn't have a Handle, it uses this File Name Format: `<Obj-Type>@internal-id-<DSpace-ID>.zip` (e.g. ITEM@internal-id-234.zip)

AIPs are only generated for objects which are currently in the "in archive" state in DSpace. This means that in-progress, uncompleted submissions are not described in AIPs and cannot be restored after a disaster.

Exporting Entire Site

To export an entire DSpace Site, pass the packager the Handle `<site-handle-prefix>/0`. For example, if your site prefix is "4321", you'd run a command similar to the following:

```
[dspace]/bin/dspace packager -d -a -t AIP -e admin@myu.edu -i 4321/0 sitewide-aip.zip
```

Again, this would export the DSpace Site AIP into the file "sitewide-aip.zip", and export AIPs for **all** Communities, Collections and Items into the same directory as the Site AIP.

5.2.2.2 Ingesting / Restoring AIPs

Ingestion Modes & Options

Ingestion of AIPs is a bit more complex than Dissemination, as there are several different "modes" available:

1. Submit/Ingest Mode(see page 419) (`-s` option, default) – submit AIP(s) to DSpace in order to create a new object(s) (i.e. AIP is treated like a SIP – Submission Information Package)
2. Restore Mode(see page 422) (`-r` option) – restore pre-existing object(s) in DSpace based on AIP(s). This also attempts to restore all handles and relationships (parent/child objects). This is a specialized type of "submit", where the object is created with a known Handle and known relationships.
3. Replace Mode(see page 424) (`-r -f` option) – replace existing object(s) in DSpace based on AIP(s). This also attempts to restore all handles and relationships (parent/child objects). This is a specialized type of "restore" where the contents of existing object(s) is replaced by the contents in the AIP(s). By default, if a normal "restore" finds the object already exists, it will back out (i.e. rollback all changes) and report which object already exists.

Again, like export, there are two types of AIP Ingestion you can perform (using any of the above modes):

- **Single AIP** (default) - Ingests just an AIP describing a single DSpace object. So, if you ran it in this default mode for a Collection AIP, you'd just create a DSpace Collection from the AIP (but not ingest any of its child objects)
- **Hierarchy of AIPs** (by including the `--all` or `-a`option after the mode) - Ingests the requested AIP describing an object, plus the AIP for all child objects. Some examples follow:
    - For a Site - this would ingest **all** Communities, Collections & Items based on the located AIP files
    - For a Community - this would ingest that Community and all SubCommunities, Collections and Items based on the located AIP files
    - For a Collection - this would ingest that Collection and all contained Items based on the located AIP files
    - For an Item – this just ingest the Item (including all Bitstreams & Bundles) based on the AIP file.

The difference between "Submit" and "Restore/Replace" modes

It's worth understanding the primary differences between a Submission (specified by `-s` parameter) and a Restore (specified by `-r` parameter).

- Submission Mode(see page 419) (`-s` mode) - creates a new object (AIP is treated like a SIP)
    - By default, a new Handle is always assigned
        - However, you can force it to use the handle specified in the AIP by specifying `-o ignoreHandle=false` as one of your parameters
    - By default, a new Parent object **must** be specified (using the `-p` parameter). This is the location where the new object will be created.

        - However, you can force it to use the parent object specified in the AIP by specifying `-o ignoreParent=false` as one of your parameters
    - By default, will respect a Collection's Workflow process when you submit an Item to a Collection
        - However, you can specifically *skip* any workflow approval processes by specifying `-w` parameter.
    - **Always** adds a new Deposit License to Items
    - **Always** adds new DSpace System metadata to Items (includes new "dc.date.accessioned", "dc.date.available", "dc.date.issued" and "dc.description.provenance" entries)
    - **WARNING:** Submission mode may not be able to maintain Item Mappings between Collections. Because these mappings are recorded via the Collection Handles, mappings may be restored improperly if the Collection handle has changed when moving content from one DSpace instance to another.
- Restore / Replace Mode(see page 422) (`-r` mode) - restores a previously existing object (as if from a backup)
    - By default, the Handle specified in the AIP is restored
        - However, for restores, you can force a new handle to be generated by specifying `-o ignoreHandle=true` as one of your parameters. (NOTE: Doesn't work for *replace* mode as the new object always retains the handle of the replaced object)
        - Although a Restore/Replace does restore Handles, it will not necessarily restore the same internal IDs in your Database.
    - By default, the object is restored under the Parent specified in the AIP
        - However, for restores, you can force it to restore under a different parent object by using the `-p` parameter. (NOTE: Doesn't work for *replace* mode, as the new object always retains the parent of the replaced object)
    - **Always** skips any Collection workflow approval processes when restoring/replacing an Item in a Collection
    - **Never** adds a new Deposit License to Items (rather it restores the previous deposit license, as long as it is stored in the AIP)
    - **Never** adds new DSpace System metadata to Items (rather it just restores the metadata as specified in the AIP)

Changing Submission/Restore Behavior

It is possible to change some of the default behaviors of both the Submission and Restore/Replace Modes. Please see the Additional Packager Options(see page 429) section below for a listing of command-line options that allow you to override some of the default settings described above.

Submitting AIP(s) to create a new object

The Submission mode (`-s`) always creates a new object with a newly assigned handle. In addition by default it respects all existing Collection approval workflows (so items may require approval unless the workflow is skipped by using the `-w` option). For information about how the "Submission Mode" differs from the "Replace / Restore mode", see The difference between "Submit" and "Restore/Replace" modes(see page 418) above.

Submitting a Single AIP

> **AIPs treated as SIPs**
>
> This option allows you to essentially use an AIP as a SIP (Submission Information Package). The default settings will create a new DSpace object (with a new handle and a new parent object, if specified) from your AIP.

To ingest a single AIP and create a new DSpace object under a parent of your choice, specify the `-p` (or `--parent`) package parameter to the command. Also, note that you are running the `packager` in `-s` (submit) mode.

NOTE: This only ingests the single AIP specified. It does **not** ingest all children objects.

```
[dspace]/bin/dspace packager -s -t AIP -e <eperson> -p <parent-handle> <file-path>
```

If you leave out the `-p` parameter, the AIP package ingester will attempt to install the AIP under the same parent it had before. As you are also specifying the `-s` (submit) parameter, the `packager` will assume you want a new Handle to be assigned (as you are effectively specifying that you are submitting a **new** object). If you want the object to retain the Handle specified in the AIP, you can specify the `-o ignoreHandle=false` option to force the packager to *not* ignore the Handle specified in the AIP.

Submitting an AIP Hierarchy

> **AIPs treated as SIPs**
>
> This option allows you to essentially use a set of AIPs as SIPs (Submission Information Packages). The default settings will create a new DSpace object (with a new handle and a new parent object, if specified) from each AIP

To ingest an AIP hierarchy from a directory of AIPs, use the `-a` (or `--all`) package parameter.

For example, use this 'packager' command template:

```
[dspace]/bin/dspace packager -s -a -t AIP -e <eperson> -p <parent-handle> <file-path>
```

for example:

```
[dspace]/bin/dspace packager -s -a -t AIP -e admin@myu.edu -p 4321/12 aip4567.zip
```

The above command will ingest the package named "aip4567.zip" as a child of the specified Parent Object (handle="4321/12"). The resulting object is assigned a new Handle (since `-s` is specified). In addition, any child AIPs referenced by "aip4567.zip" are also recursively ingested (a new Handle is also assigned for each child AIP).

Another example – **Ingesting a Top-Level Community** (by using the Site Handle, `<site-handle-prefix>/0`):

```
[dspace]/bin/dspace packager -s -a -t AIP -e admin@myu.edu -p 4321/0 community-aip.zip
```

The above command will ingest the package named "community-aip.zip" as a **top-level community** (i.e. the specified parent is "4321/0" which is a Site Handle). Again, the resulting object is assigned a new Handle. In addition, any child AIPs referenced by "community-aip.zip" are also recursively ingested (a new Handle is also assigned for each child AIP).

May want to skip Collection Approvals Workflows

Please note: If you are submitting a larger amount of content (e.g. multiple Communities/Collections) to your DSpace, you may want to tell the 'packager' command to skip over any existing Collection approval workflows by using the `-w` flag. By default, all Collection approval workflows will be respected. This means if the content you are submitting includes a Collection with an enabled workflow, you may see the following occur:

1. First, the Collection will be created & its workflow enabled
2. Second, each Item belonging to that Collection will be created & placed into the workflow approval process

   Therefore, if this content has already received some level of approval, you may want to submit it using the `-w` flag, which will skip any workflow approval processes. For more information, see Submitting AIP(s) while skipping any Collection Approval Workflows(see page 422).

Item Mappings may not be maintained when submitting an AIP hierachy

When an Item is mapped to one or more Collections, this mapping is recorded in the AIP using the mapped Collection's handle. Unfortunately, since the submission mode (-s) assigns **new handles** to all objects in the hierarchy, this may mean that the mapped Collection's handle will have changed (or even that a different Collection will be available at the original mapped Collection's handle). DSpace does not have a way to uniquely identify Collections other than by handle, which means that item mappings are only able to be retained when the Collection handle is *also retained.*

If you encounter this issue, there are a few possible workarounds:

1. Use the restore/replace mode (-r) instead, as it will retain existing Collection Handles. Unfortunately though, this may not work if the content is being moved from a Test DSpace to a Production DSpace, as these existing handles may not be valid.
2. OR, use the submission mode with the "--o ignoreHandle=false". This will also retain existing Collection Handles. Unfortunately though, this may not work if the content is being moved from a Test DSpace to a Production DSpace, as these existing handles may not be valid.
3. OR, remove all existing Item Mappings and re-export AIPs (without Item Mappings). Then, import the hierarchy into the new DSpace instance (again without Item Mappings). Finally, recreate the necessary Item Mappings using a different tool, e.g. the Batch Metadata Editing(see page 287) tool supports bulk editing of Collection memberships/mappings.

Missing Groups or EPeople cannot be created when submitting an individual Community or Collection AIP

Please note, if you are using AIPs to move an entire Community or Collection from one DSpace to another, there is a known issue (see DS-1105[353]) that the new DSpace instance will be unable to (re-)create any DSpace Groups or EPeople which are referenced by a Community or Collection AIP. The reason is that the Community or Collection AIP itself doesn't contain enough information to create those Groups or EPeople (rather that info is stored in the SITE AIP, for usage during Full Site Restores(see page 425)).

353 https://jira.duraspace.org/browse/DS-1105

However, there are two possible ways to get around this known issue:

- **EITHER**, you can manually recreate all referenced Groups/EPeople in the new DSpace that you are submitting the Community or Collection AIP into.
  - Note that if you are using Groups named with DSpace Database IDs (e.g. COMMUNITY_1_ADMIN, COLLECTION_2_SUBMIT), you may first need to rename those groups to no longer include Database IDs (e.g. MY_SUBMITTERS). The reason is that Database IDs will likely change when you move a Community or Collection to a new DSpace installation.
- **OR**, you can temporarily disable the import of Group/EPeople information when submitting the Community or Collection AIP to the new DSpace. This would mean that after you submit the AIP to the new DSpace, you'd have to manually go in and add in any special permissions (as needed). To disable the import of Group/EPeople information, add these settings to your `dspace.cfg`file, and re-run the submission of the AIP with these settings in place:

```
mets.dspaceAIP.ingest.crosswalk.METSRIGHTS = NIL
mets.dspaceAIP.ingest.crosswalk.DSPACE-ROLES = NIL
```

  - Don't forget to remove these settings after you import your Community or Collection AIP. *Leaving them in place will mean that every time you import an AIP, all of its Group/EPeople/ Permissions would be ignored.*

Submitting AIP(s) while skipping any Collection Approval Workflows

By default, the Submission mode (`-s`) always respects existing Colleciton approval workflows. So, if a Collection has a workflow, then a newly submitted Item will be placed into that workflow process (rather than immediately appearing in DSpace).

However, if you'd like to skip all workflow approval processes you can use the `-w` flag to do so. For example, the following command will skip any Collection approval workflows and immediately add the Item to a Collection.

```
[dspace]/bin/dspace packager -s -w -t AIP -e <eperson> -p <parent-handle> <file-path>
```

This `-w` flag may also be used when Submitting an AIP Hierarchy(see page 420). For example, if you are migrating one or more Collections/Communities from one DSpace to another, you may choose to submit those AIPs with the `-w` option enabled. This will ensure that, if a Collection has a workflow approval process enabled, all its Items are available immediately rather than being all placed into the workflow approval process.

Restoring/Replacing using AIP(s)

Restoring is slightly different than just **submitting**. When restoring, we make every attempt to restore the object as it **used to be** (including its handle, parent object, etc.). For more information about how the "Replace/Restore Mode" differs from the "Submit mode", see The difference between "Submit" and "Restore/Replace" modes(see page 418) above.

There are currently three restore modes:

1. Default Restore Mode(see page 423) (`-r`) = Attempt to restore object (and optionally children). Rollback all changes if any object is found to already exist.

2. Restore, Keep Existing Mode(see page 424) (`-r -k`) = Attempt to restore object (and optionally children). If an object is found to already exist, skip over it (and all children objects), and continue to restore all other non-existing objects.
3. Force Replace Mode(see page 424) (`-r -f`) = Restore an object (and optionally children) and **overwrite** any existing objects in DSpace. Therefore, if an object is found to already exist in DSpace, its contents are replaced by the contents of the AIP. *WARNING: This mode is potentially dangerous as it will permanently destroy any object contents that do not currently exist in the AIP. You may want to perform a secondary backup, unless you are sure you know what you are doing!*

Default Restore Mode

By default, the restore mode (`-r` option) will throw an error and rollback all changes if any object is found to already exist. The user will be informed if which object already exists within their DSpace installation.

Restore a Single AIP: Use this 'packager' command template to restore a single object from an AIP (not including any child objects):

```
[dspace]/bin/dspace packager -r -t AIP -e <eperson> <AIP-file-path>
```

Restore a Hierarchy of AIPs: Use this 'packager' command template to restore an object from an AIP along with all child objects (from their AIPs):

```
[dspace]/bin/dspace packager -r -a -t AIP -e <eperson> <AIP-file-path>
```

For example:

```
[dspace]/bin/dspace packager -r -a -t AIP -e admin@myu.edu aip4567.zip
```

Notice that unlike `-s` option (for submission/ingesting), the `-r` option does not require the Parent Object (`-p` option) to be specified if it can be determined from the package itself.

In the above example, the package "aip4567.zip" is restored to the DSpace installation with the Handle provided within the package itself (and added as a child of the parent object specified within the package itself). In addition, any child AIPs referenced by "aip4567.zip" are also recursively ingested (the `-a` option specifies to also restore all child AIPs). They are also restored with the Handles & Parent Objects provided with their package. If any object is found to already exist, all changes are rolled back (i.e. nothing is restored to DSpace)

Highly Recommended to Update Database Sequences after a Large Restore

In some cases, when you restore a large amount of content to your DSpace, the internal database counts (called "sequences") may get out of sync with the Handles of the content you just restored. As a best practice, it is **highly recommended to always** re-run the "update-sequences.sql" script on your DSpace database after a larger scale restore. This database script should be run while DSpace is stopped (you may either stop Tomcat or just the DSpace webapps). PostgreSQL/Oracle must be running. The script can be found in the following locations for PostgreSQL and Oracle, respectively:
`[dspace]/etc/postgres/update-sequences.sql`
`[dspace]/etc/oracle/update-sequences.sql`

More Information on using Default Restore Mode with Community/Collection AIPs

- Using the Default Restore Mode without the `-a` option, will only restore the **metadata** for that specific Community or Collection. No child objects will be restored.
- Using the Default Restore Mode with the `-a` option, will only successfully restore a Community or Collection if that object along with any child objects (Sub-Communities, Collections or Items) do not already exist. In other words, if any objects belonging to that Community or Collection already exist in DSpace, the Default Restore Mode will report an error that those object(s) could not be recreated. If you encounter this situation, you will need to perform the restore using either the Restore, Keep Existing Mode(see page 424) *or* the Force Replace Mode(see page 424) (depending on whether you want to keep or replace those existing child objects).

Restore, Keep Existing Mode

When the "Keep Existing" flag (`-k` option) is specified, the restore will attempt to skip over any objects found to already exist. It will report to the user that the object was found to exist (and was not modified or changed). It will then continue to restore all objects which do not already exist.

One special case to note: If a Collection or Community is found to already exist, its child objects are also skipped over. So, this mode will not auto-restore items to an existing Collection.

Restore a Hierarchy of AIPs: Use this 'packager' command template to restore an object from an AIP along with all child objects (from their AIPs):

```
[dspace]/bin/dspace packager -r -a -k -t AIP -e <eperson> <AIP-file-path>
```

For example:

```
[dspace]/bin/dspace packager -r -a -k -t AIP -e admin@myu.edu aip4567.zip
```

In the above example, the package "aip4567.zip" is restored to the DSpace installation with the Handle provided within the package itself (and added as a child of the parent object specified within the package itself). In addition, any child AIPs referenced by "aip4567.zip" are also recursively restored (the `-a` option specifies to also restore all child AIPs). They are also restored with the Handles & Parent Objects provided with their package. If any object is found to already exist, it is skipped over (child objects are also skipped). All non-existing objects are restored.

Force Replace Mode

When the "Force Replace" flag (`-f` option) is specified, the restore will **overwrite** any objects found to already exist in DSpace. In other words, existing content is deleted and then replaced by the contents of the AIP(s).

May also be useful in some specific restoration scenarios

This mode may also be used to restore missing objects which refer to existing objects. For example, if you are restoring a missing Collection which had existing Items linked to it, you can use this mode to auto-restore the Collection and update those existing Items so that they again link back to the newly restored Collection.

> **Potential for Data Loss**
>
> Because this mode actually **destroys** existing content in DSpace, it is potentially dangerous and may result in data loss! You may wish to perform a secondary full backup (assetstore files & database) before attempting to replace any existing object(s) in DSpace.

Replace using a Single AIP: Use this 'packager' command template to replace a single object from an AIP (not including any child objects):

```
[dspace]/bin/dspace packager -r -f -t AIP -e <eperson> <AIP-file-path>
```

Replace using a Hierarchy of AIPs: Use this 'packager' command template to replace an object from an AIP along with all child objects (from their AIPs):

```
[dspace]/bin/dspace packager -r -a -f -t AIP -e <eperson> <AIP-file-path>
```

For example:

```
[dspace]/bin/dspace packager -r -a -f -t AIP -e admin@myu.edu aip4567.zip
```

In the above example, the package "aip4567.zip" is restored to the DSpace installation with the Handle provided within the package itself (and added as a child of the parent object specified within the package itself). In addition, any child AIPs referenced by "aip4567.zip" are also recursively ingested. They are also restored with the Handles & Parent Objects provided with their package. *If any object is found to already exist, its contents are replaced by the contents of the appropriate AIP.*

If any error occurs, the script attempts to rollback the entire replacement process.

Restoring Entire Site

In order to restore an entire Site from a set of AIPs, you must do the following:

1. Install a completely "fresh" version of DSpace by following the Installation instructions in the DSpace Manual(see page 411)
    - At this point, you should have a completely empty, but fully-functional DSpace installation. You will need to create an initial Administrator user in order to perform this restore (as a full-restore can only be performed by a DSpace Administrator).
2. Once DSpace is installed, run the following command to restore all its contents from AIPs

    ```
    [dspace]/bin/dspace packager -r -a -f -t AIP -e <eperson> -i <site-handle-prefix>/0 -o
    skipIfParentMissing=true /full/path/to/your/site-aip.zip
    ```

    a. While the "`-o skipIfParentMissing=true`" flag is optional, it is often necessary whenever you are performing a large hierarchical site restoration. Please see the Additional Packager Options(see page 429) section below.

Please note the following about the above restore command:

- Notice that you are running this command in "Force Replace" mode (`-r -f`). This is necessary as your empty DSpace install will already include a few default groups (Administrators and Anonymous) and your initial administrative user. You need to replace these groups in order to restore your prior DSpace contents completely.
- `<eperson>` should be replaced with the Email Address of the initial Administrator (who you created when you reinstalled DSpace).
- `<site-handle-prefix>` should be replaced with your DSpace site's assigned Handle Prefix. This is equivalent to the `handle.prefix` setting in your `dspace.cfg`
- `/full/path/to/your/site-aip.zip` is the full path to the AIP file which represents your DSpace SITE. This file will be named whatever you named it when you actually exported your entire site(see page 418). All other AIPs are assumed to be referenced from this SITE AIP (in most cases, they should be in the same directory as that SITE AIP).

Highly Recommended to Update Database Sequences after a Large Restore

In some cases, when you restore a large amount of content to your DSpace, the internal database counts (called "sequences") may get out of sync with the Handles of the content you just restored. As a best practice, it is **highly recommended to always** re-run the "update-sequences.sql" script on your DSpace database after a larger scale restore. This database script should be run while DSpace is stopped (you may either stop Tomcat or just the DSpace webapps). PostgreSQL/Oracle must be running. The script can be found in the following locations for PostgreSQL and Oracle, respectively:
`[dspace]/etc/postgres/update-sequences.sql`
`[dspace]/etc/oracle/update-sequences.sql`

5.2.2.3 Cleaning up from a failed import

Sometimes your packager import of AIP packages can fail, due to lack of memory (see below for advice on better performance, please use JAVA_OPTS to set your memory higher than the default). If that happens, DSpace by design will leave the bitstreams it **did** import sucessfully, but they will be oprphaned, and will just occupy space in your assetstore. The standard DSpace cleanup cron job will clean up these orphaned bitstreams, however, you can also clean them up manually by running the following command:

Clean up after a failed import

```
[dspace]/bin/dspace cleanup -v
```

5.2.2.4 Performance considerations

When importing large structures like the whole site or a large collection/community, keep in mind that this can require a lot of memory, more than the default amount of heap allocated to the command-line launcher (256 Mb: `JAVA_OPTS="-Xmx256m -Dfile.encoding=UTF-8"`). This memory must be allocated in addition to the normal amount of memory allocated to Tomcat. For example, a site of 2500 fulltext items (2 Gb altogether) requires 5 Gb of maximum heap space and takes around 1 hour, including import and indexing.

You can raise the limit for a single run of the packager command by specifying memory options in the JAVA_OPTS environment variable, e.g.:

```
JAVA_OPTS="-Xmx4096m -Dfile.encoding=UTF-8" /dspace/bin/dspace packager -u -r -a -f -t AIP -e
dspace@example.com -i 123456789/0 sitewide-aip.zip
```

If the importer runs out of heap memory, it will crash either with "java.lang.OutOfMemoryError: GC overhead limit exceeded", which can be suppressed by adding "-XX:-UseGCOverheadLimit" to JAVA_OPTS, or with "java.lang.OutOfMemoryError: Java heap space". You can increase the allocated heap memory and try again, but keep in mind that although no changes were made in the database, the unsuccessfully imported files are still left in the assetstore (see DS-2227[354]).

5.2.2.5 Disable User Interaction for Cron

If you wish to run any of the following commands from a cron job (or similar), then you may wish to **disable all user interaction** using the `-u` (`--no-user-interaction`) flag. For example, supposing you wanted to perform a full Site Backup (see Exporting Entire Site(see page 418) above) via a cronjob, you could simply run that command passing it the "-u" flage like this:

```
# Perform a full site backup to AIPs(with user interaction disabled) every Sunday at 1:00AM
# NOTE: Make sure to replace "123456789" with your actual Handle Prefix, and "admin@myu.edu" with your
Administrator account email.
0 1 * * * [dspace]/bin/dspace packager -u -d -a -t AIP -e admin@myu.edu -i 123456789/0 [full-path-to-
backup-folder]/sitewide-aip.zip
```

5.2.3 Command Line Reference

The following flags are valid to pass to the `[dspace]/bin/dspace packager` command:

Flag	Ingest or Export	Description / Usage
`-a (--all)`	both ingest and export	*For Ingest*: recursively ingest all child AIPs (referenced from this AIP). *For Export*: recursively export all child objects (referenced from this parent object)
`-d (--disseminate)`	export-only	This flag simply triggers the export of AIPs from the system. See Exporting AIPs(see page 416)
`-e (-eperson) [email-address]`	ingest-only	The email address of the EPerson who is ingesting the AIPs. Oftentimes this should be an Administrative account.

354 https://jira.duraspace.org/browse/DS-2227

Flag	Ingest or Export	Description / Usage
`-f (--force-replace)`	ingest-only	Ingest the AIPs in "Force Replace Mode(see page 424)" (must be specified in conjunction with `-r` flag), where existing objects will be replaced by the contents of the AIP.
`-h (--help)`	both ingest and export	Return help information. You should specify with `-t` for additional package specific help information
`-i (--identifier) [handle]`	both ingest and export	*For Ingest*: Only valid in "Force Replace Mode(see page 424)". In that mode this is the identifier of the object to replace. *For Export:* The identifier of the object to export to an AIP
`-k (--keep-existing)`	ingest-only	Specifies to use "Restore, Keep Existing Mode(see page 424)" during ingest (must be specified in conjunction with `-r` flag). In this mode, existing objects in DSpace will NOT be replaced by their AIPs, but missing objects will be restored from AIPs.
`-o (--option) [setting]=[value]`	both ingest and export	This flag is used to pass Additional Packager Options(see page 429) to the Packager command. Each type of packager may define its own custom Additional Options. For AIPs, the valid options are documented in the Additional Packager Options(see page 429) section below. This is repeatable (e.g. `-o [setting1]=[value] -o [setting2]=value`)
`-p (--parent) [handle]`	ingest only	Handle(s) of the parent Community or Collection to into which an AIP should be ingested. This may be repeatable.
`-r (--restore)`	ingest only	Specifies that this ingest is either "Restore Mode(see page 423)" (when standalone), "Restore, Keep Existing Mode(see page 424)" (when used with `-k` flag) or "Force Replace Mode(see page 424)" (when used with `-f` flag)
`-s (--submit)`	ingest only	Specifies that this ingest is in "Submit Mode(see page 419)" where an AIP is treated as a **new** object and assigned a new Handle/Identifier, etc.

Flag	Ingest or Export	Description / Usage
`-t (--type) [package-type]`	both ingest and export	Specifies the type of package which is being ingested or exported. This controls which Ingester or Disseminator class is called. For AIPs, this is **always** set to "`-t AIP`"
`-u (--no-user-interaction)`	both ingest and export	Skips over all user interaction (e.g. question prompts). This flag can be used when running the packager from a script or cron job to bypass all user interaction. See also Disable User Interaction for Cron(see page 427)

5.2.3.1 Additional Packager Options

In additional to the various "modes" settings described under "Running the Code(see page 416)" above, the AIP Packager supports the following packager options. These options allow you to better tweak how your AIPs are processed (especially during ingests/restores/replaces).

Option	Ingest or Export	Default Value	Description
`createMetadataFields=[value]`	ingest-only	true	Tells the AIP ingester to automatically create any metadata fields which are found to be **missing** from the DSpace Metadata Registry. When 'true', this means as each AIP is ingested, new fields may be added to the DSpace Metadata Registry if they don't already exist. When 'false', an AIP ingest will fail if it encounters a metadata field that doesn't exist in the DSpace Metadata Registry. (NOTE: This will **not** create missing DSpace Metadata *Schemas*. If a schema is found to be missing, the ingest will always fail.)

Option	Ingest or Export	Default Value	Description
`filterBundles=[value]`	export-only	defaults to exporting all Bundles	This option can be used to limit the Bundles which are exported to AIPs for each DSpace Item. By default, all file Bundles will be exported into Item AIPs. You could use this option to limit the size of AIPs by only exporting certain Bundles. *WARNING: any bundles* **not** *included in AIPs will obviously be unable to be restored.* This option can be run in two ways: • **Exclude Bundles:** By default, you can provide a comma-separated list of bundles to be excluded from AIPs (e.g. "TEXT, THUMBNAIL") • **Include Bundles:** If you prepend the list with the "+" symbol, then the list specifies the bundles to be **included** in AIPs (e.g. "+ORIGINAL,LICENSE" would only include those two bundles). This second option is identical to using "includeBundles" option described below. (NOTE: If you choose to no longer export LICENSE or CC_LICENSE bundles, you will also need to disable the License Dissemination Crosswalks in the `aip.disseminate.rightsMD` configuration for the changes to take affect)
`ignoreHandle=[value]`	ingest-only	Restore/Replace Mode defaults to 'false', Submit Mode defaults to 'true'	If 'true', the AIP ingester will ignore any Handle specified in the AIP itself, and instead create a new Handle during the ingest process (this is the default when running in Submit mode, using the `-s` flag). If 'false', the AIP ingester attempts to restore the Handles specified in the AIP (this is the default when running in Restore/replace mode, using the `-r` flag).

Option	Ingest or Export	Default Value	Description
`ignoreParent=[value]`	ingest-only	Restore/Replace Mode defaults to 'false', Submit Mode defaults to 'true'	If 'true', the AIP ingester will ignore any Parent object specified in the AIP itself, and instead ingest under a new Parent object (this is the default when running in Submit mode, using the `-s` flag). The new Parent object must be specified via the `-p` flag (run `dspace packager -h` for more help). If 'false', the AIP ingester attempts to restore the object directly under its old Parent (this is the default when running in Restore/ replace mode, using the `-r` flag).
`includeBundles=[value]`	export-only	defaults to "all"	This option can be used to limit the Bundles which are exported to AIPs for each DSpace Item. By default, all file Bundles will be exported into Item AIPs. You could use this option to limit the size of AIPs by only exporting certain Bundles. *WARNING: any bundles **not** included in AIPs will obviously be unable to be restored.* This option expects a comma separated list of bundle names (e.g. "ORIGINAL,LICENSE,CC_LICENSE,METADATA"), or "all" if all bundles should be included. (See "filterBundles" option above if you wish to exclude particular Bundles. However, this "includeBundles" option **cannot** be used at the same time as "filterBundles".) (NOTE: If you choose to no longer export LICENSE or CC_LICENSE bundles, you will also need to disable the License Dissemination Crosswalks in the `aip.disseminate.rightsMD` configuration for the changes to take affect)

Option	Ingest or Export	Default Value	Description
`manifestOnly=[value]`	both ingest and export	false	If 'true', the AIP Disseminator will only import/export a METS Manifest XML file (i.e. result will be an unzipped 'mets.xml' file), instead of a full AIP. This METS Manifest contains URI references to all content files, but does *not* contain any content files. **This option is experimental and is meant for debugging purposes only. It should never be set to 'true' if you want to be able to restore content files.** Again, please note that when you use this option, the final result will be an XML file, NOT the normal ZIP-based AIP format.
`passwords=[value]`	export-only	false	If 'true' (and the 'DSPACE-ROLES' crosswalk is enabled, see #AIP Metadata Dissemination Configurations(see page 435)), then the AIP Disseminator will export user password hashes (i.e. encrypted passwords) into Site AIP's METS Manifest. This would allow you to restore user's passwords from Site AIP. If 'false', then user password hashes are not stored in Site AIP, and passwords cannot be restored at a later time.

Option	Ingest or Export	Default Value	Description
`skipIfParentMissing=[value]`	ingest-only	false	If 'true', ingestion will skip over any "Could not find a parent DSpaceObject" errors that are encountered during the ingestion process (Note: those errors will still be logged as "warning" messages in your DSpace log file). If you are performing a full site restore (or a restore of a larger Community/Collection hierarchy), you may encounter these errors if you have a larger number of Item mappings between Collections (i.e. Items which are mapped into several collections at once). When you are performing a recursive ingest, skipping these errors should not cause any problems. Once the missing parent object is ingested it will automatically restore the Item mapping that caused the error. For more information on this "Could not find a parent DSpaceObject" error see Common Issues or Error Messages(see page 438).
`unauthorized=[value]`	export-only	*unspecified*	If 'skip', the AIP Disseminator will skip over any unauthorized Bundle or Bitstream encountered (i.e. it will not be added to the AIP). If 'zero', the AIP Disseminator will add a Zero-length "placeholder" file to the AIP when it encounters an unauthorized Bitstream. If unspecified (the default value), the AIP Disseminator will throw an error if an unauthorized Bundle or Bitstream is encountered.
`updatedAfter=[value]`	export-only	*unspecified*	This option works as a basic form of "incremental backup". This option requires that an ISO-8601 date[355] is specified. When specified, the AIP Disseminator will only export Item AIPs which have a last-modified date **after** the specified ISO-8601 date. This option has no affect on the export of Site, Community or Collection AIPs as DSpace does not record a last-modified date for Sites, Communities or Collections. For example, when this option is specified during a full-site export, the AIP Disseminator will export the Site AIP, all

355 http://en.wikipedia.org/wiki/ISO_8601

Option	Ingest or Export	Default Value	Description
			Community AIPs, all Collection AIPs, and only Item AIPs modified after that date and time.
`validate =[value]`	both ingest and export	Export defaults to 'true', Ingest defaults to 'false'	If 'true', every METS file in AIP will be validated before ingesting or exporting. By default, DSpace will validate everything on export, but will skip validation during import. Validation on export will ensure that all exported AIPs properly conform to the METS profile (and will throw errors if any do not). Validation on import will ensure every METS file in every AIP is first validated before importing into DSpace (this will cause the ingestion processing to take longer, but tips on speeding it up can be found in the "AIP Configurations To Improve Ingestion Speed while Validating(see page 437)" section below). *DSpace recommends minimally validating AIPs on export. Ideally, you should validate both on export and import, but import validation is disabled by default in order to increase the speed of AIP restores.*

How to use additional options

These options can be passed in two main ways:

From the Command Line

From the command-line, you can add the option to your command by using the `-o` or `--option` parameter.

```
[dspace]/bin/dspace packager -r -a -t AIP -o [option1]=[value] -o [option2]=[value] -e admin@myu.edu
aip4567.zip
```

For example:

```
[dspace]/bin/dspace packager -r -a -t AIP -o ignoreParent=false -o createMetadataFields=false -e admin@myu.
edu aip4567.zip
```

Via the Java API call

If you are programmatically calling the `org.dspace.content.packager.DSpaceAIPIngester` from your own custom script, you can specify these options via the `org.dspace.content.packager.PackageParameters` class.

As a basic example:

```
PackageParameters params = new PackageParameters;
params.addProperty("createMetadataFields", "false");
params.addProperty("ignoreParent", "true");
```

5.2.4 Configuration in 'dspace.cfg'

The following new configurations relate to AIPs:

5.2.4.1 AIP Metadata Dissemination Configurations

The following configurations allow you to specify what metadata is stored within each METS-based AIP. In 'dspace.cfg', the general format for each of these settings is:

- `aip.disseminate.<setting> = <mdType>:<DSpace-crosswalk-name> [, ...]`
  - <setting> is the setting name (see below for the full list of valid settings)
  - <mdType> is optional. It allows you to specify the value of the @MDTYPE or @OTHERMDTYPE attribute in the corresponding METS element.
  - <DSpace-crosswalk-name> is required. It specifies the name of the DSpace Crosswalk which should be used to generate this metadata.
  - Zero or more `<label-for-METS>:<DSpace-crosswalk-name>` may be specified for each setting

> **AIP Metadata Recommendations**
>
> It is recommended to **minimally** use the default settings when generating AIPs. DSpace can only restore information that is included within an AIP. Therefore, if you choose to no longer include some information in an AIP, DSpace will no longer be able to restore that information from an AIP backup

The default settings in 'dspace.cfg' are:

- `aip.disseminate.techMD` - Lists the DSpace Crosswalks (by name) which should be called to populate the `<techMD>` section of the METS file within the AIP (Default: `PREMIS, DSPACE-ROLES`)
  - The `PREMIS` crosswalk generates PREMIS metadata for the object specified by the AIP
  - The `DSPACE-ROLES` crosswalk exports DSpace Group / EPerson information into AIPs in a DSpace-specific XML format. Using this crosswalk means that AIPs can be used to recreated Groups & People within the system. (NOTE: The `DSPACE-ROLES` crosswalk should be used alongside the `METSRights` crosswalk if you also wish to restore the *permissions* that Groups/People have within the System. See below for more info on the `METSRights` crosswalk.)
- `aip.disseminate.sourceMD` - Lists the DSpace Crosswalks (by name) which should be called to populate the `<sourceMD>` section of the METS file within the AIP (Default: `AIP-TECHMD`)
  - The AIP-TECHMD Crosswalk generates technical metadata (in DIM format) for the object specified by the AIP
- `aip.disseminate.digiprovMD` - Lists the DSpace Crosswalks (by name) which should be called to populate the `<digiprovMD>` section of the METS file within the AIP (Default: *None*)
- `aip.disseminate.rightsMD` - Lists the DSpace Crosswalks (by name) which should be called to populate the `<rightsMD>` section of the METS file within the AIP (Default: `DSpaceDepositLicense:DSPACE_DEPLICENSE, CreativeCommonsRDF:DSPACE_CCRDF, CreativeCommonsText:DSPACE_CCTEXT, METSRights`)
  - The `DSPACE_DEPLICENSE` crosswalk ensures the DSpace Deposit License is referenced/stored in AIP

    - The `DSPACE_CCRDF` crosswalk ensures any Creative Commons RDF Licenses are reference/stored in AIP
    - The `DSPACE_CCTEXT` crosswalk ensures any Creative Commons Textual Licenses are referenced/stored in AIP
    - The `METSRights` crosswalk ensures that Permissions/Rights on DSpace Objects (Communities, Collections, Items or Bitstreams) are referenced/stored in AIP. Using this crosswalk means that AIPs can be used to restore permissions that a particular Group or Person had on a DSpace Object. (NOTE: The `METSRights` crosswalk should always be used in conjunction with the `DSPACE-ROLES` crosswalk (see above) or a similar crosswalk. The `METSRights` crosswalk can only restore permissions, and cannot re-create Groups or EPeople in the system. The `DSPACE-ROLES` can actually re-create the Groups or EPeople as needed.)
- `aip.disseminate.dmd` - Lists the DSpace Crosswalks (by name) which should be called to populate the `<dmdSec>`section of the METS file within the AIP (Default: MODS, DIM)
    - The MODS crosswalk translates the DSpace descriptive metadata (for this object) into MODS. As MODS is a relatively "standard" metadata schema, it may be useful to include a copy of MODS metadata in your AIPs if you should ever want to import them into another (non-DSpace) system.
    - The DIM crosswalk just translates the DSpace internal descriptive metadata into an XML format. This XML format is proprietary to DSpace, but stores the metadata in a format similar to Qualified Dublin Core.

5.2.4.2 AIP Ingestion Metadata Crosswalk Configurations

The following configurations allow you to specify what DSpace Crosswalks are used during the ingestion/restoration of AIPs. These configurations also allow you to ignore areas of the METS file (in the AIP) if you do not want that area to be restored.

In `dspace.cfg`, the general format for each of these settings is:

- `mets.dspaceAIP.ingest.crosswalk.<mdType> = <DSpace-crosswalk-name>`
    - <mdType> is the type of metadata as specified in the METS file. This corresponds to the value of the @MDTYPE attribute (of that metadata section in the METS). When the @MDTYPE attribute is "OTHER", then the <mdType> corresponds to the @OTHERMDTYPE attribute value.
    - <DSpace-crosswalk-name> specifies the name of the DSpace Crosswalk which should be used to ingest this metadata into DSpace. You can specify the "NULLSTREAM" crosswalk if you specifically want this metadata to be ignored (and skipped over during ingestion).

By default, the settings in `dspace.cfg` are:

```
mets.dspaceAIP.ingest.crosswalk.DSpaceDepositLicense = NULLSTREAM
mets.dspaceAIP.ingest.crosswalk.CreativeCommonsRDF = NULLSTREAM
mets.dspaceAIP.ingest.crosswalk.CreativeCommonsText = NULLSTREAM
```

The above settings tell the ingester to **ignore** any metadata sections which reference DSpace Deposit Licenses or Creative Commons Licenses. These metadata sections can be safely ignored as long as the "LICENSE" and "CC_LICENSE" bundles are included in AIPs (which is the default setting). As the Licenses are included in those Bundles, they will already be restored when restoring the bundle contents.

More Info on Default Crosswalks used

If unspecified in the above settings, the AIP ingester will automatically use the Crosswalk which is named the same as the @MDTYPE or @OTHERMDTYPE attribute for the metadata section. For example, a metadata section with an @MDTYPE="PREMIS" will be processed by the DSpace Crosswalk named "PREMIS".

5.2.4.3 AIP Ingestion EPerson Configurations

The following setting determines whether the AIP Ingester should create an EPerson (if necessary) when attempting to restore or ingest an Item whose Submitter cannot be located in the system. By default it is set to "false", as for AIPs the creation of EPeople (and Groups) is generally handled by the `DSPACE-ROLES` crosswalk (see #AIP Metadata Dissemination Configurations(see page 435) for more info on `DSPACE-ROLES` crosswalk.)

- `mets.dspaceAIP.ingest.createSubmitter = false`

5.2.4.4 AIP Configurations To Improve Ingestion Speed while Validating

It is recommended to validate all AIPs on ingestion (when possible). But validation can be extremely slow, as each validation request first must download all referenced Schema documents from various locations on the web (sometimes as many as 10 schemas may be necessary to download in order to validate a single METS file). To make matters worse, the same schema will be re-downloaded each time it is used (i.e. it is not cached locally). So, if you are validating just 20 METS files which each reference 10 schemas, that results in 200 download requests.

In order to perform validations in a speedy fashion, you can pull down a local copy of **all** schemas. Validation will then use this local cache, which can sometimes increase the speed up to 10 x.

To use a local cache of XML schemas when validating, use the following settings in 'dspace.cfg'. The general format is:

- `mets.xsd.<abbreviation> = <namespace> <local-file-name>`
  - `<abbreviation>` is a unique abbreviation (of your choice) for this schema
  - `<namespace>` is the Schema namespace
  - `<local-file-name>` the full name of the cached schema file (which should reside in your `[dspace]/config/schemas/` directory, by default this directory does not exist – you will need to create it)

The default settings are all commented out. But, they provide a full listing of all schemas currently used during validation of AIPs. In order to utilize them, uncomment the settings, download the appropriate schema file, and save it to your `[dspace]/config/schemas/` directory (by default this directory does not exist – you will need to create it) using the specified file name:

```
#mets.xsd.mets = http://www.loc.gov/METS/ mets.xsd
#mets.xsd.xlink = http://www.w3.org/1999/xlink xlink.xsd
#mets.xsd.mods = http://www.loc.gov/mods/v3 mods.xsd
#mets.xsd.xml = http://www.w3.org/XML/1998/namespace xml.xsd
#mets.xsd.dc = http://purl.org/dc/elements/1.1/ dc.xsd
#mets.xsd.dcterms = http://purl.org/dc/terms/ dcterms.xsd
#mets.xsd.premis = http://www.loc.gov/standards/premis PREMIS.xsd
#mets.xsd.premisObject = http://www.loc.gov/standards/premis PREMIS-Object.xsd
#mets.xsd.premisEvent = http://www.loc.gov/standards/premis PREMIS-Event.xsd
#mets.xsd.premisAgent = http://www.loc.gov/standards/premis PREMIS-Agent.xsd
#mets.xsd.premisRights = http://www.loc.gov/standards/premis PREMIS-Rights.xsd
```

5.2.5 Common Issues or Error Messages

The below table lists common fixes to issues you may encounter when backing up or restoring objects using AIP Backup and Restore.

Issue / Error Message	**How to Fix this Problem**
Ingest/Restore Error: "Group Administrator already exists"	If you receive this problem, you are likely attempting to Restore an Entire Site(see page 425), but are not running the command in Force Replace Mode (`-r -f`). Please see the section on Restoring an Entire Site(see page 425) for more details on the flags you should be using.
Ingest/Restore Error: "Unknown Metadata Schema encountered (mycustomschema)"	If you receive this problem, one or more of your Items is using a custom metadata schema which DSpace is currently not aware of (in the example, the schema is named "mycustomschema"). Because DSpace AIPs do not contain enough details to recreate the missing Metadata Schema, you must create it manually via the DSpace Admin UI. **Please note that you only need to create the Schema. You do not need to manually create all the fields belonging to that schema, as DSpace will do that for you as it restores each AIP.** Once the schema is created in DSpace, re-run your restore command. DSpace will automatically re-create all fields belonging to that custom metadata schema as it restores each Item that uses that schema.

Issue / Error Message	How to Fix this Problem
Ingest Error: "Could not find a parent DSpaceObject referenced as 'xxx/xxx'"	When you encounter this error message it means that an object could not be ingested/restored as it belongs to a **parent** object which doesn't currently exist in your DSpace instance. During a full restore process, this error can be skipped over and treated as a warning by specifying the '`-o skipIfParentMissing=true`' option (see Additional Packager Options(see page 429)). If you have a larger number of Items which are mapped to multiple Collections, the AIP Ingester will sometimes attempt to restore an item mapping before the **Collection itself** has been restored (thus throwing this error). Luckily, this is not anything to be concerned about. As soon as the Collection is restored, the Item Mapping which caused the error will also be automatically restored. So, if you encounter this error during a full restore, it is safe to bypass this error message using the '`-o skipIfParentMissing=true`' option. All your Item Mappings should still be restored correctly.
Submit Error: PSQLException: ERROR: duplicate key value violates unique constraint "handle_handle_key"	This error means that while submitting one or more AIPs, DSpace encountered a Handle conflict. This is a general error the may occur in DSpace if your Handle sequence has somehow become out-of-date. However, it's easy to fix. Just run the `[dspace]/etc/postgres/update-sequences.sql` script (or if you are using Oracle, run: `[dspace]/etc/oracle/update-sequences.sql`).

5.2.6 DSpace AIP Format

5.2.6.1 Makeup and Definition of AIPs

AIPs only store the Latest Version of Items

If you are using the Item Level Versioning(see page 307) functionality (disabled by default), you must be aware that this "Item Level Versioning" feature is **not yet compatible** with AIP Backup & Restore. **Using them together may result in accidental data loss.** Currently the AIPs that DSpace generates only store the *latest version* of an Item. Therefore, past versions of Items will always be lost when you perform a restore / replace using AIP tools.

AIPs are Archival Information Packages.

- AIP is a package describing **one archival object** in DSpace.
  - The **archival object** may be a single **Item**, **Collection**, **Community**, or **Site** (Site AIPs contain site-wide information). Bitstreams are included in an Item's AIP.
  - Each AIP is logically self-contained, can be restored without rest of the archive. (So you could restore a single Item, Collection or Community)
  - Collection or Community AIPs do **not** include all child objects (e.g. Items in those Collections or Communities), as each AIP only describes **one** object. However, these container AIPs do contain references (links) to all child objects. These references can be used by DSpace to automatically restore all referenced AIPs when restoring a Collection or Community.
  - AIPs are only generated for objects which are currently in the "in archive" state in DSpace. This means that in-progress, uncompleted submissions are not described in AIPs and cannot be restored after a disaster. Permanently removed objects will also no longer be exported as AIPs after their removal. However, withdrawn objects will continue to be exported as AIPs, since they are still considered under the "in archive" status.
  - AIPs with identical contents will always have identical checksums[356]. This provides a basic means of validating whether the contents within an AIP have changed. For example, if a Collection's AIP has the same checksum at two different points in time, it means that Collection has not changed during that time period.
  - AIP profile favors completeness and accuracy rather than presenting the semantics of an object in a standard format. It conforms to the quirks of DSpace's internal object model rather than attempting to produce a universally understandable representation of the object. When possible, an AIP tries to use common standards to express objects.
  - An AIP *can* serve as a DIP (Dissemination Information Package) or SIP (Submission Information Package), especially when transferring custody of objects to another DSpace implementation.

356 http://en.wikipedia.org/wiki/Checksum

- In contrast to SIP or DIP, the AIP should include all available DSpace structural and administrative metadata, and basic provenance information. AIPs also describe some basic system level information (e.g. Groups and People).

General AIP Structure / Examples

Generally speaking, an AIP is an Zip file containing a METS[357] manifest and all related content bitstreams, license files and any other associated files.

Some examples include:

- Site AIP (Sample: SITE-example.zip[358])
  - METS contains basic metadata about DSpace Site and persistent IDs referencing all Top Level Communities
  - METS also contains a list of all Groups and EPeople information defined in the DSpace system. (NOTE: By default, user passwords are not stored in AIPs, unless you specify the 'passwords' flag. See Additional Packager Options(see page 429).)
- Community AIP (Sample: COMMUNITY@123456789-1.zip[359])
  - METS contains all metadata for Community and persistent IDs referencing all members (SubCommunities or Collections). Package may also include a Logo file, if one exists.
  - METS contains any Group information for Community-specific groups (e.g. `COMMUNITY_<ID>_ADMIN` group).
  - METS contains all Community permissions/policies (translated into METSRights schema[360])
- Collection AIP (Sample: COLLECTION@123456789-2.zip[361])
  - METS contains all metadata for Collection and persistent IDs referencing all members (Items). Package may also include a Logo file, if one exists.
  - METS contains any Group information for Collection-specific groups (e.g. `COLLECTION_<ID>_ADMIN`, `COLLECTION_<ID>_SUBMIT`, etc.).
  - METS contains all Collection permissions/policies (translated into METSRights schema[362])
  - If the Collection has an Item Template, the METS will also contain all the metadata for that Item Template.
- Item AIP (Sample: ITEM@123456789-8.zip[363])
  - METS contains all metadata for Item and references to all Bundles and Bitstreams. Package also includes all Bitstream files.
  - METS contains all Item/Bundle/Bitstream permissions/policies (translated into METSRights schema[364])

Notes:

- Bitstreams and Bundles are second-class archival objects; they are recorded in the context of an Item.
- BitstreamFormats are not even second-class; they are described implicitly within Item technical metadata, and reconstructed from that during restoration
- EPeople are only defined in Site AIP, but may be referenced from Community or Collection AIPs

---

357 https://www.loc.gov/standards/mets/METSOverview.v2.html
358 https://wiki.lyrasis.org/download/attachments/104566793/SITE-example.zip?api=v2&modificationDate=1540322323714&version=1
359 https://wiki.lyrasis.org/download/attachments/104566793/COMMUNITY-123456789-1.zip?api=v2&modificationDate=1540322323706&version=1
360 http://www.loc.gov/standards/rights/METSRights.xsd
361 https://wiki.lyrasis.org/download/attachments/104566793/COLLECTION-123456789-2.zip?api=v2&modificationDate=1540322323696&version=1
362 http://www.loc.gov/standards/rights/METSRights.xsd
363 https://wiki.lyrasis.org/download/attachments/104566793/ITEM-123456789-8.zip?api=v2&modificationDate=1540322323708&version=1
364 http://www.loc.gov/standards/rights/METSRights.xsd

- Groups may be defined in Site AIP, Community AIP or Collection AIP. Where they are defined depends on whether the Group relates specifically to a single Community or Collection, or is just a general site-wide group.

What is NOT in AIPs

- DSpace Site configurations ([dspace]/config/ directory) or customizations (themes, stylesheets, etc) are not described in AIPs
- DSpace Database model (or customizations therein) is not described in AIPs
- Any objects which are not currently in the "In Archive" state are not described in AIPs. This means that in-progress, unfinished submissions are never included in AIPs.

Customizing What Is Stored in Your AIPs

If you choose, you can customize exactly what information is stored in your AIPs. However, you should be aware that you can only restore information which is stored within your AIPs. If you choose to remove information from your AIPs, you will be unable to restore it later on (unless you are also backing up your entire DSpace database and assetstore folder).

> **AIP Recommendations**
>
> It is recommended to minimally use the default settings when generating AIPs. DSpace can only restore information that is included within an AIP. Therefore, if you choose to no longer include some information in an AIP, DSpace will no longer be able to restore that information from an AIP backup

There are two ways to go about customizing your AIP format:

1. You can customize your `dspace.cfg` settings pertaining to AIP generation(see page 435). These configurations will allow you to specify exactly which DSpace Crosswalks will be called when generating the AIP METS manifest.
2. You can export your AIPs using one of the special options/flags(see page 429).

5.2.6.2 AIP Details: METS Structure

> This METS Structure is based on the structure decided for the original AipPrototype[365], developed as part of the MIT & UCSD PLEDGE project.

- `mets` element
  - `@PROFILE` fixed value="http://www.dspace.org/schema/aip/1.0/mets.xsd" (this is how we identify an AIP manifest)
  - `@OBJID` URN-format persistent identifier (i.e. Handle) if available, or else a unique identifier. (e.g. "hdl:123456789/1")
  - `@LABEL` title if available
  - `@TYPE` DSpace object type, one of "DSpace ITEM", "DSpace COLLECTION", "DSpace COMMUNITY" or "DSpace SITE".
  - `@ID` is a globally unique identifier, built using the Handle and the Object type (e.g. `dspace-COLLECTION-hdl:123456789/3`).
- `mets/metsHdr` element
  - `@LASTMODDATE` last-modified date for a DSpace Item, or nothing for other objects.
  - `agent` element:

365 https://wiki.lyrasis.org/display/DSPACE/AipPrototype

        - `@ROLE` = "CUSTODIAN",
        - `@TYPE` = "OTHER",
        - `@OTHERTYPE` = "DSpace Archive",
        - `name` = *Site handle*. (Note: The Site Handle is of the format `[handle_prefix]/0`, e.g. "123456789/0")
    - `agent` element:
        - `@ROLE` = "CREATOR",
        - `@TYPE` = "OTHER",
        - `@OTHERTYPE` = "DSpace Software",
        - `name` = "DSpace [version]" (Where "[version]" is the specific version of DSpace software which created this AIP, e.g. "1.7.0")
- `mets/dmdSec` element(s)
    - By default, two `dmdSec` elements are included for all AIPs:
        i. object's descriptive metadata crosswalked to MODS (specified by `mets/dmdSec/mdWrap@MDTYPE="MODS"`). See #MODS Schema(see page 447) section below for more information.
        ii. object's descriptive metadata in DSpace native DIM intermediate format, to serve as a complete and precise record for restoration or ingestion into another DSpace. Specified by `mets/dmdSec/mdWrap@MDTYPE="OTHER",@OTHERMDTYPE="DIM"`. See #DIM (DSpace Intermediate Metadata) Schema(see page 445) section below for more information.
    - For Collection AIPs, additional `dmdSec` elements may exist which describe the Item Template for that Collection. Since an Item template is not an actual Item (i.e. it only includes metadata), it is stored within the Collection AIP. The Item Template's `dmdSec` elements will be referenced by a `div @TYPE="DSpace ITEM Template"` in the METS `structMap`.
    - When the `mdWrap` `@TYPE` value is `OTHER`, the element *MUST* include a value for the `@OTHERTYPE` attribute which names the crosswalk that produced (or interprets) that metadata, e.g. `DIM`.
- `mets/amdSec` element(s)
    - One or more `amdSec` elements are include for all AIPs. The first `amdSec` element contains administrative metadata (technical, source, rights, and provenance) for the entire archival object. Additional `amdSec` elements may exist to describe parts of the archival object (e.g. Bitstreams or Bundles in an Item).
        - `techMD` elements. By default, two types of `techMD` elements may be included:
            - `PREMIS` metadata about an object may be included here (*currently only specified for Bitstreams (files)*). Specified by `mdWrap@MDTYPE="PREMIS"`. See #PREMIS Schema(see page 451) section below for more information.
            - `DSPACE-ROLES` metadata may appear here to describe the Groups or EPeople related to this object (_currently only specified for Site, Community and Collection). Specified by `mdWrap@MDTYPE="OTHER",@OTHERMDTYPE="DSPACE-ROLES"`. See #DSPACE-ROLES Schema(see page 452) section below for more information.
        - `rightsMD` elements. By default, there are four possible types of `rightsMD` elements which may be included:
            - `METSRights` metadata may appear here to describe the permissions on this object. Specified by `mdWrap@MDTYPE="OTHER",@OTHERMDTYPE="METSRIGHTS"`. See #METSRights Schema(see page 455) section below for more information.
            - `DSpaceDepositLicense` if the object is an Item and it has a deposit license, it is contained here. Specified by `mdWrap@MDTYPE="OTHER",@OTHERMDTYPE="DSpaceDepositLicense"`.
            - `CreativeCommonsRDF` If the object is an Item with a Creative Commons license expressed in RDF, it is included here. Specified by `mdWrap@MDTYPE="OTHER",@OTHERMDTYPE="CreativeCommonsRDF"`.
            - `CreativeCommonsText` If the object is an Item with a Creative Commons license in plain text, it is included here. Specified by `mdWrap@MDTYPE="OTHER",@OTHERMDTYPE="CreativeCommonsText"`.

        - `sourceMD` element. By default, there is only one type of `sourceMD` element which may appear:
            - `AIP-TECHMD` metadata may appear here. This stores basic technical/source metadata about in object in a DSpace native format. Specified by `mdWrap@MDTYPE="OTHER",@OTHERMDTYPE="AIP-TECHMD"`. See #AIP Technical Metadata Schema (AIP-TECHMD)(see page 448) section below for more information.
        - `digiprovMD` element.
            - *Not used at this time.*
- `mets/fileSec` element
    - For ITEM objects:
        - Each distinct Bundle in an Item goes into a `fileGrp`. The `fileGrp` has a `@USE` attribute which corresponds to the Bundle name.
        - Bitstreams in bundles become `file` elements under `fileGrp`.
        - `mets/fileSec/fileGrp/file`elements
            - Set `@SIZE` to length of the bitstream. There is a redundant value in the <techMD> but it is more accessible here.
            - Set `@MIMETYPE`, `@CHECKSUM`, `@CHECKSUMTYPE` to corresponding bitstream values. There is redundant info in the <techMD>. (For DSpace, the `@CHECKSUMTYPE="MD5"` at all times)
            - SET `@SEQ` to bitstream's SequenceID if it has one.
            - SET `@ADMID` to the list of `<amdSec>` element(s) which describe this bitstream.
    - For COLLECTION and COMMUNITY objects:
        - *Only* if the object has a *logo bitstream*, there is a `fileSec` with one `fileGrp` child of `@USE="LOGO"`.
        - The `fileGrp` contains one `file` element, representing the logo Bitstream. It has the same `@MIMETYPE`, `@CHECKSUM`, `@CHECKSUMTYPE` attributes as the Item content bitstreams, but does NOT include metadata section references (e.g. `@ADMID`) or a `@SEQ` attribute.
        - See the main `structMap` for the `fptr` reference to this logo file.
- `mets/structMap` - Primary structure map, `@LABEL="DSpace Object", @TYPE="LOGICAL"`
    - For ITEM objects:
        i. Top-Level `div` with `@TYPE="DSpace Object Contents"`.
            - For every Bitstream in Item it contains a `div` with `@TYPE="DSpace BITSTREAM"`. Each Bitstream `div` has a single `fptr` element which references the bitstream location.
        - If Item has primary bitstream, put it in `structMap/div/fptr` (i.e. directly under the `div` with `@TYPE="DSpace Object Contents"`)
    - For COLLECTION objects:
        i. Top-Level `div` with `@TYPE="DSpace Object Contents"`.
            - For every Item in the Collection, it contains a `div` with `@TYPE="DSpace ITEM"`. Each Item `div` has up to two child `mptr`elements:
                a. One linking to the Handle of that Item. Its `@LOCTYPE="HANDLE"`, and `@xlink:href` value is the raw Handle.
                b. (Optional) one linking to the location of the local AIP for that Item (if known). Its `@LOCTYPE="URL"`, and `@xlink:href` value is a relative link to the AIP file on the local filesystem.
        - If Collection has a Logo bitstream, there is an `fptr` reference to it in the very first `div`.
        - If the Collection includes an Item Template, there will be a `div` with `@TYPE="DSpace ITEM Template"` within the very first `div`. This `div @TYPE="DSpace ITEM Template"` must have a `@DMDID` specified, which links to the `dmdSec` element(s) that contain the metadata for the Item Template.
    - For COMMUNITY objects:
        i. Top-Level `div` with `@TYPE="DSpace Object Contents"`.
            - For every Sub-Community in the Community it contains a `div` with `@TYPE="DSpace COMMUNITY"`. Each Community `div` has up to two `mptr`elements:

            a. One linking to the Handle of that Community. Its `@LOCTYPE="HANDLE"`, and `@xlink:href` value is the raw Handle.
            b. (Optional) one linking to the location of the local AIP file for that Community (if known). Its `@LOCTYPE="URL"`, and `@xlink:href` value is a relative link to the AIP file on the local filesystem.
        - For every Collection in the Community there is a `div` with `@TYPE="DSpace COLLECTION"`. Each Collection `div` has up to two `mptr`elements:
            a. One linking to the Handle of that Collection. Its `@LOCTYPE="HANDLE"`, and `@xlink:href` value is the raw Handle.
            b. (Optional) one linking to the location of the local AIP file for that Collection (if known). Its `@LOCTYPE="URL"`, and `@xlink:href` value is a relative link to the AIP file on the local filesystem.
    - If Community has a Logo bitstream, there is an `fptr` reference to it in the very first `div`.
- For SITE objects:
    i. Top-Level `div` with `@TYPE="DSpace Object Contents"`.
        - For every Top-level Community in Site, it contains a `div` with `@TYPE="DSpace COMMUNITY"`. Each Item `div` has up to two child `mptr`elements:
            a. One linking to the Handle of that Community. Its `@LOCTYPE="HANDLE"`, and `@xlink:href` value is the raw Handle.
            b. (Optional) one linking to the location of the local AIP for that Community (if known). Its `@LOCTYPE="URL"`, and `@xlink:href` value is a relative link to the AIP file on the local filesystem.
- `mets/structMap` - Structure Map to indicate object's Parent, `@LABEL="Parent", @TYPE="LOGICAL"`
    - Contains one `div` element which has the unique attribute value `TYPE="AIP Parent Link"` to identify it as the older of the *parent pointer*.
        - It contains a `mptr` element whose `xlink:href` attribute value is the raw Handle of the parent object, e.g. `1721.1/4321`.

5.2.6.3 Metadata in METS

The following tables describe how various metadata schemas are populated (via DSpace Crosswalks) in the METS file for an AIP.

DIM (DSpace Intermediate Metadata) Schema

DIM Schema[366] is essentially a way of representing DSpace internal metadata structure in XML. DSpace's internal metadata is very similar to a Qualified Dublin Core in its structure, and is primarily meant for descriptive metadata. However, DSpace's metadata allows for custom elements, qualifiers or schemas to be created (so it is extendable to any number of schemas, elements, qualifiers). These custom fields/schemas may or may not be able to be translated into normal Qualified Dublin Core. So, the DIM Schema must be able to express metadata schemas, elements or qualifiers which may or may not exist within Qualified Dublin Core.

In the METS structure, DIM metadata always appears within a `dmdSec` inside an `<mdWrap MDTYPE="OTHER" OTHERMDTYPE="DIM">` element. For example:

```
<dmdSec ID="dmdSec_2190">
   <mdWrap MDTYPE="OTHER" OTHERMDTYPE="DIM">
    ...
   </mdWrap>
</dmdSec>
```

366 https://wiki.lyrasis.org/display/DSPACE/DSpaceIntermediateMetadata

By default, DIM metadata is always included in AIPs. It is controlled by the following configuration in your `dspace.cfg`:

```
aip.disseminate.dmd = MODS, DIM
```

DIM Descriptive Elements for Item objects

As all DSpace Items already have user-assigned DIM (essentially Qualified Dublin Core) metadata fields, those fields are just exported into the DIM Schema[367] within the METS file.

DIM Descriptive Elements for Collection objects

For Collections, the following fields are translated to the DIM schema:

DIM Metadata Field	Database field or value
dc.description	'introductory_text' field
dc.description.abstract	'short_description' field
dc.description.tableofcontents	'side_bar_text' field
dc.identifier.uri	Collection's handle
dc.provenance	'provenance_description' field
dc.rights	'copyright_text' field
dc.rights.license	'license' field
dc.title	'name' field

DIM Descriptive Elements for Community objects

For Communities, the following fields are translated to the DIM schema:

367 https://wiki.lyrasis.org/display/DSPACE/DSpaceIntermediateMetadata

DIM Metadata Field	Database field or value
dc.description	'introductory_text' field
dc.description.abstract	'short_description' field
dc.description.tableofcontents	'side_bar_text' field
dc.identifier.uri	Handle of Community
dc.rights	'copyright_text' field
dc.title	'name' field

DIM Descriptive Elements for Site objects

For the Site Object, the following fields are translated to the DIM schema:

Metadata Field	Value
dc.identifier.uri	Handle of Site (format: `[handle_prefix]/0`)
dc.title	Name of Site (from dspace.cfg 'dspace.name' config)

MODS Schema

By default, all DSpace descriptive metadata (DIM) is also translated into the MODS Schema[368] by utilizing DSpace's `MODSDisseminationCrosswalk`. DSpace's DIM to MODS crosswalk is defined within your `[dspace]/config/crosswalks/mods.properties` configuration file. This file allows you to customize the MODS that is included within your AIPs.

For more information on the MODS Schema, see http://www.loc.gov/standards/mods/mods-schemas.html

In the METS structure, MODS metadata always appears within a `dmdSec` inside an `<mdWrap MDTYPE="MODS">` element. For example:

368 http://www.loc.gov/standards/mods/

```
<dmdSec ID="dmdSec_2189">
   <mdWrap MDTYPE="MODS">
    ...
   </mdWrap>
</dmdSec>
```

By default, MODS metadata is always included in AIPs. It is controlled by the following configuration in your `dspace.cfg`:

```
aip.disseminate.dmd = MODS, DIM
```

The MODS metadata is included within your AIP to support interoperability. It provides a way for other systems to interact with or ingest the AIP without needing to understand the DIM Schema. You may choose to disable MODS if you wish, however this may decrease the likelihood that you'd be able to easily ingest your AIPs into a non-DSpace system (unless that non-DSpace system is able to understand the DIM schema). When restoring/ingesting AIPs, DSpace will always first attempt to restore DIM descriptive metadata. Only if no DIM metadata is found, will the MODS metadata be used during a restore.

AIP Technical Metadata Schema (AIP-TECHMD)

The AIP Technical Metadata Schema is a way to translate technical metadata about a DSpace object into the DIM Schema[369]. It is kept separate from DIM as it is considered technical metadata rather than descriptive metadata.

In the METS structure, AIP-TECHMD metadata always appears within a `sourceMD` inside an `<mdWrap MDTYPE="OTHER" OTHERMDTYPE="AIP-TECHMD">` element. For example:

```
<amdSec ID="amd_2191">
    ...
    <sourceMD ID="sourceMD_2198">
       <mdWrap MDTYPE="OTHER" OTHERMDTYPE="AIP-TECHMD">
       ...
       </mdWrap>
    </sourceMD>
    ...
</amdSec>
```

By default, AIP-TECHMD metadata is always included in AIPs. It is controlled by the following configuration in your `dspace.cfg`:

```
aip.disseminate.sourceMD = AIP-TECHMD
```

369 https://wiki.lyrasis.org/display/DSPACE/DSpaceIntermediateMetadata

AIP Technical Metadata for Item

Metadata Field	Value
dc.contributor	Submitter's email address
dc.identifier.uri	Handle of Item
dc.relation.isPartOf	Owning Collection's Handle (*as a URN*)
dc.relation.isReferencedBy	All other Collection's this item is linked to (*Handle URN of each non-owner*)
dc.rights.accessRights	*"WITHDRAWN"* if item is withdrawn

AIP Technical Metadata for Bitstream

Metadata Field	Value
dc.title	Bitstream's name/title
dc.title.alternative	Bitstream's source
dc.description	Bitstream's description
dc.format	Bitstream Format Description
dc.format.medium	Short Name of Format
dc.format.mimetype	MIMEType of Format
dc.format.supportlevel	System Support Level for Format (necessary to recreate Format during restore, if the format isn't know to DSpace by default)

Metadata Field	Value
dc.format.internal	Whether Format is internal (necessary to recreate Format during restore, if the format isn't know to DSpace by default)

- Outstanding Question: Why are we recording the file format support status? That's a DSpace property, rather than an Item property. Do DSpace instances rely on objects to tell them their support status?
    - Possible answer (from Larry Stone): Format support and other properties of the BitstreamFormat are recorded here in case the Item is restored in an empty DSpace that doesn't have that format yet, and the relevant bits of the format entry have to be reconstructed from the AIP. --lcs

AIP Technical Metadata for Collection

Metadata Field	Value
dc.identifier.uri	Handle of Collection
dc.relation.isPartOf	Owning Community's Handle (*as a URN*)
dc.relation.isReferencedBy	All other Communities this Collection is linked to (*Handle URN of each non-owner*)

AIP Technical Metadata for Community

Metadata Field	Value
dc.identifier.uri	Handle of Community
dc.relation.isPartOf	Handle of Parent Community (*as a URN*)

AIP Technical Metadata for Site

Metadata Field	Value
dc.identifier.uri	Site Handle (format: `[handle_prefix]/0`)

PREMIS Schema

At this point in time, the PREMIS Schema[370] is only used to represent technical metadata about DSpace Bitstreams (i.e. Files). The PREMIS metadata is generated by DSpace's `PREMISCrosswalk`. Only the PREMIS Object Entity Schema[371] is used.

In the METS structure, PREMIS metadata always appears within a `techMD` inside an `<mdWrap MDTYPE="PREMIS">` element. PREMIS metadata is **always** wrapped within a `<premis:premis>` element. For example:

```
<amdSec ID="amd_2209">
    ...
    <techMD ID="techMD_2210">
       <mdWrap MDTYPE="PREMIS">
          <premis:premis>
            ...
          </premis:premis>
       </mdWrap>
    </techMD>
    ...
</amdSec>
```

Each Bitstream (file) has its own `amdSec` within a METS manifest. So, there will be a separate PREMIS `techMD` for each Bitstream within a single Item.

By default, PREMIS metadata is always included in AIPs. It is controlled by the following configuration in your `dspace.cfg`:

```
aip.disseminate.techMD = PREMIS, DSPACE-ROLES
```

PREMIS Metadata for Bitstream

The following Bitstream information is translated into PREMIS for each DSpace Bitstream (file):

Metadata Field	Value
<premis:objectIdentifier>	Contains Bitstream direct URL
<premis:objectCategory>	Always set to "File"
<premis:fixity>	Contains MD5 Checksum of Bitstream
<premis:format>	Contains File Format information of Bistream

370 http://www.loc.gov/standards/premis/
371 http://www.loc.gov/standards/premis/schemas.html

Metadata Field	Value
<premis:originalName>	Contains original name of file

DSPACE-ROLES Schema

All DSpace Groups and EPeople objects are translated into a custom `DSPACE-ROLES` XML Schema. This XML Schema is a very simple representation of the underlying DSpace database model for Groups and EPeople. The `DSPACE-ROLES` Schemas is generated by DSpace's `RoleCrosswalk`.

Only the following DSpace Objects utilize the DSPACE-ROLES Schema in their AIPs:

- Site AIP – all Groups and EPeople are represented in DSPACE-ROLES Schema
- Community AIP – only Community-based groups (e.g. `COMMUNITY_1_ADMIN`) are represented in DSPACE-ROLES Schema
- Collection AIP – only Collection-based groups (e.g. `COLLECTION_2_ADMIN`, `COLLECTION_2_SUBMIT`, etc.) are represented in DSPACE-ROLES Schema

In the METS structure, DSPACE-ROLES metadata always appears within a `techMD` inside an `<mdWrap MDTYPE="OTHER" OTHERMDTYPE="DSPACE-ROLES">` element. For example:

```
<amdSec ID="amd_2068">
    ...
    <techMD ID="techMD_2070">
       <mdWrap MDTYPE="OTHER" OTHERMDTYPE="DSPACE-ROLES">
            ...
       </mdWrap>
    </techMD>
    ...
</amdSec>
```

By default, DSPACE-ROLES metadata is always included in AIPs. It is controlled by the following configuration in your `dspace.cfg`:

```
aip.disseminate.techMD = PREMIS, DSPACE-ROLES
```

Example of DSPACE-ROLES Schema for a SITE AIP

Below is a general example of the structure of a DSPACE-ROLES XML file, as it would appear in a SITE AIP.

```
<DSpaceRoles>
  <Groups>
    <Group ID="1" Name="Administrator">
      <Members>
        <Member ID="1" Name="bsmith@myu.edu" />
      </Members>
    </Group>
    <Group ID="0" Name="Anonymous" />
    <Group ID="70" Name="COLLECTION_hdl:123456789/57_ADMIN">
      <Members>
        <Member ID="1" Name="bsmith@myu.edu" />
      </Members>
    </Group>
    <Group ID="75" Name="COLLECTION_hdl:123456789/57_DEFAULT_READ">
      <MemberGroups>
        <MemberGroup ID="0" Name="Anonymous" />
      </MemberGroups>
    </Group>
    <Group ID="71" Name="COLLECTION_hdl:123456789/57_SUBMIT">
      <Members>
        <Member ID="1" Name="bsmith@myu.edu" />
      </Members>
    </Group>
    <Group ID="72" Name="COLLECTION_hdl:123456789/57_WORKFLOW_STEP_1">
      <MemberGroups>
        <MemberGroup ID="1" Name="Administrator" />
      </MemberGroups>
    </Group>
    <Group ID="73" Name="COLLECTION_hdl:123456789/57_WORKFLOW_STEP_2">
      <MemberGroups>
        <MemberGroup ID="1" Name="Administrator" />
      </MemberGroups>
    </Group>
    <Group ID="8" Name="COLLECTION_hdl:123456789/6703_DEFAULT_READ" />
    <Group ID="9" Name="COLLECTION_hdl:123456789/2_ADMIN">
      <Members>
        <Member ID="1" Name="bsmith@myu.edu" />
      </Members>
    </Group>
  </Groups>
  <People>
    <Person ID="1">
      <Email>bsmith@myu.edu</Email>
      <Netid>bsmith</Netid>
      <FirstName>Bob</FirstName>
      <LastName>Smith</LastName>
      <Language>en</Language>
      <CanLogin />
    </Person>
    <Person ID="2">
      <Email>jjones@myu.edu</Email>
      <FirstName>Jane</FirstName>
      <LastName>Jones</LastName>
      <Language>en</Language>
      <CanLogin />
```

```
      <SelfRegistered />
    </Person>
  </People>
</DSpaceRoles>
```

Why are there Group Names with Handles?

You may have noticed several odd looking group names in the above example, where a Handle is embedded in the name (e.g. "COLLECTION_hdl:123456789/57_SUBMIT"). This is a translation of a Group name which included a Community or Collection *Internal ID* (e.g. "COLLECTION_45_SUBMIT"). Since you are exporting these Groups outside of DSpace, the *Internal ID* may no longer be valid or be understandable. Therefore, before export, these Group names are all translated to include an externally understandable identifier, in the form of a Handle. If you use this AIP to restore your groups later, they will be translated back to the normal DSpace format (i.e. the handle will be translated back to the new *Internal ID*).

Orphaned Groups are Renamed on Export

If a Group name includes a Community or Collection *Internal ID* (e.g. "COLLECTION_45_SUBMIT"), and that Community or Collection no longer exists, then the Group is considered "Orphaned".

- In 1.8.2 and above, the Group is renamed using the following format: "ORPHANED_[object-type]_GROUP_[obj-id]_[group-type]" (e.g. "ORPHANED_COLLECTION_GROUP_10_ADMIN").
- Prior to 1.8.2, the Group was renamed with a random key: "GROUP_[random-hex-key]_[object-type]_[group-type]" (e.g. "GROUP_123eb3a_COLLECTION_ADMIN"). *This old format was discontinued as giving the groups a randomly generated name caused the SITE AIP to have a different checksum every time it was regenerated (see* DS-1120[372]*).*

The reasoning is that we were unable to translate an *Internal ID* into an *External ID* (i.e. Handle). If we are unable to do that translation, re-importing or restoring a group with an *old* internal ID could cause conflicts or instability in your DSpace system. In order to avoid such conflicts, these groups are renamed using a random, unique key.

Example of DSPACE-ROLES Schema for a Community or Collection

Below is a general example of the structure of a DSPACE-ROLES XML file, as it would appear in a Community or Collection AIP.

This specific example is for a Collection, which has associated Administrator, Submitter, and Workflow approver groups. In this very simple example, each group only has one Person as a member of it. Please notice that the Person's information (Name, NetID, etc) is NOT contained in this content (however they are available in the DSPACE-ROLES example for a SITE, as shown above)

372 https://jira.duraspace.org/browse/DS-1120

```
<DSpaceRoles>
  <Groups>
    <Group ID="9" Name="COLLECTION_hdl:123456789/2_ADMIN" Type="ADMIN">
      <Members>
        <Member ID="1" Name="bsmith@myu.edu" />
      </Members>
    </Group>
    <Group ID="13" Name="COLLECTION_hdl:123456789/2_SUBMIT" Type="SUBMIT">
      <Members>
        <Member ID="2" Name="jjones@myu.edu" />
      </Members>
    </Group>
    <Group ID="10" Name="COLLECTION_hdl:123456789/2_WORKFLOW_STEP_1" Type="WORKFLOW_STEP_1">
      <Members>
        <Member ID="1" Name="bsmith@myu.edu" />
      </Members>
    </Group>
    <Group ID="11" Name="COLLECTION_hdl:123456789/2_WORKFLOW_STEP_2" Type="WORKFLOW_STEP_2">
      <Members>
        <Member ID="2" Name="jjones@myu.edu" />
      </Members>
    </Group>
    <Group ID="12" Name="COLLECTION_hdl:123456789/2_WORKFLOW_STEP_3" Type="WORKFLOW_STEP_3">
      <Members>
        <Member ID="1" Name="bsmith@myu.edu" />
      </Members>
    </Group>
  </Groups>
</DSpaceRoles>
```

METSRights Schema

All DSpace Policies (permissions on objects) are translated into the METSRights schema[373]. This is different than the above DSPACE-ROLES schema, which only represents Groups and People objects. Instead, the METSRights schema is used to translate the permission statements (e.g. a group named "Library Admins" has Administrative permissions on a Community named "University Library"). But the METSRights schema doesn't represent who is a member of a particular group (that is defined in the DSPACE-ROLES schema, as described above).

> **METSRights should always be used with DSPACE-ROLES**
>
> The METSRights Schema must be used in conjunction with the DSPACE-ROLES Schema for Groups, People and Permissions to all be restored properly. As mentioned above, the METSRights metadata can only be used to restore permissions (i.e. DSpace policies). The DSPACE-ROLES metadata must also exist if you wish to restore the actual Group or EPeople objects to which those permissions apply.

All DSpace Object's AIPs (except for the SITE AIP) utilize the METSRights Schema in order to define what permissions people and groups have on that object. Although there are several sections to the METSRights Schema, DSpace AIPs *only use* the `<RightsDeclarationMD>` section, as this is what is used to describe rights on an object.

373 http://www.loc.gov/standards/rights/METSRights.xsd

In the METS structure, METSRights metadata always appears within a `rightsMD` inside an `<mdWrap MDTYPE="OTHER" OTHERMDTYPE="METSRIGHTS">` element. For example:

```
<amdSec ID="amd_2068">
    ...
    <rightsMD ID="rightsMD_2074">
       <mdWrap MDTYPE="OTHER" OTHERMDTYPE="METSRIGHTS">
            ...
       </mdWrap>
    </rightsMD>
    ...
</amdSec>
```

By default, METSRights metadata is always included in AIPs. It is controlled by the following configuration in your `dspace.cfg`:

```
aip.disseminate.rightsMD = DSpaceDepositLicense:DSPACE_DEPLICENSE, \
    CreativeCommonsRDF:DSPACE_CCRDF, CreativeCommonsText:DSPACE_CCTEXT, METSRIGHTS
```

Example of METSRights Schema for an Item

An Item AIP will almost always contain several METSRights metadata sections within its METS Manifest. A separate METSRights metadata section is used to describe the permissions on:

- the Item itself
- each Bundle (group of files) in the Item
- each Bitstream (file) within an Item's bundle

Below is an example of a METSRights sections for a publicly visible Bitstream, Bundle or Item. Notice it specifies that the "GENERAL PUBLIC" has the permission to DISCOVER or DISPLAY this object.

```
<rights:RightsDeclarationMD xmlns:rights="http://cosimo.stanford.edu/sdr/metsrights/"
RIGHTSCATEGORY="LICENSED">
  <rights:Context CONTEXTCLASS="GENERAL PUBLIC">
    <rights:Permissions DISCOVER="true" DISPLAY="true" MODIFY="false" DELETE="false" />
  </rights:Context>
</rights:RightsDeclarationMD>
```

As of DSpace 3, DSpace policies/permissions may also have a "start-date" or "end-date" (to support Embargo(see page 113) functionality). Such a policy on an Item may look like this. Notice it specifies that the "GENERAL PUBLIC" has the permission to DISCOVER or DISPLAY this object *starting on* 2015-01-01, while the Group "Staff" has permission to DISCOVER or DISPLAY this object *until* 2015-01-01.

```
<rights:RightsDeclarationMD xmlns:rights="http://cosimo.stanford.edu/sdr/metsrights/"
RIGHTSCATEGORY="LICENSED">
  <rights:Context CONTEXTCLASS="GENERAL PUBLIC" start-date="2015-01-01" in-effect="false">
    <rights:Permissions DISCOVER="true" DISPLAY="true" MODIFY="false" DELETE="false" />
  </rights:Context>
  <rights:Context CONTEXTCLASS="MANAGED_GRP" end-date="2015-01-01" in-effect="true">
    <rights:UserName USERTYPE="GROUP">Staff</rights:UserName>
    <rights:Permissions DISCOVER="true" DISPLAY="true" MODIFY="false" DELETE="false" />
  </rights:Context>
</rights:RightsDeclarationMD>
```

Example of METSRights Schema for a Collection

A Collection AIP contains one METSRights section, which describes the permissions different Groups or People have within the Collection

Below is an example of a METSRights sections for a publicly visible Collection, which also has an Administrator group, a Submitter group, and a group for each of the three DSpace workflow approval steps. You'll notice that each of the groups is provided with very specific permissions within the Collection. Submitters & Workflow approvers can "ADD CONTENTS" to a collection (but cannot delete the collection). Administrators have full rights.

```
<rights:RightsDeclarationMD xmlns:rights="http://cosimo.stanford.edu/sdr/metsrights/"
RIGHTSCATEGORY="LICENSED">
  <rights:Context CONTEXTCLASS="MANAGED_GRP">
    <rights:UserName USERTYPE="GROUP">COLLECTION_hdl:123456789/2_SUBMIT</rights:UserName>
    <rights:Permissions DISCOVER="true" DISPLAY="true" MODIFY="true" DELETE="false" OTHER="true"
 OTHERPERMITTYPE="ADD CONTENTS" />
  </rights:Context>
  <rights:Context CONTEXTCLASS="MANAGED_GRP">
    <rights:UserName USERTYPE="GROUP">COLLECTION_hdl:123456789/2_WORKFLOW_STEP_3</rights:UserName>
    <rights:Permissions DISCOVER="true" DISPLAY="true" MODIFY="true" DELETE="false" OTHER="true"
 OTHERPERMITTYPE="ADD CONTENTS" />
  </rights:Context>
  <rights:Context CONTEXTCLASS="MANAGED_GRP">
    <rights:UserName USERTYPE="GROUP">COLLECTION_hdl:123456789/2_WORKFLOW_STEP_2</rights:UserName>
    <rights:Permissions DISCOVER="true" DISPLAY="true" MODIFY="true" DELETE="false" OTHER="true"
 OTHERPERMITTYPE="ADD CONTENTS" />
  </rights:Context>
  <rights:Context CONTEXTCLASS="MANAGED_GRP">
    <rights:UserName USERTYPE="GROUP">COLLECTION_hdl:123456789/2_WORKFLOW_STEP_1</rights:UserName>
    <rights:Permissions DISCOVER="true" DISPLAY="true" MODIFY="true" DELETE="false" OTHER="true"
 OTHERPERMITTYPE="ADD CONTENTS" />
  </rights:Context>
  <rights:Context CONTEXTCLASS="MANAGED_GRP">
    <rights:UserName USERTYPE="GROUP">COLLECTION_hdl:123456789/2_ADMIN</rights:UserName>
    <rights:Permissions DISCOVER="true" DISPLAY="true" COPY="true" DUPLICATE="true" MODIFY="true"
DELETE="true" PRINT="true" OTHER="true" OTHERPERMITTYPE="ADMIN" />
  </rights:Context>
  <rights:Context CONTEXTCLASS="GENERAL PUBLIC">
    <rights:Permissions DISCOVER="true" DISPLAY="true" MODIFY="false" DELETE="false" />
  </rights:Context>
</rights:RightsDeclarationMD>
```

Example of METSRights Schema for a Community

A Community AIP contains one METSRights section, which describes the permissions different Groups or People have within that Community.

Below is an example of a METSRights sections for a publicly visible Community, which also has an Administrator group. As you'll notice, this content looks very similar to the Collection METSRights section (as described above)

```
<rights:RightsDeclarationMD xmlns:rights="http://cosimo.stanford.edu/sdr/metsrights/"
RIGHTSCATEGORY="LICENSED">
  <rights:Context CONTEXTCLASS="MANAGED_GRP">
    <rights:UserName USERTYPE="GROUP">COMMUNITY_hdl:123456789/10_ADMIN</rights:UserName>
    <rights:Permissions DISCOVER="true" DISPLAY="true" COPY="true" DUPLICATE="true" MODIFY="true"
DELETE="true" PRINT="true" OTHER="true" OTHERPERMITTYPE="ADMIN" />
  </rights:Context>
  <rights:Context CONTEXTCLASS="GENERAL PUBLIC">
    <rights:Permissions DISCOVER="true" DISPLAY="true" MODIFY="false" DELETE="false" />
  </rights:Context>
</rights:RightsDeclarationMD>
```

5.3 Ant targets and options

Ant targets should be run as the service user

A word of warning: in order to ensure proper permissions and file ownership are maintained, you are advised to run these ant targets as the service user (commonly 'dspace' or 'tomcat'). Running them as any other user may cause permission problems

5.3.1 Options

DSpace allows three property values to be set using the -D<property>=<value> option. They may be used in other contexts than noted below, but take care to understand how a particular property will affect a target's outcome.

overwrite

Whether to overwrite configuration files in [dspace]/config. If true, files from [dspace]/config and subdirectories are backed up with .old extension and new files are installed from [dspace-src]/dspace/config and subdirectories; if false, existing config files are untouched, and new files are written beside them with .new extension.	
Possible values:	true, false

Default:	true
Context:	update, init_configs

config

If a path is specified, ant uses values from the specified file and installs it in [dspace]/config in the appropriate contexts.	
Possible values:	path to configuration file to be used
Default:	[dspace-src]/config/dspace.cfg
Context:	update, update_configs, update_code, update_webapps, init_configs, fresh_install, test_database, setup_database, load_registries, clean_database

wars

If true, builds .war files; if false, no .war files are built.	
Possible values:	true, false
Default:	true
Context:	update, update_webapps, fresh_install

5.3.2 Targets

Target	Effect
update	Creates backup copies of the [dspace]/bin, /etc, /lib, and /webapps directories with the form /<directory>.bak-<date-time>. Creates new copies of [dspace]/config, /etc, and /lib directories. Does not affect data files or the database. (See *overwrite*, *config*, *war* options.)
update_configs	Updates the [dspace]/config directory with new configuration files. (See *config* option.)
update_code	Creates backup copies of the [dspace]/bin, /etc, and /lib directories with the form /<directory>.bak-<date-time>. Creates new copies of [dspace]/config, /etc, and /lib directories. (See *config* option.)
update_webapps	Updates [dspace]/webapps directory. (See *config*, *war* options.)
init_configs	Writes configuration files to [dspace]/config. (See *overwrite*, *config* options.)

Target	Effect
install_code	Deletes existing [dspace]/bin, /lib, and /etc directories, and installs new copies; overwrites /solr application files, leaving data intact. (See *config* option.)
fresh_install	Performs a fresh installation of the software, including the database & config. (See *config*, *war* options.)
test_database	Tests database connection using parameters specified in dspace.cfg. (See *config* option.)
clean_backups	Removes [dspace]/bin, /etc, /lib, and /webapps directories with .bak* extensions.

5.4 Command Line Operations

The DSpace command launcher or CLI interface offers the execution of different maintenance operations. As most of these are already documented in related parts of the documentation, this page is mainly intended to provide an overview of all available CLI operations, with links to the appropriate documentation.

5.4.1 Executing command line operations

The CLI interface is found at `[dspace]/bin/dspace`. Execute it without arguments or with the `-h` option to see all available operations. Execute `dspace` *op* `-h` to see details about the *op* operation.

Examples:

bin/dspace -h
bin/dspace cleanup -h
bin/dspace cleanup
bin/dspace cleanup --verbose

5.4.2 Available operations

5.4.2.1 General use

- bitstore-migrate(see page 696): Migrate all files (bitstreams) from one assetstore (bitstore) to another
- checker(see page 493): Run the checksum checker
- checker-emailer(see page 493): Send emails related to the checksum checker

- classpath: Calculate and display the DSpace classpath
- cleanup(see page 686): Remove deleted bitstreams from the assetstore
- community-filiator(see page 336): Tool to manage community and sub-community relationships
- create-administrator: Create a DSpace administrator account (see Installing DSpace(see page 53))
- curate(see page 539): Perform curation tasks on DSpace objects
- database(see page 462): Perform various tasks / checks of the DSpace database
- doi-organiser(see page 302): Transmit information about DOIs to the registration agency.
- dsprop: View the value of a DSpace property from any configuration file (see Configuration Reference(see page 552))
- dsrun: Run a (DSpace) Java class directly (used mainly for test purposes)
- embargo-lifter(see page 120): Pre DSpace 3.0 embargo manager tool used to check, list and lift embargoes
- export(see page 233): Export items or collections
- filter-media(see page 469): Perform the media filtering to extract full text from documents and to create thumbnails
- generate-sitemaps: Generate search engine and html sitemaps (see Search Engine Optimization(see page 485))
- harvest: Manage the OAI-PMH harvesting of external collections (see OAI(see page 179) harvesting docs)
- import: Import items into DSpace (see Importing and Exporting Items via Simple Archive Format(see page 233))
- index-authority[374]: import authorities and keep SOLR authority index up to date
- index-discovery(see page 386): Update Discovery(see page 386) (Solr) search and browse Index
- itemupdate: Item update tool for altering metadata and bitstream content in items (see Updating Items via Simple Archive Format(see page 333))
- make-handle-config(see page 460): Run the handle server simple setup command
- metadata-export(see page 287): Export metadata for batch editing
- metadata-import(see page 287): Import metadata after batch editing
- migrate-embargo(see page 115): Embargo manager tool used to migrate old version of Embargo to the new one included in dspace3
- oai[375]: OAI script manager
- packager(see page 244): Execute a packager
- rdfizer(see page 160): tool to convert contents to RDF
- read(see page 464) : execute a stream of commands from a file or pipe
- registry-loader: Load entries into a registry (see Metadata and Bitstream Format Registries(see page 640))
- structure-builder: Build DSpace community and collection structure (see Exporting and Importing Community and Collection Hierarchy(see page 226))
- sub-daily(see page 125): Send daily subscription notices
- test-email(see page 460): Test the DSpace email server settings are OK
- update-handle-prefix(see page 460): Update handle records and metadata when moving from one Handle prefix to another
- user(see page 121): Create, List, Update, Delete EPerson (user) records
- validate-date(see page 545): Test date-time format rules
- version(see page 492): Show DSpace version and other troubleshooting information

374 https://wiki.duraspace.org/display/DSDOC6x/ORCID+Integration#ORCIDIntegration-Importingexistingauthors&keepingtheindexuptodate

375 https://wiki.duraspace.org/display/DSDOC6x/OAI+2.0+Server#OAI2.0Server-OAIManager(SolrDataSource)

5.4.2.2 Legacy statistics

DSpace 7.0 does not yet support

Legacy/log based statistics are not available in DSpace 7.0. They are under discussion as this feature is not widely used. Tentatively they are scheduled for a possible release/replacement in 7.1, see DSpace Release 7.0 Status[376]

Legacy statistics parse the DSpace log files and compile information based on the "[dspace]/config/dstat.cfg" configuration file. They are no longer actively maintained, but still exist in the codebase because there is information they report on that is not yet accessible in (or replaced by) SOLR Statistics(see page 338). Where possible, we recommend using SOLR Statistics(see page 338) and/or Google Analytics(see page 363) for more accurate data.

- stat-general: Compile the general statistics
- stat-initial: Compile the initial statistics
- stat-monthly: Compile the monthly statistics
- stat-report-general: Create the general statistics report
- stat-report-initial: Create the initial statistics report
- stat-report-monthly: Create the monthly statistics report

5.4.2.3 SOLR Statistics

Scripts for the statistics that are stored in SOLR(see page 338):

- solr-export-statistics(see page 351):Export Solr statistics data to CSV (for backup or moving to another server)
- solr-import-statistics(see page 351): Import Solr statistics data from CSV (for restoration, or moving to another server)
- solr-reindex-statistics(see page 351): Reindex Solr statistics data (for upgrades or updates to Solr schema)
- stats-log-converter(see page 338): Convert dspace.log files ready for import into solr statistics
- stats-log-importer(see page 338): Import previously converted log files into solr statistics
- stats-util(see page 338): Statistics Client for Maintenance of Solr Statistics Indexes

5.4.3 Database Utilities

This command can be used at any time to manage or upgrade the Database. It will also assist in troubleshooting PostgreSQL and Oracle connection issues with the database.

Command used:	`[dspace]/bin/dspace database`
Java class:	org.dspace.storage.rdbms.DatabaseUtils

376 https://wiki.lyrasis.org/display/DSPACE/DSpace+Release+7.0+Status

Valid Arguments:	Description
`test`	Test the database connection settings (in `[dspace]/config/dspace.cfg` or `local.cfg`) are OK and working properly. This command also validates the database version is compatible with DSpace.
`info`	Provide detailed information about the DSpace database itself. This includes the database type, version, driver, schema, and any successful/failed/pending database migrations. This command, along with "test", is very useful in debugging issues with your database.
`migrate`	Migrate the database to the latest version (if not already on the latest version). This uses FlywayDB[377] along with embedded migrations scripts to automatically update your database to the latest version. Optionally, you can run "`migrate ignored`" to also include any database migrations which are flagged as "Ignored" by the "info" command. If these "Ignored" migrations succeed, they will now be noted (in the "info" command) as having run "OutOfOrder" (i.e. out of the normal, numerical order).
`repair`	Attempt to "repair" any migrations which are flagged as "Failed" by the "info" command and/or resolve failed checksum validation. This runs the FlywayDB repair command[378]. Please note however, this will NOT automatically repair corrupt or broken data in your database. It merely tries to re-run previously "Failed" migrations and/or realign the checksums of the applied migrations to the ones of the available migrations.

377 http://Flywaydb.org
378 https://flywaydb.org/documentation/command/repair

`clean`	Completely and permanently delete all tables and data in this database. WARNING: There is no turning back! If you run this command, you will lose your entire database and all its contents. This command is only useful for testing or for reverting your database to a "fresh install" state (e.g. running "`dspace database clean`" followed by "`dspace database migrate`" will return your database to a fresh install state) By default the 'clean' command is **disabled** (to avoid accidental data loss). In order to enable it, you must first set `db.cleanDisabled=false` in either your local.cfg or dspace.cfg.
`validate`	Validate the checksums of all previously run database migrations. This runs the FlywayDB 'validate' command[379].

5.4.4 Executing streams of commands

You can pass a sequence of commands into the `dspace` command-line tool using the `read` command.

Execute commands...	**this way**
...from a file	`[dspace]/bin/dspace read a-command-file`
...in a pipeline	`some-other-command | [dspace]/bin/dspace read -` `some-other-command | [dspace]/bin/dspace read`

5.5 Handle.Net Registry Support

DSpace comes with support for CNRI's Handle.Net Registry (HNR)[380]. This feature is *completely optional*, as DSpace functions the same with or without using a Handle Server/Registry.

A few things to keep in mind:

- You'll notice that while you've been playing around with a test server, DSpace has apparently been creating (fake) handles for you looking like *hdl:123456789/24* and so forth. These aren't really Handles, since the global Handle system doesn't actually know about them, and lots of other DSpace test installs will have created the same IDs. They're only really Handles once you've registered a prefix with CNRI (see below) and have correctly set up the Handle server included in the DSpace distribution. This Handle server communicates with the rest of the global Handle infrastructure so that anyone that understands Handles can find the Handles your DSpace has created.
- If you want to use the Handle system, you'll need to set up a Handle server. One is included with DSpace.

379 https://flywaydb.org/documentation/command/validate
380 https://handle.net/

- If you want to use the Handle system, you'll need to obtain a Handle prefix from the central CNRI Handle site[381]. This requires a small yearly fee to CNRI
- Again, all of this is **completely optional.** But, the key benefit is that it provides you with persistent, permanent URLs (of the form `https://hdl.handle.net/[prefix]/[suffix]`) for every object within your DSpace site. Those persistent URLs may be useful for citations or even during upgrades/migrations, as DSpace + Handle.Net ensures that these URLs always go to the right object, even if your site's main URL changes.

A Handle server runs as a separate process that receives TCP requests from other Handle servers, and issues resolution requests to a global server or servers if a Handle entered locally does not correspond to some local content. The Handle protocol is based on TCP, so it will need to be installed on a server that can send and receive TCP on port 2641.

You can either use a Handle server running on the same machine as DSpace, or you can install it on a separate machine. Installing it on the same machine is a little bit easier. If you install it on a separate machine, you can use one Handle server for more than one DSpace installation.

- To install your Handle resolver on the host where DSpace runs(see page 465)
- To install a Handle resolver on a separate machine(see page 466)
- To install a Handle resolver on a separate machine using template handles(see page 468)
- Updating Existing Handle Prefixes(see page 468)

5.5.1 To install your Handle resolver on the host where DSpace runs

⚠ We recommend configuring your Handle server **without a passphrase**, as the current DSpace `start-handle-server` scripts do not yet support startup with a passphrase.

If you choose to set a passphrase, you may need to start the Handle Server via: `[dspace]\bin\dspace dsrun net.handle.server.Main [dspace]\handle-server`

1. To configure your DSpace installation to run the handle server, run the following command:

```
[dspace]/bin/make-handle-config
```

   a. If you are using Windows, the proper command is:

```
[dspace]/bin/dspace dsrun net.handle.server.SimpleSetup [dspace]/handle-server
```

   Ensure that *[dspace]/handle-server* matches whatever you have in *dspace.cfg* for the *handle.dir* property. You will need to answer a series of qestions to configure the server. For the most part, you can use the default options, except you should choose to **not** encrypt your certificates when prompted.
2. Edit the resulting *[dspace]/handle-server/config.dct* file to include the following lines in the *"server_config"*clause:

381 http://www.handle.net/

```
"storage_type" = "CUSTOM"
"storage_class" = "org.dspace.handle.HandlePlugin"
"enable_txn_queue" = "no"
```

This tells the Handle server to get information about individual Handles from the DSpace code and to disable transaction replication. If you used the make-handle-config script, these should already be set in your config.dct file.

3. Once the configuration file has been generated, you will need to go to https://hdl.handle.net/4263537/5014[382] to upload the generated sitebndl.zip file. The upload page will ask you for your contact information. An administrator will then create the naming authority/prefix on the root service (known as the Global Handle Registry), and notify you when this has been completed. You will not be able to continue the handle server installation until you receive further information concerning your naming authority.
4. When CNRI has sent you your naming authority prefix, you will need to edit the *config.dct* file. The file will be found in */[dspace]/handle-server*. Look for *"300:0.NA/123456789"*. Replace 123456789 with the assigned naming authority prefix sent to you. Also change the value of `handle.prefix` in `[dspace]/config/local.cfg` from "123456789" to your assigned naming authority prefix, so that DSpace will use that prefix in assigning new Handles.
5. Now start your handle server (as the dspace user):

```
[dspace]/bin/start-handle-server
```

   a. If you are using Windows, there is a corresponding 'start-handle-server.bat' script:

```
[dspace]/bin/start-handle-server.bat
```

Note that since the DSpace code manages individual Handles, administrative operations such as Handle creation and modification aren't supported by DSpace's Handle server.

5.5.2 To install a Handle resolver on a separate machine

Not yet supported in DSpace 7

The option to run the Handle resolver on a separate machine is not yet available in DSpace 7 codebase. See this ticket:

DS-4314[383] - Remote Handle Resolver currently unsupported in DSpace 7 codebase

VOLUNTEER NEEDED

The Handle server you use must be dedicated to resolve Handles from DSpace. You cannot use a Handle server that is in use with other software already. You can use CNRI's Handle Software -- all you have to do is to add to it a plugin that is provided by DSpace. The following instructions were tested with CNRI's Handle software version 9.1.0. You can do the following steps on another machine than the machine DSpace runs on, but you have to copy some files from the machine on which DSpace is installed.

382 http://hdl.handle.net/4263537/5014
383 https://jira.lyrasis.org/browse/DS-4314?src=confmacro

1. Download the CNRI Handle Software: http:s//www.handle.net/download.html[384]. In the tarball you'll find an `README.txt` with installation instructions -- follow it.
2. After installing the CNRI Handle Software you should have two directories: once that contains the CNRI software and one that contains the configuration of you local Handle Server. For the rest of this instruction we assume that the directory containing the CNRI Software is `/hs/handle-9.1.0` and the directory containing the configuration of your local server is `/hs/srv_1`. (We use the same paths here as CNRIs README.txt.)
3. Download the plugin from https://github.com/DSpace/Remote-Handle-Resolver/releases. Select a release. You can get the source and build it yourself, or just use the JAR file included in the release. In either case, once you have a `dspace-remote-handle-resolver-VERSION.jar`, copy it to the directory containing the CNRI software (`/hs/handle-9.1.0/lib`).
4. Create the directory `/hs/srv_1/logs`.
5. Create the following two files in `/hs/srv_1`.

   log4j-handle-plugin.properties

   ```
   log4j.rootCategory=INFO, A1
   log4j.appender.A1=org.apache.log4j.DailyRollingFileAppender
   log4j.appender.A1.File=/hs/srv_1/logs/handle-plugin.log
   log4j.appender.A1.DatePattern= '.' yyyy-MM-dd
   log4j.appender.A1.layout=org.apache.log4j.PatternLayout
   log4j.appender.A1.layout.ConversionPattern=%d %-5p %c @ %m%n
   log4j.logger.org.apache.axis.handlers.http.HTTPAuthHandler=INFO
   ```

   Change the path in the third line, if necessary.

   handle-dspace-plugin.cfg

   ```
   dspace.handle.endpoint1 = http: //example.org/dspace/handleresolver
   ```

   If you run more than one DSpace Installation, you may add more DSpace Endpoints. Just increase the number at the end of the key for each: `endpoint2`, `endpoint3`....
6. Edit the file `/hs/srv_1/config.dct` to include the following lines in the "`server_config`" clause:

   ```
   "storage_type" = "CUSTOM"
   "storage_class" = "org.dspace.handle.MultiRemoteDSpaceRepositoryHandlePlugin"
   ```

7. Edit `/hs/handle-9.1.0/bin/hdl`:
   a. Find a line that contains `exec java ... net.handle.server.Main ...`
   b. Add "`-Dlog4j.configuration=file:///hs/srv_1/log4j-handle-plugin.properties -Ddspace.handle.plugin.configuration=/hs/srv_1/handle-dspace-plugin.cfg`" right in front of `net.handle.server.Main`.
8. If your handle server is running, restart it.

Please note: The Handle Server will only start if it is able to connect to at least one running DSpace Installation. It only resolves the handles of the DSpace Installations that were running when it was started.

384 http://www.handle.net/download.html

5.5.3 To install a Handle resolver on a separate machine using template handles

Instead of using the described plugin above, you can configure a Handle server (version 8+) to resolve handles based on a template. Template handle require less configuration than the plugin, and do not require an additional download. However, there are two things to keep in mind when using template handles:

1. Handles that don't exist will still generate a Handle record with a URL, even though resolving that URL will show an error page.
2. Handle records can only be generated based on the handle and the template. If you need to look up information in DSpace in or to geneate the correct url for a given handle, you will need to use a storage plugin instead.

The Handle server you use must be dedicated to resolve Handles from DSpace. You cannot use a Handle server that is in use with other software already. The following instructions were tested with CNRI's Handle software version 9.1.0.

1. Download the CNRI Handle Software: https://www.handle.net/download.html.
2. In the tarball you'll find an `README.txt` with installation instructions. Follow the directions to install and configure your Handle server. Importantly, make sure your prefixes are set correctly in the "auto_homed_prefixes" setting.
3. Edit the `server's config.dct` file to include the following line in the `"server_config"` clause:

   ```
   "namespace" = "<namespace><template delimiter'/'><value type='URL' index='1' data='https://
   demo.dspace.org/handle/${handle}'/></template></namespace>"
   ```

   In the "namespace" section, replace "`https://demo.dspace.org/handle/`" with the url endpoint for your DSpace server. The "${handle}" part of the template will be replaced with the full handle to be resolved.
4. If your handle server is running, restart it.

This configuration is a minimal example of how to configure template handles for DSpace. For more details about configuring template handles, see the Handle Technical Manual, Chapter 11[385] (PDF download).

5.5.4 Updating Existing Handle Prefixes

If you need to update the handle prefix on items created before the CNRI registration process you can run the *[dspace]/bin/dspace update-handle-prefix script*. You may need to do this if you loaded items prior to CNRI registration (e.g. setting up a demonstration system prior to migrating it to production). The script takes the current and new prefix as parameters. For example:

```
[dspace]/bin/dspace update-handle-prefix 123456789 1303
```

This script will change any handles currently assigned prefix 123456789 to prefix 1303, so for example handle 123456789/23 will be updated to 1303/23 in the database.

385 https://hdl.handle.net/20.1000/113

5.6 Mediafilters for Transforming DSpace Content

mageMagick Image Thumbnail Generator

5.6.1 MediaFilters: Transforming DSpace Content

5.6.1.1 Overview

DSpace can apply filters or transformations to files/bitstreams, creating new content. Filters are included that extract text for **full-text searching**, and create **thumbnails** for items that contain images. The media filters are controlled by the `dspace filter-media` script which traverses the asset store, invoking all configured `MediaFilter` or `FormatFilter` classes on files/bitstreams (see Configuring Media Filters(see page 586) for more information on how they are configured).

5.6.1.2 Available Media Filters

Below is a listing of all currently available Media Filters, and what they actually do:

Name	Java Class	Function	Default input formats	Enabled by Default?
PDF Text Extractor	`org.dspace.app.mediafilter.PDFFilter`	extracts the full text of Adobe PDF documents (only if text-based or OCRed) for full text indexing. (Uses the Apache PDFBox[386] tool)	Adobe PDF	yes
HTML Text Extractor	`org.dspace.app.mediafilter.HTMLFilter`	extracts the full text of HTML documents for full text indexing. (Uses Swing's HTML Parser[387])	HTML, Text	yes

386 http://pdfbox.apache.org/
387 http://java.sun.com/products/jfc/tsc/articles/bookmarks/

Name	Java Class	Function	Default input formats	Enabled by Default?
Word Text Extractor	`org.dspace.app.mediafilter.PoiWordFilter`	extracts the full text of Microsoft Word and Microsoft Word XML documents for full text indexing. (Uses the "Apache POI"[388] tools.)	Microsoft Word, Microsoft Word XML	yes
Excel Text Extractor	`org.dspace.app.mediafilter.ExcelFilter`	extracts the full text of Microsoft Excel documents for full text indexing. (Uses the "Apache POI"[389] tools.)	Microsoft Excel, Microsoft Excel XML	yes
PowerPoint Text Extractor	`org.dspace.app.mediafilter.PowerPointFilter`	extracts the full text of slides and notes in Microsoft PowerPoint and PowerPoint XML documents for full text indexing. (Uses the Apache POI[390] tools.)	Microsoft Powerpoint, Microsoft Powerpoint XML	yes
PDFBox JPEG Thumbnail	`org.dspace.app.mediafilter.PDFBoxThumbnail`	creates thumbnail images of the first page of PDF files	Adobe PDF	yes
JPEG Thumbnail	`org.dspace.app.mediafilter.JPEGFilter`	creates thumbnail images of GIF, JPEG and PNG files	BMP, GIF, JPEG, image/png	yes
Branded Preview JPEG	`org.dspace.app.mediafilter.BrandedPreviewJPEGFilter`	creates a branded preview image for GIF, JPEG and PNG files	BMP, GIF, JPEG, image/png	no
ImageMagick Image Thumbnail Generator	`org.dspace.app.mediafilter.ImageMagickImageThumbnailFilter`	Uses ImageMagick to generate thumbnails for image bitstreams. Requires installation of ImageMagick[391]on your	BMP, GIF, image/png, JPG, TIFF, JPEG, JPEG 2000	no

388 https://poi.apache.org/
389 https://poi.apache.org/
390 http://poi.apache.org
391 http://www.imagemagick.org/

Name	Java Class	Function	Default input formats	Enabled by Default?
		server. See ImageMagick Media Filters(see page 474).		
ImageMagick PDF Thumbnail Generator	`org.dspace.app.mediafilter.ImageMagickPdfThumbnailFilter`	Uses ImageMagick and Ghostscript to generate thumbnails for PDF bitstreams. Requires installation of ImageMagick[392] and Ghostscript[393] on your server. See ImageMagick Media Filters(see page 474).	Adobe PDF	no

Please note that the `filter-media` script will automatically update the DSpace search index by default.

5.6.1.3 Enabling/Disabling MediaFilters

The media filter plugin configuration `filter.plugins` in `dspace.cfg` contains a list of all enabled media/format filter plugins (see Configuring Media Filters(see page 586) for more information). By modifying the value of `filter.plugins` you can disable or enable MediaFilter plugins.

5.6.1.4 Executing (via Command Line)

The media filter system is intended to be run from the command line (or regularly as a cron task):

```
[dspace]/bin/dspace filter-media
```

With no options, this traverses the asset store, applying media filters to bitstreams, and skipping bitstreams that have already been filtered.

Available Command-Line Options:

- **Help** : `[dspace]/bin/dspace filter-media -h`
  - Display help message describing all command-line options.
- **Force mode** : `[dspace]/bin/dspace filter-media -f`
  - Apply filters to ALL bitstreams, even if they've already been filtered. If they've already been filtered, the previously filtered content is overwritten.
- **Identifier mode** : `[dspace]/bin/dspace filter-media -i 123456789/2`
  - Restrict processing to the community, collection, or item named by the identifier - by default, all bitstreams of all items in the repository are processed. The identifier must be a Handle, not a DB key. This option may be combined with any other option.
- **Maximum mode** : `[dspace]/bin/dspace filter-media -m 1000`
  - Suspend operation after the specified maximum number of items have been processed - by default, no limit exists. This option may be combined with any other option.

392 http://www.imagemagick.org/
393 http://www.ghostscript.com/

- **Plugin mode** : `[dspace]/bin/dspace filter-media -p "PDF Text Extractor","Word Text Extractor"`
    - Apply ONLY the filter plugin(s) listed (separated by commas). By default all named filters listed in the *filter.plugins* field of *dspace.cfg* are applied. This option may be combined with any other option. *WARNING:* multiple plugin names must be separated by a comma (i.e. ',') and NOT a comma followed by a space (i.e. ', ').
- **Skip mode** : `[dspace]/bin/dspace filter-media -s 123456789/9,123456789/100`
    - SKIP the listed identifiers (separated by commas) during processing. The identifiers must be Handles (not DB Keys). They may refer to items, collections or communities which should be skipped. This option may be combined with any other option. *WARNING:* multiple identifiers must be separated by a comma (i.e. ',') and NOT a comma followed by a space (i.e. ', ').
    - NOTE: If you have a large number of identifiers to skip, you may maintain this comma-separated list within a separate file (e.g. *filter-skiplist.txt*). Use the following format to call the program. *Please note the use of the "grave" or "tick" (`) symbol and do not use the single quotation.*
        - `[dspace]/bin/dspace filter-media -s `less filter-skiplist.txt``
- **Verbose mode** : `[dspace]/bin/dspace filter-media -v`
    - Verbose mode - print all extracted text and other filter details to STDOUT.
      Adding your own filters is done by creating a class which *implements* the `org.dspace.app.mediafilter.FormatFilter` interface. See the Creating a new Media/Format Filter(see page 0) topic and comments in the source file `FormatFilter.java` for more information. In theory filters could be implemented in any programming language (C, Perl, etc.) However, they need to be invoked by the Java code in the Media Filter class that you create.

5.6.1.5 Creating Custom MediaFilters

Creating a simple Media Filter

New Media Filters **must implement** the *org.dspace.app.mediafilter.FormatFilter* interface. More information on the methods you need to implement is provided in the *FormatFilter.java* source file. For example:

```
public class MySimpleMediaFilter implements FormatFilter
```

Alternatively, you could extend the *org.dspace.app.mediafilter.MediaFilter* class, which just defaults to performing no pre/post-processing of bitstreams before or after filtering.

```
public class MySimpleMediaFilter extends MediaFilter
```

You must give your new filter a "name", by adding it and its name to the *plugin.named.org.dspace.app.mediafilter.FormatFilter* field in *dspace.cfg*. In addition to naming your filter, make sure to specify its input formats in the *filter.<class path>.inputFormats* config item. Note the input formats must match the *short description* field in the Bitstream Format Registry (i.e. *bitstreamformatregistry* table).

```
plugin.named.org.dspace.app.mediafilter.FormatFilter = \
        org.dspace.app.mediafilter.MySimpleMediaFilter = My Simple Text Filter, \ ...

filter.org.dspace.app.mediafilter.MySimpleMediaFilter.inputFormats =
        Text
```

If you neglect to define the *inputFormats* for a particular filter, the *MediaFilterManager* will never call that filter, since it will never find a bitstream which has a format matching that filter's input format(s).

If you have a complex Media Filter class, which actually performs different filtering for different formats (e.g. conversion from Word to PDF **and** conversion from Excel to CSV), you should define this as described in Chapter 13.3.2.2 .

Creating a Dynamic or "Self-Named" Format Filter

If you have a more complex Media/Format Filter, which actually performs **multiple** filtering or conversions for different formats (e.g. conversion from Word to PDF **and** conversion from Excel to CSV), you should have define a class which implements the *FormatFilter* interface, while also extending the Chapter 13.3.2.2 *SelfNamedPlugin* class. For example:

```
public class MyComplexMediaFilter extends SelfNamedPlugin implements FormatFilter
```

Since *SelfNamedPlugins* are self-named (as stated), they must provide the various names the plugin uses by defining a getPluginNames() method. Generally speaking, each "name" the plugin uses should correspond to a different type of filter it implements (e.g. "Word2PDF" and "Excel2CSV" are two good names for a complex media filter which performs both Word to PDF and Excel to CSV conversions).

Self-Named Media/Format Filters are also configured differently in *dspace.cfg*. Below is a general template for a Self Named Filter (defined by an imaginary *MyComplexMediaFilter* class, which can perform both Word to PDF and Excel to CSV conversions):

```
#Add to a list of all Self Named filters
plugin.selfnamed.org.dspace.app.mediafilter.FormatFilter = \
        org.dspace.app.mediafilter.MyComplexMediaFilter
#Define input formats for each "named" plugin this filter implements
        filter.org.dspace.app.mediafilter.MyComplexMediaFilter.Word2PDF.inputFormats = Microsoft Word
        filter.org.dspace.app.mediafilter.MyComplexMediaFilter.Excel2CSV.inputFormats = Microsoft Excel
```

As shown above, each Self-Named Filter class must be listed in the `plugin.selfnamed.org.dspace.app.mediafilter.FormatFilter` item in `dspace.cfg`. In addition, each Self-Named Filter **must** define the input formats for *each named plugin* defined by that filter. In the above example the *MyComplexMediaFilter* class is assumed to have defined two named plugins, `Word2PDF` and `Excel2CSV`. So, these two valid plugin names ("Word2PDF" and "Excel2CSV") **must** be returned by the `getPluginNames()` method of the `MyComplexMediaFilter` class.

These named plugins take different input formats as defined above (see the corresponding *inputFormats* setting).

If you neglect to define the `inputFormats` for a particular named plugin, the `MediaFilterManager` will never call that plugin, since it will never find a bitstream which has a format matching that plugin's input format(s).

For a particular Self-Named Filter, you are also welcome to define additional configuration settings in *dspace.cfg*. To continue with our current example, each of our imaginary plugins actually results in a different output format (Word2PDF creates "Adobe PDF", while Excel2CSV creates "Comma Separated Values"). To allow this complex Media Filter to be even more configurable (especially across institutions, with potential different "Bitstream Format Registries"), you may wish to allow for the output format to be customizable for each named plugin. For example:

```
#Define output formats for each named plugin
filter.org.dspace.app.mediafilter.MyComplexMediaFilter.Word2PDF.output Format = Adobe PDF
filter.org.dspace.app.mediafilter.MyComplexMediaFilter.Excel2CSV.outputFormat = Comma Separated Values
```

Any custom configuration fields in *dspace.cfg* defined by your filter are ignored by the *MediaFilterManager*, so it is up to your custom media filter class to read those configurations and apply them as necessary. For example, you could use the following sample Java code in your *MyComplexMediaFilter* class to read these custom *outputFormat* configurations from *dspace.cfg*:

```
#Get "outputFormat" configuration from dspace.cfg
String outputFormat =  ConfigurationManager.getProperty(MediaFilterManager.FILTER_PREFIX + "." +
MyComplexMediaFilter.class.getName() + "." + this.getPluginInstanceName() + ".outputFormat");
```

5.6.1.6 Configuration parameters

Property	**filter.org.dspace.app.mediafilter.publicPermission**
Example Value	filter.org.dspace.app.mediafilter.publicPermission = JPEGFilter
Informational Note	By default mediafilter derivatives / thumbnails inherit the permissions of the parent bitstream, but you can override this, in case you want to make publicly accessible derivative / thumbnail content, typically the thumbnails of objects for the browse list. List the MediaFilter names that would get public accessible permissions. Any media filters not listed will instead inherit the permissions of the parent bitstream.

5.6.2 ImageMagick Media Filters

5.6.2.1 ImageMagic Media Filters

Overview

The ImageMagick Media Filters provide consistent, high quality thumbnails for image bitstreams and PDF bitstreams.

These filters require a separate software installation of the conversion utilities (ImageMagick and Ghostscript).

The media filters use the library im4java[394] to invoke the conversion utilities. This library constructs a conversion command launches a sub-process to perform the generation of media files.

Installation

- Install ImageMagick[395] on your server
- If you wish to generate PDF thumbnails, install Ghostscript[396] on your server
- The ImageMagick and Ghostscript executables should be accessible from the same directory (e.g. `/usr/bin`)

DSpace Configuration

In the **filter.plugins** section of your **dspace.cfg** file, uncomment the ImageMagick media filter definition.

```
    ImageMagick Image Thumbnail, ImageMagick PDF Thumbnail, \
```

This will activate the following settings which are already present in **dspace.cfg**:

```
    org.dspace.app.mediafilter.ImageMagickImageThumbnailFilter = ImageMagick Image Thumbnail, \
    org.dspace.app.mediafilter.ImageMagickPdfThumbnailFilter = ImageMagick PDF Thumbnail
```

These media filters contain the several properties which can be configured.

Thumbnail Dimensions

The following properties are used to define the dimensions of the generated thumbnails:

```
# maximum width and height of generated thumbnails
thumbnail.maxwidth  = 80
thumbnail.maxheight = 80
```

Conversion Utility Path

The following property provides a path to the ImageMagick and GhostScript utilities.

```
org.dspace.app.mediafilter.ImageMagickThumbnailFilter.ProcessStarter = /usr/bin
```

Overwriting Existing Thumbnails

The ImageMagick media filters can differentiate thumbnails created by the DSpace default thumbnail generator and thumbnails that were manually uploaded by a user. The media filter reads the bitstream description field to make this determination. A regular expression can be provided to define the set of thumbnails that should be

394 http://sourceforge.net/projects/im4java/
395 http://www.imagemagick.org/
396 http://www.ghostscript.com/

overwritten by the ImageMagick thumbnail generator. Thumbnail descriptions matching this pattern will be overwritten even if the -f option is not passed to the filter media process.

```
org.dspace.app.mediafilter.ImageMagickThumbnailFilter.replaceRegex = ^Generated Thumbnail$
```

The ImageMagick media filter will use the bitstream description field to identify bitstreams that it has created using the following setting. Bitstreams containing this label will be overwritten only if the -f filter is applied.

```
org.dspace.app.mediafilter.ImageMagickThumbnailFilter.bitstreamDescription = IM Thumbnail
```

Thumbnail descriptions that do not match either of the patterns listed above are presumed to be manually uploaded thumbnails. These thumbnails will not be replaced even if the -f option is passed to the filter media process.

Flatten

DSpace uses the JPEG format for thumbnails. While JPEG doesn't support transparency, PDF, PNG and other formats do. As those formats are used as outgoing material in DSpace, DSpace has to care about transparency during the generation of the thumbnails. In combinations of specific versions of ImageMagick and Ghostscript it may occur that completely transparent areas will become black. As a solution ImageMagick recommends to flatten images extracted from PDFs before they are stored as JPEG.

Since DSpace 5.2 the ImageMagick media filter flattens thumbnails extracted from PDFs. If you run into problems caused by flattening of the extracted images, you can switch the flattening off by setting the following property in **dspace.cfg** to false:

```
org.dspace.app.mediafilter.ImageMagickThumbnailFilter.flatten = false
```

ICC Profiles

PDFs optimized for physical printing often use the CMYK color space. On the web, however, the de facto color system is sRGB. By default, DSpace's ImageMagick-based thumbnailing system will create thumbnails that use the same color space as the source PDF. Most web browsers are not able to correctly display images that use the CMYK color space, which leads to images with visibly inaccurate colors.

If you are using Ghostscript version 9 or above, it is possible for DSpace to correctly convert images from CMYK to sRGB by providing it with appropriate ICC color profiles to use during thumbnail creation. Default ones are provided by most Ghostscript installations (version 9 or above). The following configuration options tell DSpace where those ICC profiles are located.

```
# org.dspace.app.mediafilter.ImageMagickThumbnailFilter.cmyk_profile = /usr/share/ghostscript/9.18/
iccprofiles/default_cmyk.icc
# org.dspace.app.mediafilter.ImageMagickThumbnailFilter.srgb_profile = /usr/share/ghostscript/9.18/
iccprofiles/default_rgb.icc
```

You may need to adjust those paths for your OS or the version of Ghostscript that you have.

Providing ICC profiles to ImageMagick is optional. If these configuration properties are unset, no profiles will be supplied to ImageMagick, and thumbnails produced from PDFs using the CMYK color space will also use CMYK. The transformation from CMYK to RGB is optional.

Additional Customization

The ImageMagick conversion software provides a large number of conversion options. Subclasses of these media filters could be written to take advantage of the additional conversion properties available in the software.

Note: The PDF thumbnail generator is hard-coded to generate a thumbnail from the first page of the PDF.

Possible Errors / Issues

"convert.im6: not authorized" errors

On Ubuntu (possibly other OSes), you may see errors like these when attempting to generate PDF thumbnails:

```
ERROR filtering, skipping bitstream:
        Item Handle: 1234/5678
        Bundle Name: ORIGINAL
        File Size: 30406135
        Checksum: c1df4b3a4755e9bed956383b61fc5042 (MD5)
        Asset Store: 0
org.im4java.core.CommandException: org.im4java.core.CommandException: convert.im6: not authorized `/tmp/
impdfthumb6294641076817830415.pdf' @ error/constitute.c/ReadImage/454.
```

These may be caused by a change in your ImageMagick policy configuration on your server.

In Ubuntu, the default "policy.xml" was recently updated to **exclude** all Ghostscript formats (including PDF, PS, etc). See this ticket: https://bugs.launchpad.net/ubuntu/+source/imagemagick/+bug/1796563

- This exclusion was implemented to workaround a security vulnerability in Ghostscript reported here: https://www.kb.cert.org/vuls/id/332928
- This vulnerability looks like it *may have been* fixed in Ghostscript v9.25: https://www.ghostscript.com/doc/9.25/News.htm (Still pending verification – see launchpad ticket linked above)

The newly added lines in the `/etc/ImageMagick/policy.xml` are these ones:

```
<!-- disable ghostscript format types -->
<policy domain="coder" rights="none" pattern="PS" />
<policy domain="coder" rights="none" pattern="EPS" />
<policy domain="coder" rights="none" pattern="PDF" />
<policy domain="coder" rights="none" pattern="XPS" />
```

If you wish to re-enable ImageMagick to process Ghostscript format types, you can simply comment out those lines in the configuration file. Be aware that, as the fix to this vulnerability is not yet verified, you should only do so at your own risk.

5.7 Performance Tuning DSpace

The software DSpace relies on does not come out of the box optimized for large repositories. Here are some tips to make it all run faster.

5.7.1 Performance Tuning the Backend (REST API)

5.7.1.1 Give Tomcat More Memory

Give Tomcat More Java Heap Memory

Java Heap Memory Recommendations

At the time of writing, DSpace recommends you should give Tomcat >= 512MB of Java Heap Memory to ensure optimal DSpace operation. Most larger sized or highly active DSpace installations however tend to allocate more like 1024MB to 2048MB of Java Heap Memory.

Performance tuning in Java basically boils down to memory. If you are seeing "`java.lang.OutOfMemoryError: Java heap space`" errors, this is a sure sign that Tomcat isn't being provided with enough Heap Memory.

Tomcat is especially memory hungry, and will benefit from being given lots of RAM. To set the amount of memory available to Tomcat, use either the `JAVA_OPTS` or `CATALINA_OPTS` environment variable, e.g:

```
CATALINA_OPTS=-Xmx512m -Xms512m
```

OR

```
JAVA_OPTS=-Xmx512m -Xms512m
```

The above example sets the maximum Java Heap memory to 512MB.

Difference between JAVA_OPTS and CATALINA_OPTS

You can use either environment variable. `JAVA_OPTS` is also used by other Java programs (besides just Tomcat). `CATALINA_OPTS` is *only used* by Tomcat. So, if you only want to tweak the memory available to Tomcat, it is recommended that you use `CATALINA_OPTS`. If you set **both** `CATALINA_OPTS` and `JAVA_OPTS`, Tomcat will default to using the settings in `CATALINA_OPTS`.

If the machine is dedicated to DSpace a decent rule of thumb is to give tomcat half of the memory on your machine. **At a minimum, you should give Tomcat >= 512MB of memory for optimal DSpace operation.** (*NOTE: As your DSpace instance gets larger in size, you may need to increase this number to the several GB range.*) The latest guidance is to also set `-Xms` to the same value as `-Xmx` for server applications such as Tomcat.

Give Tomcat More Java PermGen Memory

> **Java PermGen Memory Recommendations**
>
> At the time of writing, DSpace recommends you should give Tomcat >= 128MB of PermGen Space to ensure optimal DSpace operation.

If you are seeing "`java.lang.OutOfMemoryError: PermGen space`" errors, this is a sure sign that Tomcat is running out PermGen Memory. (More info on PermGen Space: http://blogs.sun.com/fkieviet/entry/classloader_leaks_the_dreaded_java)

To increase the amount of PermGen memory available to Tomcat (default=64MB), use either the `JAVA_OPTS` or `CATALINA_OPTS` environment variable, e.g:

```
CATALINA_OPTS=-XX:MaxPermSize=128m
```

OR

```
JAVA_OPTS=-XX:MaxPermSize=128m
```

The above example sets the maximum PermGen memory to 128MB.

> **Difference between JAVA_OPTS and CATALINA_OPTS**
>
> You can use either environment variable. `JAVA_OPTS` is also used by other Java programs (besides just Tomcat). `CATALINA_OPTS` is *only used* by Tomcat. So, if you only want to tweak the memory available to Tomcat, it is recommended that you use `CATALINA_OPTS`. If you set **both** `CATALINA_OPTS` and `JAVA_OPTS`, Tomcat will default to using the settings in `CATALINA_OPTS`.

> Please note that you can obviously set **both** Tomcat's Heap space and PermGen Space together similar to: `CATALINA_OPTS=-Xmx512m -Xms512m -XX:MaxPermSize=128m`

> On an Ubuntu machine (10.04) at least, the file `/etc/default/tomcat6` appears to be the best place to put these environmental variables.

Choosing the size of memory spaces allocated to DSpace Backend

psi-probe is a webapp that can be deployed in DSpace and be used to watch memory usage of the other webapps deployed in the same instance of Tomcat (in our case, the DSpace server webapp).

1. Download the latest version of psi-probe from https://github.com/psi-probe/psi-probe
2. Unzip probe.war into [dspace]/webapps/

```
cd [dspace]/webapps/
unzip ~/probe-3.1.0.zip
unzip probe.war -d probe
```

3. Add a Context element in Tomcat's configuration, and make it privileged (so that it can monitor the other webapps):
   EITHER in `$CATALINA_HOME/conf/server.xml`

   ```
   <Context docBase="[dspace]/webapps/probe" privileged="true" path="/probe" />
   ```

   OR in `$CATALINA_HOME/conf/Catalina/localhost/probe.xml`

   ```
   <Context docBase="[dspace]/webapps/probe" privileged="true" />
   ```

4. Edit `$CATALINA_HOME/conf/tomcat-users.xml` to add a user for logging into psi-probe (see more in https://github.com/psi-probe/psi-probe/wiki/InstallationApacheTomcat)

   ```
   <?xml version='1.0' encoding='utf-8'?>
   <tomcat-users>
     <user username="admin" password="t0psecret" roles="manager" />
   </tomcat-users>
   ```

5. Restart Tomcat
6. Open http://yourdspace.com:8080/probe/ (edit domain and port number as necessary) in your browser and use the username and password from tomcat-users.xml to log in.

In the "System Information" tab, go to the "Memory utilization" menu. Note how much memory Tomcat is using upon startup and use a slightly higher value than that for the `-Xms` parameter (initial Java heap size). Watch how big the various memory spaces get over time (hours or days), as you run various common DSpace tasks that put load on memory, including indexing, reindexing, importing items into the oai index etc. These maximum values will determine the `-Xmx` parameter (maximum Java heap size). Watching PS Perm Gen grow over time will let you choose the value for the `-XX:MaxPermSize` parameter.

5.7.1.2 Give the Command Line Tools More Memory

Give the Command Line Tools More Java Heap Memory

Similar to Tomcat, you may also need to give the DSpace Java-based command-line tools more Java Heap memory. If you are seeing "`java.lang.OutOfMemoryError: Java heap space`" errors, when running a command-line tool, this is a sure sign that it isn't being provided with enough Heap Memory.

By default, DSpace only provides 256MB of maximum heap memory to its command-line tools.

If you'd like to provide **more** memory to command-line tools, you can do so via the `JAVA_OPTS` environment variable (which is used by the `[dspace]/bin/dspace` script). Again, it's the same syntax as above:

```
JAVA_OPTS=-Xmx512m -Xms512m
```

This is especially useful for big batch jobs, which may require additional memory.

⚠ You can also edit the `[dspace]/bin/dspace` script and add the environmental variables to the script directly.

Give the Command Line Tools More Java PermGen Space Memory

Similar to Tomcat, you may also need to give the DSpace Java-based command-line tools more PermGen Space. If you are seeing "`java.lang.OutOfMemoryError: PermGen space`" errors, when running a command-line tool, this is a sure sign that it isn't being provided with enough PermGen Space.

By default, Java only provides 64MB of maximum PermGen space.

If you'd like to provide **more** PermGen Space to command-line tools, you can do so via the `JAVA_OPTS` environment variable (which is used by the `[dspace]/bin/dspace` script). Again, it's the same syntax as above:

```
JAVA_OPTS=-XX:MaxPermSize=128m
```

This is especially useful for big batch jobs, which may require additional memory.

⚠ Please note that you can obviously set **both** Java's Heap space and PermGen Space together similar to: `JAVA_OPTS=-Xmx512m -Xms512m -XX:MaxPermSize=128m`

5.7.2 Give PostgreSQL Database More Memory

On many linux distros PostgreSQL comes out of the box with an incredibly conservative configuration - it uses only 8Mb of memory! To put some more fire in its belly edit the `shared_buffers` parameter in `postgresql.conf`. The memory usage is 8KB multiplied by this value. The advice in the Postgres docs is not to increase it above 1/3 of the memory on your machine.

ⓘ **For More PostgreSQL Tips**

For more hints/tips with PostgreSQL configurations and performance tuning, see also:

- PostgresPerformanceTuning[397]
- PostgresqlConfiguration[398]

5.8 Scheduled Tasks via Cron

Several DSpace features **require** that a script is run regularly (via cron, or similar). Some of these features include:

- the e-mail subscription feature(see page 125) that alerts users of new items being deposited;
- the 'media filter' tool(see page 469), that generates thumbnails of images and extracts the full-text of documents for indexing;
- the 'checksum checker(see page 493)' that tests the bitstreams in your repository for corruption;
- the curation system queueing feature(see page 0), which allows administrators to "queue" tasks (to run at a later time) from the Admin UI;

397 https://wiki.lyrasis.org/display/DSPACE/PostgresPerformanceTuning
398 https://wiki.lyrasis.org/display/DSPACE/PostgresqlConfiguration

- and the registration of DOIs(see page 296) using DataCite as registration agency.

There are some optional periodic tasks as well:

- Updating the geolocation database(see page 338) used to enrich usage statistics. At this writing, the database publisher issues monthly updates.

These regularly scheduled tasks should be setup via either cron[399] (for Linux/Mac OSX) or Windows Task Scheduler[400] (for Windows).

5.8.1 Recommended Cron Settings

If you are on Linux or Mac OSX, **you should add these cron settings under the OS account which is running Tomcat (and owns the [dspace] installation directory)**. For example, login as that user and type the following to edit the user's crontab.

```
crontab -e
```

While every DSpace installation is unique, in order to get the most out of DSpace, we highly recommend enabling these basic cron settings (the settings are described in the comments):

399 http://en.wikipedia.org/wiki/Cron
400 http://windows.microsoft.com/en-us/windows/schedule-task

```
## SAMPLE CRONTAB FOR A PRODUCTION DSPACE
## You obviously may wish to tweak this for your own installation,
## but this should give you an idea of what you likely wish to schedule via cron.
##
## NOTE: You may also need to add additional sysadmin related tasks to your crontab
## (e.g. zipping up old log files, or even removing old logs, etc).

#-----------------
# GLOBAL VARIABLES
#-----------------
# Full path of your local DSpace Installation (e.g. /home/dspace or /dspace or similar)
# MAKE SURE TO CHANGE THIS VALUE!!!
DSPACE = [dspace]

# Shell to use
SHELL=/bin/sh

# Add all major 'bin' directories to path
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# Set JAVA_OPTS with defaults for DSpace Cron Jobs.
# Only provides 512MB of memory by default (which should be enough for most sites).
JAVA_OPTS="-Xmx512M -Xms512M -Dfile.encoding=UTF-8"

#--------------
# HOURLY TASKS (Recommended to be run multiple times per day, if possible)
# At a minimum these tasks should be run daily.
#--------------

# Send information about new and changed DOIs to the DOI registration agency
# NOTE: ONLY NECESSARY IF YOU REGISTER DOIS USING DATACITE AS REGISTRATION AGENCY
0 4,12,20 * * *  $DSPACE/bin/dspace doi-organiser -u -q ; [dspace]/bin/dspace doi-organiser -s -q ;
[dspace]/bin/dspace doi-organiser -r -q ; [dspace]/bin/dspace doi-organiser -d -q

#----------------
# DAILY TASKS
# (Recommended to be run once per day. Feel free to tweak the scheduled times below.)
#----------------

# Update the OAI-PMH index with the newest content at midnight every day
# NOTE: ONLY NECESSARY IF YOU ARE RUNNING OAI-PMH
# (This ensures new content is available via OAI-PMH)
0 0 * * * $DSPACE/bin/dspace oai import > /dev/null

# Clean and Update the Discovery indexes at midnight every day
# (This ensures that any deleted documents are cleaned from the Discovery search/browse index)
0 0 * * * $DSPACE/bin/dspace index-discovery > /dev/null

# run the index-authority script once a day at 12:45 to ensure the Solr Authority cache is up to date
45 0 * * * $DSPACE/bin/dspace index-authority > /dev/null

 # Cleanup Web Spiders from DSpace Statistics Solr Index at 01:00 every day
# NOTE: ONLY NECESSARY IF YOU ARE RUNNING SOLR STATISTICS
# (This removes any known web spiders from your usage statistics)
0 1 * * * $DSPACE/bin/dspace stats-util -i
```

```
# Send out subscription e-mails at 02:00 every day
# (This sends an email to any users who have "subscribed" to a Collection, notifying them of newly added
content.)
0 2 * * * $DSPACE/bin/dspace sub-daily

# Run the media filter at 03:00 every day.
# (This task ensures that thumbnails are generated for newly add images,
# and also ensures full text search is available for newly added PDF/Word/PPT/HTML documents)
0 3 * * * $DSPACE/bin/dspace filter-media

# Run any Curation Tasks queued from the Admin UI at 04:00 every day
# (Ensures that any curation task that an administrator "queued" from the Admin UI is executed
# asynchronously behind the scenes)
0 4 * * * $DSPACE/bin/dspace curate -q admin_ui

#----------------
# WEEKLY TASKS
# (Recommended to be run once per week, but can be run more or less frequently, based on your local needs/
policies)
#----------------
# Run the checksum checker at 04:00 every Sunday
# By default it runs through every file (-l) and also prunes old results (-p)
# (This re-verifies the checksums of all files stored in DSpace. If any files have been changed/corrupted,
checksums will differ.)
0 4 * * * $DSPACE/bin/dspace checker -l -p
# NOTE: LARGER SITES MAY WISH TO USE DIFFERENT OPTIONS. The above "-l" option tells DSpace to check
*everything*.
# If your site is very large, you may need to only check a portion of your content per week. The below
commented-out task
# would instead check all the content it can within *one hour*. The next week it would start again where it
left off.
#0 4 * * 0 $DSPACE/bin/dspace checker -d 1h -p

# Mail the results of the checksum checker (see above) to the configured "mail.admin" at 05:00 every
Sunday.
# (This ensures the system administrator is notified whether any checksums were found to be different.)
0 5 * * 0 $DSPACE/bin/dspace checker-emailer

#----------------
# MONTHLY TASKS
# (Recommended to be run once per month, but can be run more or less frequently, based on your local needs/
policies)
#----------------
# Permanently delete any bitstreams flagged as "deleted" in DSpace, on the first of every month at 01:00
# (This ensures that any files which were deleted from DSpace are actually removed from your local
filesystem.
#  By default they are just marked as deleted, but are not removed from the filesystem.)
0 1 1 * * $DSPACE/bin/dspace cleanup > /dev/null

#----------------
# YEARLY TASKS (Recommended to be run once per year)
#----------------
# At 2:00AM every January 1, "shard" the DSpace Statistics Solr index.
# This ensures each year has its own Solr index, which improves performance.
# NOTE: ONLY NECESSARY IF YOU ARE RUNNING SOLR STATISTICS
```

```
# NOTE: This is scheduled here for 2:00AM so that it happens *after* the daily cleaning of this index.
0 2 1 1 * $DSPACE/bin/dspace stats-util -s
```

5.9 Search Engine Optimization

Please be aware that individual search engines also have their own guidelines and recommendations for inclusion. While the guidelines below apply to **most** DSpace sites, you may also wish to review these guidelines for specific search engines:

- "Indexing Repositories: Pitfalls and Best Practices[401]" talk from Anurag Acharya (co-creator of Google Scholar) presented at the Open Repositories 2015 conference
- Google Scholar Inclusion Guidelines[402]
- Bing Webmaster Guidelines[403]

5.9.1 Ensuring your DSpace is indexed

Anyone who has analyzed traffic to their DSpace site (e.g. using Google Analytics or similar) will notice that a significant (and in many cases a majority) of visitors arrive via a search engine such as Google or Yahoo. Hence, to help maximize the impact of content and thus encourage further deposits, it is important to ensure that your DSpace instance is indexed effectively.

DSpace comes with tools that ensure major search engines (Google, Bing, Yahoo, Google Scholar) are able to easily and effectively index all your content. However, many of these tools provide some basic setup. Here's how to ensure your site is indexed.

For the optimum indexing, you should:

1. Keep your DSpace up to date.(see page 485) We are constantly adding new indexing improvements in new releases
2. Ensure your DSpace is visible to search engines.(see page 486)
3. Ensure the sitemaps feature is enabled(see page 486). *(enabled by default)*
4. Ensure server-side rendering is enabled in the UI.(see page 488) *(enabled by default)*
5. Ensure your robots.txt allows access to item "splash" pages and full text.(see page 488)
6. Ensure item metadata appears in HTML headers correctly.(see page 490)
7. Avoid redirecting file downloads to Item landing pages(see page 491)
8. Turn OFF any generation of PDF cover pages(see page 491)
9. As an aside, it's worth noting that OAI-PMH is generally not useful to search engines(see page 491). OAI-PMH has its own uses, but do not expect search engines to use it.

5.9.1.1 Keep your DSpace up to date

We are constantly adding new indexing improvements to DSpace. In order to ensure your site gets all of these improvements, you should strive to keep it up-to-date. For example:

- As of DSpace 7.0, Sitemaps are enabled by default (see next section)

401 http://www.or2015.net/wp-content/uploads/2015/06/or-2015-anurag-google-scholar.pdf
402 https://scholar.google.com/intl/en-US/scholar/inclusion.html
403 http://www.bing.com/webmaster/help/webmaster-guidelines-30fba23a

- As of DSpace 5.0, the DSpace robots.txt file now includes references to Sitemaps(see page 0) by default (see DS-1936[404]), and also blocks known bad bots (see DS-2335[405]).
- As of DSpace 4.0, DSpace has provided several enhancements, which were requested by the Google Scholar team. These included providing users (and web indexers) a way to browse content by the date it was added to DSpace (see DS-1482[406]), ensuring the "dc.date.issued" field is set more accurately (see DS-1481[407]), and enhancing the logic behind the "citation_pdf_url" HTML <meta> tag (see DS-1483[408])
- As of DSpace 1.7, DSpace has improved how its Item-level metadata is made available to Google Scholar. For the 1.7.0 release, the DSpace Developers worked directly with the Google Scholar developers, to ensure DSpace is generating the "citation_*" HTML "<meta>" tags (i.e. Highwire Press tags) that Google Scholar recommends in their Indexing Guidelines[409].
- As of DSpace 1.5, DSpace has support for sitemaps (both simple HTML pages of links, as well as the sitemaps.org protocol[410]). It also includes item metadata in the HTML HEAD element of item display pages, ensuring that the metadata can be effectively indexed no matter what changes you might have made to your DSpace's layout or style.
- As of DSpace 1.4, DSpace has support for the "if-modified-since" HTTP header. This basically means that if an item (or bitstream therein) has not changed since the last time a search engine's crawler indexed it, that item/bitstream does not have to be re-retrieved, sparing your server.

Additional minor improvements / bug fixes have been made to more recent releases of DSpace.

5.9.1.2 Ensure your DSpace is visible to search engines

First ensure your DSpace instance is visible, e.g. with: https://www.google.com/webmasters/tools/sitestatus

If your site is not indexed at all, all search engines have a way to add your URL, e.g.:

- Google: http://www.google.com/addurl
- Yahoo: http://siteexplorer.search.yahoo.com/submit
- Bing: http://www.bing.com/docs/submit.aspx

5.9.1.3 Ensure the sitemaps feature is enabled

As of DSpace 7, sitemaps are *enabled by default and automatically update on a daily basis.* This is the recommended setup to prefer proper indexing. So, there's nothing you need to do unless you wish to either change their schedule, or disable them.

In the dspace.cfg, the Sitemap generation schedule is controlled by this setting

```
# By default, sitemaps regenerate daily at 1:15am server time
sitemap.cron = 0 15 1 * * ?
```

You can modify this schedule by using the Cron syntax defined at https://www.quartz-scheduler.org/api/2.3.0/org/quartz/CronTrigger.html . Any modifications can be placed in your local.cfg.

If you want to disable this automated scheduler, you can either comment it out, or set it to a single "-" (dash) in your local.cfg

404 https://jira.duraspace.org/browse/DS-1936
405 https://jira.duraspace.org/browse/DS-2335
406 https://jira.duraspace.org/browse/DS-1482
407 https://jira.duraspace.org/browse/DS-1481
408 https://jira.duraspace.org/browse/DS-1483
409 http://scholar.google.com/intl/en/scholar/inclusion.html
410 http://sitemaps.org/

```
# This disables the automatic updates
sitemap.cron = -
```

Again, we **highly recommend** keeping them enabled. However, you may choose to disable this scheduler if you wish to define these in your local system cron settings.

Once you've enabled your sitemaps, they will be accessible at the following URLs:

- HTML Sitemaps: ${dspace.ui.url}/sitemap_index.html
- XML Sitemaps: ${dspace.ui.url}/sitemap_index.xml

So, for example, if your "dspace.ui.url = https://mysite.org" in your "dspace.cfg" configuration file, then the HTML Sitemaps would be at: "http://mysite.org/sitemap_index.html"

By default, the Sitemap URLs also will appear in your UI's `robots.txt` (in order to announce them to search engines):

```
# The URL to the DSpace sitemaps
# XML sitemap is listed first as it is preferred by most search engines
Sitemap: /sitemap_index.xml
Sitemap: /sitemap_index.html
```

The generate-sitemaps command

If you wanted to generate your sitemaps manually, you can use a commandline tool to do so.

WARNING: Keep in mind, you do NOT need to run these manually in most situations, as sitemaps are autoupdated on a regular schedule (see documentation above)

```
# Commandline option (run from the backend)
[dspace]/bin/dspace generate-sitemaps
```

This command accepts several options:

Option	meaning
-h --help	Explain the arguments and options.
-s --no_sitemaps	Do not generate a sitemap in sitemaps.org format.
-b -no_htmlmap	Do not generate a sitemap in htmlmap format.

Option	meaning
-a --ping_all	Notify all configured search engines that new sitemaps are available.
-p *URL* --ping *URL*	Notify the given URL that new sitemaps are available. The URL of the new sitemap will be appended to the value of *URL*.

You can configure the list of "all search engines" by setting the value of `sitemap.engineurls` in `dspace.cfg`.

5.9.1.4 Ensure Server-side rendering is enabled in the UI

Server-side rendering is enabled by default. So, you don't need to do anything, unless you've accidentally turned it off.

The DSpace UI is built on Angular.io[411], which is a JavaScript (TypeScript) based web framework. As some search engines do not support JavaScript, you MUST ensure the UI's server-side rendering is enabled. This allows the UI to send plain HTML to search engine spiders (or other clients) which do not support JavaScript.

For information on enabling, see "Universal (Server-side Rendering) settings" in User Interface Configuration(see page 367)

You can test whether server-side rendering is enabled by temporarily disabling JavaScript in your browser (usually this is in the settings of the Developer Tools) and attempting to access your DSpace site. All basic browse/search functionality should work with JavaScript disabled. (However, all dynamic menus or actions obviously will not work, as all pages will be static HTML.)

5.9.1.5 Create a good robots.txt

The trick here is to minimize load on your server, but without actually blocking anything vital for indexing. Search engines need to be able to index item, collection and community pages, and all bitstreams within items – full-text access is critically important for effective indexing, e.g. for citation analysis as well as the usual keyword searching.

If you have restricted content on your site, search engines will not be able to access it; they access all pages as an anonymous user.

Ensure that your robots.txt file is at the top level of your site: i.e. at http://repo.foo.edu/robots.txt, and NOT e.g. http://repo.foo.edu/dspace/robots.txt. If your DSpace instance is served from e.g. http://repo.foo.edu/dspace/, you'll need to add /dspace to all the paths in the examples below (e.g. /dspace/browse-subject).

NEVER BLOCK THESE PATHS

Some URLs can be disallowed without negative impact, but be ABSOLUTELY SURE the following URLs can be reached by crawlers, i.e. DO NOT put these on Disallow: lines, or your DSpace instance might not be indexed properly.

- `/bitstreams`
- `/browse/*` (UNLESS USING SITEMAPS)

411 https://angular.io/

- `/collections`
- `/communities`
- `/community-list` (UNLESS USING SITEMAPS)
- `/entities/*`
- `/handle`
- `/items`

Example good robots.txt

Below is an example good robots.txt. The highly recommended settings are uncommented. Additional, optional settings are displayed in comments – based on your local configuration you may wish to enable them by uncommenting the corresponding "Disallow:" line.

```
# The URL to the DSpace sitemaps
# XML sitemap is listed first as it is preferred by most search engines
Sitemap: /sitemap_index.xml
Sitemap: /sitemap_index.html

##########################
# Default Access Group
# (NOTE: blank lines are not allowable in a group record)
##########################
User-agent: *

# Disable access to Discovery search and filters; admin pages; processes; submission; workspace; workflow &
profile page
Disallow: /search
Disallow: /admin/*
Disallow: /processes
Disallow: /submit
Disallow: /workspaceitems
Disallow: /profile
Disallow: /workflowitems

# Optionally uncomment the following line ONLY if sitemaps are working
# and you have verified that your site is being indexed correctly.
# Disallow: /browse/*
#
# If you have configured DSpace (Solr-based) Statistics to be publicly
# accessible, then you may not want this content to be indexed
# Disallow: /statistics
#
# You also may wish to disallow access to the following paths, in order
# to stop web spiders from accessing user-based content
# Disallow: /contact
# Disallow: /feedback
# Disallow: /forgot
# Disallow: /login
# Disallow: /register
```

WARNING: for your additional disallow statements to be recognized under the `User-agent: *` group, they *cannot be separated by white lines* from the declared `user-agent: *` block. A white line indicates the start of a new user

agent block. Without a leading user-agent declaration on the first line, blocks are ignored. Comment lines are allowed and will not break the user-agent block.

This is OK:

```
User-agent: *
# Disable access to Discovery search and filters; admin pages; processes
Disallow: /search
Disallow: /admin/*
Disallow: /processes
```

This is **not OK**, as the two lines at the bottom will be completely ignored.

```
User-agent: *
# Disable access to Discovery search and filters; admin pages; processes
Disallow: /search

Disallow: /admin/*
Disallow: /processes
```

To identify if a specific user agent has access to a particular URL, you can use this handy robots.txt tester[412].

For more information on the robots.txt format, please see the Google Robots.txt documentation[413].

5.9.1.6 Ensure Item Metadata appears in the HTML HEAD

It's possible to greatly customize the look and feel of your DSpace, which makes it harder for search engines, and other tools and services such as Zotero[414], Connotea[415] and SIMILE Piggy Bank[416], to correctly pick out item metadata fields. To address this, DSpace includes item metadata in the <head> element of each item's HTML display page.

```
<meta name="DC.type" content="Article" />
<meta name="DCTERMS.contributor" content="Tansley, Robert" />
```

If you have heavily customized your metadata fields away from Dublin Core, you can modify the crosswalk that generates these elements by modifying `[dspace]/config/crosswalks/xhtml-head-item.properties`.

Google Scholar Metadata in HTML HEAD

In addition to Dublin Core <meta> tags in the HTML HEAD, DSpace also includes Google Scholar specific metadata fields in each item's HTML display page.

```
<meta property="citation_authors" content="Tansley, Robert; Donohue, Timothy"/>
<meta property="citation_title" content="Ensuring your DSpace is indexed" />
```

412 http://www.frobee.com/robots-txt-check
413 https://developers.google.com/webmasters/control-crawl-index/docs/robots_txt
414 http://www.zotero.org/
415 http://www.connotea.org/
416 http://simile.mit.edu/wiki/Piggy_Bank

These meta tags are the "Highwire Press tags" which Google Scholar recommends[417]. If you have heavily customized your metadata fields, or wish to change the default "mappings" to these Highwire Press tags, they are configurable in `[dspace]/config/crosswalks/google-metadata.properties`

Much more information is available in the Configuration section on Google Scholar Metadata Mappings[418].

5.9.1.7 Avoid redirecting file downloads to Item landing pages

Make sure that you never redirect "direct file downloads" (i.e. users who directly jump to downloading a file, often from a search engine) to the associated Item's splash/landing page. In the past, some DSpace sites have added these custom URL redirects in order to facilitate capturing statistics via Google Analytics or similar.

While these URL redirects may seem harmless, they may be flagged as cloaking[419] or spam by Google, Google Scholar and other major search engines. This may hurt your site's search engine ranking or even cause your entire site to be flagged for removal from the search engine.

If you have these URL redirects in place, it is highly recommended to remove them immediately. If you created these redirects to facilitate capturing download statistics in Google Analytics, you should consider upgrading to DSpace 5.0 or above, which is able to automatically record bitstream downloads in Google Analytics (see DS-2088[420]) without the need for any URL redirects.

5.9.1.8 Turn OFF any generation of PDF cover pages

While DSpace offers a PDF Citation Cover Page[421] option, this option may affect your content's visibility in search engines like Google Scholar. Google Scholar (and possibly other search engines) specifically extracts metadata by analyzing the contents of the first page of a PDF. Dynamically inserting a custom cover page can break the metadata extraction techniques of Google Scholar and may result in all or much of your site being dropped from the Google Scholar search engine.

For more information, please see the "Indexing Repositories: Pitfalls and Best Practices[422]" talk from Anurag Acharya (co-creator of Google Scholar) presented at the Open Repositories 2015 conference[423].

5.9.1.9 In general, OAI-PMH is not useful to Search Engines

Feel free to support OAI-PMH, but be aware that in general it is not useful for search engines:

- No reliable way to determine OAI-PMH base URL for a DSpace site.
- No standard or predictable way to get to item display page or full text from an OAI-PMH record, making effective indexing and presenting meaningful results difficult.
- In most cases provides only access to simple Dublin Core, a subset of available metadata.
- **NOTE:** Back in 2008, Google officially announced they were retiring support for OAI-PMH based Sitemaps[424]. So, OAI-PMH will no longer help you get better indexing through Google. Instead, you should be using the DSpace 'generate-sitemaps' feature described above.

T

---

417 http://scholar.google.com/intl/en/scholar/inclusion.html#indexing
418 https://wiki.lyrasis.org/display/DSDOC5x/Google+Scholar+Metadata+Mappings
419 https://en.wikipedia.org/wiki/Cloaking
420 https://jira.duraspace.org/browse/DS-2088
421 https://wiki.lyrasis.org/display/DSDOC5x/PDF+Citation+Cover+Page
422 http://www.or2015.net/wp-content/uploads/2015/06/or-2015-anurag-google-scholar.pdf
423 http://www.or2015.net
424 http://googlewebmastercentral.blogspot.com/2008/04/retiring-support-for-oai-pmh-in.html

5.9.2 Google Scholar Metadata Mappings

> While DSpace 7.0 supports Google Scholar `meta` tags, they are no longer configurable & are currently hardcoded into the User Interface codebase. Configurability may be coming back in a later 7.x release (based on user feedback), see https://github.com/DSpace/dspace-angular/issues/1198

Google Scholar, in crawling sites, prefers Highwire Press tags[425]. This schema contains names which are all prefixed by the string "citation_", and provide various metadata about the article/item being indexed.

In DSpace, there is a mapping facility to connect metadata fields with these citation fields in HTML. In order to enable this functionality, the switch needs to be flipped in dspace.cfg:

```
google-metadata.enable = true
```

Once the feature is enabled, the mapping is configured by a separate configuration file located here:

```
[dspace]/config/crosswalks/google-metadata.properties
```

This file contains name/value pairs linking meta-tags with DSpace metadata fields. E.g…

```
google.citation_title = dc.title
google.citation_publisher = dc.publisher
google.citation_author = dc.author | dc.contributor.author | dc.creator
```

There is further documentation in this configuration file explaining proper syntax in specifying which metadata fields to use. If a value is omitted for a meta-tag field, the meta-tag is simply not included in the HTML output.

The values for each item are interpolated when the item is viewed, and the appropriate meta-tags are included in the HTML head tag, on both the Brief Item Display and the Full Item Display in the UI.

Note: In DSpace 5, the field google.citation_authors was changed to google.citation_author.

5.10 Troubleshooting Information

You can quickly get some basic information about the DSpace version and the products supporting it by using the `[dspace]/bin/dspace version` command.

```
$ bin/dspace version
DSpace version:  4.0-SNAPSHOT
 SCM revision:  da53991b6b7e9f86c2a7f5292e3c2e9606f9f44c
   SCM branch:  UNKNOWN
        OS:  Linux(amd64) version 3.7.10-gentoo
Discovery enabled.
Lucene search enabled.
       JRE:  Oracle Corporation version 1.7.0_21
```

425 http://scholar.google.com/intl/en/scholar/inclusion.html#indexing

```
  Ant version:  Apache Ant(TM) version 1.8.4 compiled on June 25 2012
 Maven version:  3.0.4
  DSpace home:  /home/dspace
$
```

To troubleshoot a specific error, see our Troubleshoot an error[426] **guide**

5.11 Validating CheckSums of Bitstreams

5.11.1 Checksum Checker

Checksum Checker is program that can run to verify the checksum of every item within DSpace. Checksum Checker was designed with the idea that most System Administrators will run it from the cron. Depending on the size of the repository choose the options wisely.

Command used:	`[dspace]/bin/dspace checker`
Java class:	org.dspace.app.checker.ChecksumChecker
Arguments short and (long) forms):	Description
`-L` or `--continuous`	Loop continuously through the bitstreams
`-a` or `--handle`	Specify a handle to check
`-b <bitstream-ids>`	Space separated list of bitstream IDs
`-c` or `--count`	Check count
`-d` or `--duration`	Checking duration

426 https://wiki.lyrasis.org/display/DSPACE/Troubleshoot+an+error

`-h` or `--help`	Calls online help
`-l` or `--looping`	Loop once through bitstreams
`-p <prune>`	Prune old results (optionally using specified properties file for configuration
`-v` or `--verbose`	Report all processing

There are three aspects of the Checksum Checker's operation that can be configured:

- the execution mode
- the logging output
- the policy for removing old checksum results from the database
  The user should refer to Chapter 5. Configuration for specific configuration beys in the *dspace.cfg* file.

5.11.1.1 Checker Execution Mode

Execution mode can be configured using command line options. Information on the options are found in the previous table above. The different modes are described below.

Unless a particular bitstream or handle is specified, the Checksum Checker will always check bitstreams in order of the least recently checked bitstream. (Note that this means that the most recently ingested bitstreams will be the last ones checked by the Checksum Checker.)

Available command line options

- **Limited-count mode:** `[dspace]/bin/dspace checker -c` To check a specific number of bitstreams. The *-c* option if followed by an integer, the number of bitstreams to check. Example: `[dspace/bin/dspace checker -c 10` This is particularly useful for checking that the checker is executing properly. The Checksum Checker's default execution mode is to check a single bitstream, as if the option was *-c 1*
- **Duration mode:** `[dspace]/bin/dspace checker -d` To run the Check for a specific period of time with a time argument. You may use any of the time arguments below: Example: `[dspace/bin/dspace checker -d 2h`(Checker will run for 2 hours)

s	Seconds
m	Minutes
h	Hours
d	Days
w	Weeks

y	Years

The checker will keep starting new bitstream checks for the specific durations, so actual execution duration will be slightly longer than the specified duration. Bear this in mind when scheduling checks.

- **Specific Bitstream mode:** `[dspace]/bin/dspace checker -b` Checker will only look at the internal bitstream IDs. Example: `[dspace]/bin/dspace checker -b 112 113 4567` Checker will only check bitstream IDs 112, 113 and 4567.
- **Specific Handle mode:** `[dspace]/bin/dspace checker -a` Checker will only check bitstreams within the Community, Community or the item itself. Example: `[dspace]/bin/dspace checker -a 123456/999` Checker will only check this handle. If it is a Collection or Community, it will run through the entire Collection or Community.
- **Looping mode:** `[dspace]/bin/dspace checker -l` or `[dspace]/bin/dspace checker -L` There are two modes. The lowercase 'el' (-l) specifies to check every bitstream in the repository once. This is recommended for smaller repositories who are able to loop through all their content in just a few hours maximum. An uppercase 'L' (-L) specifies to continuously loops through the repository. This is not recommended for most repository systems. **Cron Jobs**. For large repositories that cannot be completely checked in a couple of hours, we recommend the -d option in cron.
- **Pruning mode:** `[dspace]/bin/dspace checker -p` The Checksum Checker will store the result of every check in the checksum_history table. By default, successful checksum matches that are eight weeks old or older will be deleted when the -p option is used. (Unsuccessful ones will be retained indefinitely). Without this option, the retention settings are ignored and the database table may grow rather large!

5.11.1.2 Checker Results Pruning

As stated above in "Pruning mode", the checksum_history table can get rather large, and that running the checker with the -p assists in the size of the checksum_history being kept manageable. The amount of time for which results are retained in the checksum_history table can be modified by one of two methods:

1. Editing the retention policies in `[dspace]/config/dspace.cfg` See Chapter 5 Configuration for the property keys. OR
2. Pass in a properties file containing retention policies when using the -p option.To do this, create a file with the following two property keys:

```
checker.retention.default = 10y
checker.retention.CHECKSUM_MATCH = 8w
```

You can use the table above for your time units. At the command line: `[dspace]/bin/dspace checker -p retention_file_name <ENTER>`

5.11.1.3 Checker Reporting

Checksum Checker uses log4j to report its results. By default it will report to a log called `[dspace]/log/checker.log`, and it will report only on bitstreams for which the newly calculated checksum does not match the stored checksum. To report on all bitstreams checked regardless of outcome, use the *-v* (verbose) command line option:

`[dspace]/bin/dspace checker -l -v` (This will loop through the repository once and report in detail about every bitstream checked.

To change the location of the log, or to modify the prefix used on each line of output, edit the `[dspace]/config/templates/log4j.properties` file and run `[dspace]/bin/install_configs`.

5.11.1.4 Cron or Automatic Execution of Checksum Checker

You should schedule the Checksum Checker to run automatically, based on how frequently you backup your DSpace instance (and how long you keep those backups). The size of your repository is also a factor. For very large repositories, you may need to schedule it to run for an hour (e.g. `-d 1h` option) each evening to ensure it makes it through your entire repository within a week or so. Smaller repositories can likely get by with just running it weekly.

Unix, Linux, or MAC OS. You can schedule it by adding a cron entry similar to the following to the crontab for the user who installed DSpace:

```
0 4 * * 0 [dspace]/bin/dspace checker -d2h -p
```

The above cron entry would schedule the checker to run the checker every Sunday at 400 (4:00 a.m.) for 2 hours. It also specifies to 'prune' the database based on the retention settings in *dspace.cfg*.

Windows OS. You will be unable to use the checker shell script. Instead, you should use Windows Schedule Tasks to schedule the following command to run at the appropriate times:

```
[dspace]/bin/dspace checker -d2h -p
```

(This command should appear on a single line).

5.11.1.5 Automated Checksum Checkers' Results

Optionally, you may choose to receive automated emails listing the Checksum Checkers' results to the email address specified in the `mail.admin` configuration property. Schedule it to run **after** the Checksum Checker has completed its processing (otherwise the email may not contain all the results). As of DSpace 4.1, an email is only generated if the selected report contains at least one bitstream needing attention.

Command used:	`[dspace]/bin/dspace checker-emailer`
Java class:	org.dspace.checker.DailyReportEmailer
Arguments short and (long) forms):	Description
`-a` or `--All`	Send all the results (everything specified below)
`-d` or `--Deleted`	Send E-mail report for all bitstreams set as deleted for today.
`-m` or `--Missing`	Send E-mail report for all bitstreams not found in assetstore for today.

`-c` or `--Changed`	Send E-mail report for all bitstreams where checksum has been changed for today.
`-u` or `--Unchanged`	Send the Unchecked bitstream report.
`-n` or `--Not Processed`	Send E-mail report for all bitstreams set to longer be processed for today.
`-h` or `--help`	Help

You can also combine options (e.g. `-m -c`) for combined reports.

Cron. Follow the same steps above as you would running checker in cron. Change the time but match the regularity. Remember to schedule this **after** Checksum Checker has run. For an example cron setup, see Scheduled Tasks via Cron(see page 481).

5.11.1.6 Database Query

A query like the following can be used to check the results of the checker (Postgres):

```
SELECT *
FROM checksum_history
WHERE date_trunc('day', process_start_date) = CURRENT_DATE
AND result != 'CHECKSUM_MATCH'
AND result != 'BITSTREAM_MARKED_DELETED';
```

Example of a more detailed query:

```
SELECT
    ch.process_start_date,
    ch.process_end_date,
    ch.result,
    ch.checksum_expected,
    ch.checksum_calculated,
    b.bitstream_id,
    bfr.short_description,
    b.store_number,
    substring(b.internal_id for 2) || '/' || substring(b.internal_id from 3 for 2) || '/' || substring(b.in
ternal_id from 5 for 2) || '/' || b.internal_id AS bitstream_path,
    hi.handle AS item_handle,
    hc.handle AS collection_handle
FROM checksum_history ch
JOIN bitstream b
ON ch.bitstream_id = b.uuid
JOIN bitstreamformatregistry bfr
ON b.bitstream_format_id = bfr.bitstream_format_id
LEFT JOIN bundle2bitstream bb
ON b.uuid = bb.bitstream_id
LEFT JOIN item2bundle ib
ON bb.bundle_id = ib.bundle_id
LEFT JOIN item i
ON ib.item_id = i.uuid
LEFT JOIN handle hi
ON i.uuid = hi.resource_id
AND hi.resource_type_id = 2
LEFT JOIN handle hc
ON i.owning_collection = hc.resource_id
AND hc.resource_type_id = 3
WHERE ch.result != 'CHECKSUM_MATCH'
AND date_trunc('day', process_start_date) = CURRENT_DATE
ORDER BY ch.check_id DESC;
```

6 DSpace Development

This section contains information on how to modify, extend and customize the DSpace source code.

6.1 Advanced Customisation

If you are looking for ways to override specific classes or resources in DSpace (specifically in the backend), this page provides a guide for how to do so.

6.1.1 Additions module

Location: `[dspace-source]/dspace/modules/additions/`

This module may be used to store dspace-api changes, custom plugins, etc. Classes placed in `[dspace-source]/dspace/modules/additions` will override those located in the `[dspace-source]/dspace-api`

This module may be used to override classes across *all* webapps located in `[dspace-source]/dspace/modules/` directory, as well as in the command line interface. Therefore, this modules is for global overrides only. If you have overrides specific to a single webapp, use the "Maven WAR Overlays" option below.

6.1.2 Server Webapp Overlay

Location: `[dspace-source]/dspace/modules/server/`

This module overlay directory allows you to override any classes, resources or files available (by default) in the Server Webapp. This includes overriding files of any of the following source directories:

- `[dspace-source]/dspace-oai/` (Bundled into the Server Webapp as a JAR)
- `[dspace-source]/dspace-rdf/` (Bundled into the Server Webapp as a JAR)
- `[dspace-source]/dspace-server-webapp/` (The Server Webapp itself)
- `[dspace-source]/dspace-sword/` (Bundled into the Server Webapp as a JAR)
- `[dspace-source]/dspace-swordv2/` (Bundled into the Server Webapp as a JAR)

Java classes place in `[dspace-source]/dspace/modules/server/` will override classes (of the same path/name) in any of the above modules.

You can also override resources (i.e. any files under a /src/main/resources/ directory) which are embedded in one of the JARs by putting them under `[dspace-source]/dspace/modules/server/src/main/resources/`. For example, to override the "`[dspace-source]/dspace-oai/src/main/resources/templates/index.twig.html`" file embedded in the `dspace-oai.jar`, you'd place your own version at `[dspace-source]/dspace/modules/server/src/main/resources/templates/index.twig.html`. This results in the resource/

file being copied over into the `WEB-INF/classes/` subdirectory of the "server" webapp, and in that location it will override any file of the same name embedded in a JAR (per Servlet Spec 3.0).

6.1.3 Rest (Deprecated) Webapp Overlay

Location: `[dspace-source]/dspace/modules/rest`

If you have chosen to install the deprecated REST API v6 webapp, you can similar override any classes/files of that separate webapp by just placing those files in the `[dspace-source]/dspace/modules/rest/` directory

6.1.4 DSpace Service Manager

6.1.4.1 Introduction

The DSpace Spring Service Manager supports overriding configuration at many levels.

6.1.4.2 Configuration

Configuring Addons to Support Spring Services

Configuring Addons to support Spring happens at two levels. Default Spring configuration is available in the DSpace JAR or WAR resources directory and allows the addon developer to inject configuration into the service manager at load time. The second level is in the deployed [dspace]/config/spring directory where configurations can be provided on a addon module by addon module basis.

This latter method requires the addon to implement a SpringLoader to identify the location to look for Spring configuration and a place configuration files into that location. This can be seen inside the current [dspace-source]/config/modules/spring.cfg

Configuration Priorities

The ordering of the loading of Spring configuration is the following:

1. configPath = "spring/spring-dspace-applicationContext.xml" relative to the current classpath
2. addonResourcePath = "classpath*:spring/spring-dspace-addon-*-services.xml" relative to the current classpath

3. coreResourcePath = "classpath*:spring/spring-dspace-core-services.xml" relative to the current classpath
4. Finally, an array of SpringLoader API implementations that are checked to verify "config/spring/module" can actually be loaded by its existence on the classpath. The configuration of these SpringLoader API classes can be found in dspace.dir/config/modules/spring.cfg.

Configuring a new Addon

There are 2 ways to create a new Spring addon: a new Spring file can be located in the resources directory or in the configuration [dspace]/config/spring directory. A Spring file can also be located in both of these locations but the configuration directory gets preference and will override any configurations located in the resources directory.

Addon located as resource in jar
In the resources directory of a certain module, a Spring file can be added if it matches the following pattern: "spring/spring-dspace-addon-*-services.xml". An example of this can be found in the dspace-discovery-solr block in the DSpace trunk. (`spring-dspace-addon-discovery-services.xml`)
Wherever this jar is loaded in a Maven module, the Spring files will be processed into services.

Addon located in the [dspace]/config/spring directory
This directory has the following subdirectories in which Spring files can be placed:

- api: when placed in this module the Spring files will always be processed into services (since all of the DSpace modules are dependent on the API).
- discovery: when placed in this module the Spring files will only be processed when the discovery library is present

The reason why there is a separate directory is that if a service cannot be loaded, the kernel will crash and DSpace will not start.

Configuring an additional subdirectory for a custom module

So you need to indeed create a new directory in [dspace]/config/spring. Next you need to create a class that inherits from the "org.dspace.kernel.config.SpringLoader". This class only contains one method named getResourcePaths(). What we do now at the moment is implement this in the following manner:

```
@Override
public String[] getResourcePaths(ConfigurationService configurationService) {
     StringBuffer filePath = new StringBuffer();
     filePath.append(configurationService.getProperty("dspace.dir"));
     filePath.append(File.separator);
     filePath.append("config");
     filePath.append(File.separator);
     filePath.append("spring");
     filePath.append(File.separator);
     filePath.append("{module.name}"); //Fill in the module name in this string
     filePath.append(File.separator);
     try {

          //By adding the XML_SUFFIX here it doesn't matter if there should be some kind of spring.xml.old
file in there it will only load in the active ones.
          return new String[]{new File(filePath.toString()).toURI().toURL().toString() + XML_SUFFIX};
      } catch (MalformedURLException e) {
          return new String[0];
      }
}
```

After the class has been created you will also need to add it to the "spring.springloader.modules" property located in the [dspace]/config/modules/spring.cfg.
The Spring service manager will check this property to ensure that only the interface implementations which it can find the class for are loaded in.

By doing this way we give some flexibility to the developers so that they can always create their own Spring modules and then Spring will not crash when it can't find a certain class.

The Core Spring Configuration

Utilizing Autowiring to minimize configuration complexity.

Please see the following tutorials:

- DSpace Spring Services Tutorial[427]
- The TAO of DSpace Services[428]

Accessing the Services Via Service Locator / Java Code

Please see the following tutorials:

- DSpace Spring Services Tutorial[429]
- The TAO of DSpace Services[430]

6.1.4.3 Architectural Overview

Please see Architectural Overview here: DSpace Services Framework(see page 681)

Service Manager Startup in Webapplications and CLI

Please see the DSpace Services Framework(see page 681)

6.1.4.4 Tutorials

Several good Spring / DSpace Services Tutorials are already available:

- DSpace Spring Services Tutorial[431]
- The TAO of DSpace Services[432]

6.2 REST API

427 https://wiki.lyrasis.org/display/DSPACE/DSpace+Spring+Services+Tutorial
428 https://wiki.lyrasis.org/display/DSPACE/The+TAO+of+DSpace+Services
429 https://wiki.lyrasis.org/display/DSPACE/DSpace+Spring+Services+Tutorial
430 https://wiki.lyrasis.org/display/DSPACE/The+TAO+of+DSpace+Services
431 https://wiki.lyrasis.org/display/DSPACE/DSpace+Spring+Services+Tutorial
432 https://wiki.lyrasis.org/display/DSPACE/The+TAO+of+DSpace+Services

6.2.1 Overview

The REST API for DSpace is provided as part of the "server" webapp (`[dspace-source]/dspace-server-webapp/`). It is available on the `/api/` subpath of that webapp (i.e. `${dspace.server.url}/api/`), though a human browseable/searchable interface (using the HAL Browser[433]) is also available at the root path (i.e. `${dspace.server.url}`).

The REST API only responds in JSON at this time.

6.2.2 REST Contract / Documentation

The REST Contract is maintained in GitHub at https://github.com/DSpace/RestContract/blob/main/README.md

This contract provides detailed information on how to interact with the API, what endpoints are available, etc. All features/capabilities of the DSpace UI are available in this API.

6.2.3 REST Configuration

The following REST API configurations are provided in `[dspace]/config/rest.cfg` and may be overridden in your local.cfg

Property:	`rest.cors.allow-origins`
Example Value:	`rest.cors.allow-origins = ${dspace.ui.url}`

433 https://github.com/mikekelly/hal-browser

Informational Note:	Allowed Cross-Origin-Resource-Sharing (CORS) origins (in "Access-Control-Allow-Origin" header). Only these origins (client URLs) can successfully authenticate with your REST API. Defaults to `${dspace.ui.url}` if unspecified (as the UI must have access to the REST API). If you customize that setting, MAKE SURE TO include `${dspace.ui.url}` in that setting if you wish to continue trusting the UI. Multiple allowed origin URLs may be comma separated (or this configuration can be defined multiple times). Wildcard value (*) is NOT SUPPORTED. Keep in mind any URLs added to this setting must be *an exact match with the origin*: mode (http vs https), domain, port, and subpath(s) all must match.. So, for example, these URLs are all considered different origins: "http://mydspace.edu", "http://mydspace.edu:4000" (different port), "https://mydspace.edu" (http vs https), "https://myapp.mydspace.edu" (different domain), and "https://mydspace.edu/myapp" (different subpath). *NOTE:* If you modify this value to allow additional UIs to access your REST API, then you may also need to modify `proxies.trusted.ipranges` to trust the IP address of each UI. Modifying trusted proxies is only necessary if the `X-FORWARDED-FOR` header must be trusted from each additional UIs. (The DSpace UI currently requires the `X-FORWARDED-FOR` header to be trusted). By default, `proxies.trusted.ipranges` will only trust the IP address of the `${dspace.ui.url}` configuration. (Requires reboot of servlet container, e.g. Tomcat, to reload)
Property:	`rest.cors.allow-credentials`
Example Value:	`rest.cors.allow-credentials = true`
Informational Note:	Whether or not to allow credentials (e.g. cookies) sent by the client/browser in CORS requests (in "Access-Control-Allow-Credentials" header). For DSpace, this MUST be set to "true" to support CSRF checks (which use Cookies) and external authentication via Shibboleth (and similar). Defaults to "true" if unspecified. (Requires reboot of servlet container, e.g. Tomcat, to reload)
Property:	`rest.projections.full.max`
Example Value:	`rest.projections.full.max = 2`
Informational Note:	This property determines the max embeddepth for a FullProjection. This is also used by the SpecificLevelProjection as a fallback in case the property is defined on the bean. Usually, this should be kept as-is for best performance.

Property:	`rest.projection.specificLevel.maxEmbed`
Example Value:	`rest.projection.specificLevel.maxEmbed = 5`
Informational Note:	This property determines the max embed depth for a SpecificLevelProjection. Usually, this should be kept as-is for best performance.
Property:	`rest.properties.exposed`
Example Value:	`rest.properties.exposed = plugin.named.org.dspace.curate.CurationTask` `rest.properties.exposed = google.analytics.key`
Informational Note:	Define which configuration properties are exposed through the `http://<dspace.server.url>/api/config/properties/` REST API endpoint. If a rest request is made for a property which exists, but isn't listed here, the server will respond that the property wasn't found. This property can be defined multiple times to allow access to multiple configuration properties. Generally, speaking, it is ONLY recommended to expose configuration settings where they are necessary for the UI or client, as exposing too many configurations could be a security issue. This is why we only expose the two above settings by default.

6.2.4 Technical Design

The REST API & Server Webapp are built on Spring Boot[434] and Spring HATEOAS[435], using Spring Security[436]. It also aligns with Spring Data REST[437] (though at this time it doesn't use it directly because of incompatibility with the DSpace data model).

The REST API is stateless, aligns with HATEOAS (Hypertext as the Engine of Application State)[438] principles, returning HAL formatted JSON[439]. This allows the REST API to be easily browsable/interactable via third-party tools that understand HAL & HATEOAS, such as the HAL Browser[440]. JSON Web Tokens (JWT)[441] are used to store state/session information between requests.

For better security, the REST API requires usage of CSRF tokens[442] for all modifying requests.

434 https://spring.io/projects/spring-boot
435 https://spring.io/projects/spring-hateoas
436 https://spring.io/projects/spring-security
437 https://spring.io/projects/spring-data-rest
438 https://en.wikipedia.org/wiki/HATEOAS
439 http://stateless.co/hal_specification.html
440 https://github.com/mikekelly/hal-browser
441 https://jwt.io/
442 https://github.com/DSpace/RestContract/blob/main/csrf-tokens.md

More information can be found in the REST Contract at https://github.com/DSpace/RestContract/blob/main/README.md#rest-design-principles

6.3 REST API v6 (deprecated)

This documentation describes the old, deprecated REST API

This documentation describes the deprecated DSpace v4-6 REST API. This old API is still available in DSpace 7, but will be removed in DSpace 8.

We highly recommend all users migrate scripts/tools to use the new REST API(see page 502)**.** This API is no longer actively supported or maintained.

6.3.1 What is DSpace REST API (v4-v6)

The REST API module provides a programmatic interface to DSpace Communities, Collections, Items, and Bitstreams.

DSpace 4 introduced the initial REST API, which did not allow for authentication, and provided only READ-ONLY access to publicly accessible Communities, Collections, Items, and Bitstreams. DSpace 5 builds off of this and allows authentication to access restricted content, as well as allowing Create, Edit and Delete on the DSpace Objects. DSpace 5 REST API also provides improved pagination over resources and searching. There has been a minor drift between the DSpace 4 REST API and the DSpace 5 REST API, so client applications will need to be targeted per version.

6.3.1.1 Installing the REST API (v4-v6)

The REST API deploys as a standard webapp for your servlet container / tomcat. For example, depending on how you deploy webapps, one way would be to alter tomcat-home/conf/server.xml and add:

```
<Context path="/rest"  docBase="/dspace/webapps/rest" />
```

In DSpace 4, the initial/official Jersey-based REST API was added to DSpace. The DSpace 4 REST API provides READ-ONLY access to DSpace Objects.

In DSpace 5, the REST API adds authentication, allows Creation, Update, and Delete to objects, can access restricted materials if authorized, and it requires SSL.

Disabling SSL

For localhost development purposes, SSL can add additional getting-started difficulty, so security can be disabled. To disable DSpace REST's requirement to require security/ssl, alter `[dspace]/webapps/rest/WEB-INF/web.xml` or `[dspace-source]/dspace-rest/src/main/webapp/WEB-INF/web.xml` and comment out the `<security-constraint>` block, and restart your servlet container. Production usages of the REST API should use SSL, as authentication credentials should not go over the internet unencrypted.

6.3.1.2 REST Endpoints

The REST API is modeled after the DSpace Objects of Communities, Collections, Items, and Bitstreams. The API is not a straight database schema dump of these entities, but provides some wrapping that makes it easy to follow relationships in the API output.

> **HTTP Header: Accept**
>
> Note: You must set your request header's "Accept" property to either JSON (application/json) or XML (application/xml) depending on the format you prefer to work with.

Example usage from command line in XML format with pretty printing:

```
curl -s -H "Accept: application/xml" http://localhost:8080/rest/communities | xmllint --format -
```

Example usage from command line in JSON format with pretty printing:

```
curl -s -H "Accept: application/json" http://localhost:8080/rest/communities | python -m json.tool
```

For this documentation, we will assume that the URL to the "REST" webapp will be http://localhost:8080/rest/ for production systems, this address will be slightly different, such as: https://demo.dspace.org/rest/. The path to an endpoint, will go after the /rest/, such as /rest/communities, all-together this is: http://localhost:8080/rest/communities

Another thing to note is that there are Query Parameters that you can tack on to the end of an endpoint to do extra things. The most commonly used one in this API is "?expand". Instead of every API call defaulting to giving you every possible piece of information about it, it only gives a most commonly used set by default and gives the more "expensive" information when you deliberately request it. Each endpoint will provide a list of available expands in the output, but for getting started, you can start with ?expand=all, to make the endpoint provide all of its

information (parent objects, metadata, child objects). You can include multiple expands, such as: ?expand=collections,subCommunities .

Two other query parameters of note are `limit` and `offset`. `Endpoints which return arrays of objects, such as` /communities, are "paginated": the full list is broken into "pages" which start at `offset` from the beginning of the list and contain at most `limit` elements. By repeated queries you can retrieve any portion of the array or all of it. Offsets begin at zero. So, to retrieve the sixth through tenth elements of the full list of Collections, you could do this:

```
curl -s -H "Accept: application/json" http://localhost:8080/rest/collections?offset=5\&limit=5
```

Index / Authentication

⚠ REST API Authentication has changed in DSpace 6.x. It now uses a `JSESSIONID` cookie (see below). The previous (5.x) authentication scheme using a `rest-dspace-token` is no longer supported.

Method	Endpoint	Description
GET	/	REST API static documentation page

Method	Endpoint	Description
POST	/login	Login to the REST API using a DSpace EPerson (user). It returns a `JSESSIONID` cookie, that can be used for future authenticated requests. *Example Request:* `# Can use either POST or GET (POST recommended). Must pass the parameters "email" and "password".` `curl -v -X POST --data "email=admin@dspace.org&password=mypass" https://dspace.myu.edu/rest/login` *Example Response:* `HTTP/1.1 200 OK` `Set-Cookie: JSESSIONID=6B98CF8648BCE57DCD99689FE77CB1B8; Path=/rest/; Secure; HttpOnly` *Example of using JSESSIONID cookie for subsequent (authenticated) requests:* `curl -v --cookie "JSESSIONID=6B98CF8648BCE57DCD99689FE77CB1B8" https://dspace.myu.edu/rest/status` `# This should return <authenticated>true</authenticated>, and information about the authenticated user session` Invalid email/password combinations will receive an `HTTP 401 Unauthorized` response. *Please note, special characters need to be HTTP URL encoded.* *For example, an email address like `dspacedemo+admin@gmail.com` (notice the + special character) would need to be encoded as `dspacedemo%2Badmin@gmail.com`.*

Method	Endpoint	Description
GET	/shibboleth-login	Login to the REST API using Shibboleth authentication. In order to work, this requires additional Apache configuration(see page 0). To authenticate, execute the following steps: 1. Call the REST Shibboleth login point with a Cookie jar: `curl -v -L -c cookiejar "https://dspace.myu.edu/rest/shibboleth-login"` 2. This should take you again to the IdP login page. You can submit this form using curl using the same cookie jar. However this is IdP dependant so we cannot provide an example here. 3. Once you submit the form using curl, you should be taken back to the /rest/shibboleth-login URL which will return you the JSESSIONID. 4. Using that JSESSIONID, check if you have authenticated successfully: `curl -v "http://localhost:8080/dspace-rest/status" --cookie "JSESSIONID=0633C6379266A283E53F65DF8EF61AB9"`

Method	Endpoint	Description
POST	/logout	Logout from the REST API, by providing a `JSESSIONID` cookie. After being posted this cookie will no longer work. *Example Request:* `curl -v -X POST --cookie "JSESSIONID=6B98CF8648BCE57DCD99689FE77CB1B8" https://dspace.myu.edu/rest/logout` After posting a logout request, cookie is invalidated and the "/status" path should show you as unauthenticated (even when passing that same cookie). For example: `curl -v --cookie "JSESSIONID=6B98CF8648BCE57DCD99689FE77CB1B8" https://dspace.myu.edu/rest/status # This should show <authenticated>false</authenticated>` Invalid token will result in HTTP 400 Invalid Request
GET	/test	Returns string "REST api is running", for testing that the API is up. *Example Request:* `curl https://dspace.myu.edu/rest/test` *Example Response:* `REST api is running.`

Method	Endpoint	Description
GET	/status	Receive information about the currently authenticated user token, or the API itself (e.g. version information). *Example Request (XML by default)*: `curl -v --cookie "JSESSIONID=6B98CF8648BCE57DCD99689FE77CB1B8" https://dspace.myu.edu/rest/status` *Example Request (JSON):* `curl -v -H "Accept: application/json" --cookie "JSESSIONID=6B98CF8648BCE57DCD99689FE77CB1B8" https://dspace.myu.edu/rest/status` *Example JSON Response:* `{ "okay":true, "authenticated":true, "email":"admin@dspace.org", "fullname":"DSpace Administrator", "sourceVersion":"6.0", "apiVersion":"6" }`

Shibboleth Apache configuration for the REST API

Before Shibboleth authentication for the REST API will work, you need to secure the `/rest/shibboleth-login` endpoint. Add this configuration section to your Apache HTTPD Shibboleth configuration:

```
<Location "/rest/shibboleth-login">
        AuthType shibboleth
        ShibRequireSession On
        # Please note that setting ShibUseHeaders to "On" is a potential security risk.
        # You may wish to set it to "Off". See the mod_shib docs for details about this setting:
        # https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPApacheConfig#NativeSPApacheConfig-
AuthConfigOptions
        # Here's a good guide to configuring Apache + Tomcat when this setting is "Off":
        # https://www.switch.ch/de/aai/support/serviceproviders/sp-access-rules.html#javaapplications
        ShibUseHeaders On
        require valid-user
</Location>
```

You can test your configuration in 3 different ways:

1. Using a web browser:
    a. Go to `https://dspace.myu.edu/rest/shibboleth-login`, this should redirect you to the login page of your IdP if you don't have a Shibboleth session yet.
    b. Enter your test credentials and this should take you back to the `/rest/shibboleth-login` URL. You should then see a blank page but in the response headers, the JSESSIONID cookie should be present.
    c. Then go to `/rest/status` and you should see information on the current authenticated ePerson.
2. Using curl without a Shibboleth Session
    a. Call the REST Shibboleth login point with a Cookie jar:

```
curl -v -L -c cookiejar "https://dspace.myu.edu/rest/shibboleth-login"
```

    b. This should take you again to the IdP login page. You can submit this form using curl using the same cookie jar. However this is IdP dependant so I cannot provide an example here.
    c. Once you submit the form using curl, you should be taken back to the `/rest/shibboleth-login` URL which will return you the JSESSIONID.
    d. Using that JSESSIONID, check if you have authenticated successfully:

```
curl -v "https://dspace.myu.edu/dspace-rest/status" --cookie
"JSESSIONID=0633C6379266A283E53F65DF8EF61AB9"
```

3. Using curl with a Shibboleth Session (cookie)
    a. When you post the Shibboleth login form, the Shibboleth daemon on the **DSpace server** also returns you a Shibboleth Cookie. This cookie looks like `_shibsession_64656661756c74687...` You can also grab this cookie from your browser.
    b. Double check that the cookie you took is valid:

```
curl -v 'https://dspace-url/Shibboleth.sso/Session' -H 'Cookie:
_shibsession_64656661756c7468747470733a2f2f7265706f7369746f72792e6361726469666666d65742e61632e
756b2f73686962626f6c657468=_a8d3ad20d8b655250c7357f7ac0e2910;'
```

    c. This should give you information if the Shibboleth session is valid and on the number of attributes.
    d. Use this cookie to obtain a Tomcat JSESSIONID:

```
curl -v 'https://dspace-url/rest/shibboleth-login' -H 'Cookie:
_shibsession_64656661756c7468747470733a2f2f7265706f7369746f72792e6361726469666666d65742e61632e
756b2f73686962626f6c657468=_a8d3ad20d8b655250c7357f7ac0e2910;'
```

    e. Use the returned JSESSIONID to check if you have authenticated successfully:

```
curl -v "http://dspace-url/rest/status" --cookie
"JSESSIONID=0633C6379266A283E53F65DF8EF61AB9"
```

Communities

Communities in DSpace are used for organization and hierarchy, and are containers that hold sub-Communities and Collections. (ex: Department of Engineering)

- GET /communities - Returns array of all communities in DSpace.
- GET /communities/top-communities - Returns array of all top communities in DSpace.
- GET /communities/{communityId} - Returns community.
- GET /communities/{communityId}/collections - Returns array of collections of community.
- GET /communities/{communityId}/communities - Returns array of subcommunities of community.
- POST /communities - Create new community at top level. You must post community.
- POST /communities/{communityId}/collections - Create new collections in community. You must post Collection.
- POST /communities/{communityId}/communities - Create new subcommunity in community. You must post Community.
- PUT /communities/{communityId} - Update community. You must put Community
- DELETE /communities/{communityId} - Delete community.
- DELETE /communities/{communityId}/collections/{collectionId} - Delete collection in community.
- DELETE /communities/{communityId}/communities/{communityId2} - Delete subcommunity in community.

Collections

Collections in DSpace are containers of Items. (ex: Engineering Faculty Publications)

- GET /collections - Return all collections of DSpace in array.
- GET /collections/{collectionId} - Return collection with id.
- GET /collections/{collectionId}/items - Return all items of collection.
- POST /collections/{collectionId}/items - Create posted item in collection. You must post an Item
- POST /collections/find-collection - Find collection by passed name.
- PUT /collections/{collectionId} - Update collection. You must put Collection.
- DELETE /collections/{collectionId} - Delete collection from DSpace.
- DELETE /collections/{collectionId}/items/{itemId} - Delete item in collection.

Items

Items in DSpace represent a "work" and combine metadata and files, known as Bitstreams.

- GET /items - Return list of items.
- GET /items/{item id} - Return item.
- GET /items/{item id}/metadata - Return item metadata.
- GET /items/{item id}/bitstreams - Return item bitstreams.
- POST /items/find-by-metadata-field - Find items by metadata entry. You must post a MetadataEntry.
- POST /items/{item id}/metadata - Add metadata to item. You must post an array of MetadataEntry.
- POST /items/{item id}/bitstreams - Add bitstream to item. You must post a Bitstream.
- PUT /items/{item id}/metadata - Update metadata in item. You must put a MetadataEntry.
- DELETE /items/{item id} - Delete item.
- DELETE /items/{item id}/metadata - Clear item metadata.
- DELETE /items/{item id}/bitstreams/{bitstream id} - Delete item bitstream.

Bitstreams

Bitstreams are files. They have a filename, size (in bytes), and a file format. Typically in DSpace, the Bitstream will the "full text" article, or some other media. Some files are the actual file that was uploaded (tagged with bundleName:ORIGINAL), others are DSpace-generated files that are derivatives or renditions, such as text-extraction, or thumbnails. You can download files/bitstreams. DSpace doesn't really limit the type of files that it takes in, so this could be PDF, JPG, audio, video, zip, or other. Also, the logo for a Collection or a Community, is also a Bitstream.

- GET /bitstreams - Return all bitstreams in DSpace.

- GET /bitstreams/{bitstream id} - Return bitstream.
- GET /bitstreams/{bitstream id}/policy - Return bitstream policies.
- GET /bitstreams/{bitstream id}/retrieve - Return data of bitstream.
- POST /bitstreams/{bitstream id}/policy - Add policy to item. You must post a ResourcePolicy
- PUT /bitstreams/{bitstream id}/data - Update data/file of bitstream. You must put the data
- PUT /bitstreams/{bitstream id} - Update metadata of bitstream. You must put a Bitstream, does not alter the file/data
- DELETE /bitstreams/{bitstream id} - Delete bitstream from DSpace.
- DELETE /bitstreams/{bitstream id}/policy/{policy_id} - Delete bitstream policy.

You can access the parent object of a Bitstream (normally an Item, but possibly a Collection or Community when it is its logo) through: /bitstreams/:bitstreamID?expand=parent

As the documentation may state "You must post a ResourcePolicy" or some other object type, this means that there is a structure of data types, that your XML or JSON must be of type, when it is posted in the body.

Handle

In DSpace, Communities, Collections, and Items typically get minted a Handle Identifier. You can reference these objects in the REST API by their handle, as opposed to having to use the internal item-ID.

- GET /handle/{handle-prefix}/{handle-suffix} - Returns a Community, Collection, or Item object that matches that handle.

Hierarchy

Assembling a full representation of the community and collection hierarchy using the communities and collections endpoints can be inefficient. Retrieve a lightweight representation of the nested community and collection hierarchy. Each node of the hierarchy contains minimal information (id, handle, name).

- GET /hierarchy - Retrieve a lightweight representation of the nested community and collection hierarchy.

Schema and Metadata Field Registry

- GET /registries/schema - Return the list of schemas in the registry
- GET /registries/schema/{schema_prefix} - Returns the specified schema
- GET /registries/schema/{schema_prefix}/metadata-fields/{element} - Returns the metadata field within a schema with an unqualified element name
- GET /registries/schema/{schema_prefix}/metadata-fields/{element}/{qualifier} - Returns the metadata field within a schema with a qualified element name
- POST /registries/schema/ - Add a schema to the schema registry
- POST /registries/schema/{schema_prefix}/metadata-fields - Add a metadata field to the specified schema
- GET /registries/metadata-fields/{field_id} - Return the specified metadata field
- PUT /registries/metadata-fields/{field_id} - Update the specified metadata field
- DELETE /registries/metadata-fields/{field_id} - Delete the specified metadata field from the metadata field registry
- DELETE /registries/schema/{schema_id} - Delete the specified schema from the schema registry

Note: since the schema object contains no data fields, the following method has not been implemented: PUT / registries/schema/{schema_id}

Report Tools

Reporting Tools that allow a repository manager to audit a collection for metadata consistency and bitstream consistency. See REST Based Quality Control Reports(see page 518) for more information.

Collection Report Tool on demo.dspace.org[443]

Metadata Query Tool on demo.dspace.org[444]

- GET /reports - Return a list of report tools built on the rest api
- GET /reports/{nickname} - Return a redirect to a specific report
- GET /filters - Return a list of use case filters available for quality control reporting
- GET /filtered-collections - Return collections and item counts based on pre-defined filters
- GET /filtered-collections/{collection_id} - Return items and item counts for a collection based on pre-defined filters
- GET /filtered-items - Retrieve a set of items based on a metadata query and a set of filters

6.3.1.3 Model - Object data types

Here are all of the data types, not all fields are necessary or supported when posting/putting content, but the output contains this information:

Community Object

{"id":456,"name":"Reports Community","handle":"10766/10213","type":"community","link":"/rest/communities/456","expand":["parentCommunity","collections","subCommunities","logo","all"],"logo":null,"parentCommunity":null,"copyrightText":"","introductoryText":"","shortDescription":"Collection contains materials pertaining to the Able Family","sidebarText":"","countItems":3,"subcommunities":[],"collections":[]}

Collection Object

{"id":730,"name":"Annual Reports Collection","handle":"10766/10214","type":"collection","link":"/rest/collections/730","expand":["parentCommunityList","parentCommunity","items","license","logo","all"],"logo":null,"parentCommunity":null,"parentCommunityList":[],"items":[],"license":null,"copyrightText":"","introductoryText":"","shortDescription":"","sidebarText":"","numberItems":3}

Item Object

{"id":14301,"name":"2015 Annual Report","handle":"123456789/13470","type":"item","link":"/rest/items/14301","expand":["metadata","parentCollection","parentCollectionList","parentCommunityList","bitstreams","all"],"lastModified":"2015-01-12 15:44:12.978","parentCollection":null,"parentCollectionList":null,"parentCommunityList":null,"bitstreams":null,"archived":"true","withdrawn":"false"}

Bitstream Object

443 https://demo.dspace.org/rest/static/reports/index.html
444 https://demo.dspace.org/rest/static/reports/query.html

{"id":47166,"name":"appearance and physiology 100 percent copied from wikipedia.pdf","handle":null,"type":"bitstream","link":"/rest/bitstreams/47166","expand":["parent","policies","all"],"bundleName":"ORIGINAL","description":"","format":"Adobe PDF","mimeType":"application/pdf","sizeBytes":129112,"parentObject":null,"retrieveLink":"/bitstreams/47166/retrieve","checkSum":{"value":"62778292a3a6dccbe2662a2bfca3b86e","checkSumAlgorithm":"MD5"},"sequenceId":1,"policies":null}

ResourcePolicy Object

[{"id":317127,"action":"READ","epersonId":-1,"groupId":0,"resourceId":47166,"resourceType":"bitstream","rpDescription":null,"rpName":null,"rpType":"TYPE_INHERITED","startDate":null,"endDate":null}]

MetadataEntry Object

{"key":"dc.description.abstract", "value":"This is the description abstract", "language": null}

User Object

{"email":"test@dspace.org","password":"pass"}

Status Object

{"okay":true,"authenticated":true,"email":"test@dspace.org","fullname":"DSpace Test User","token":"6d45daaa-7b02-4ae7-86de-a960838fae5c"}

6.3.2 Introduction to Jersey for developers

The REST API for DSpace is implemented using Jersey, the reference implementation of the Java standard for building RESTful Web Services (JAX-RS 1). That means this API should be easier to expand and maintain than other API approaches, as this approach has been widely adopted in the industry. If this client documentation does not fully answer about how an endpoint works, it is helpful to look directly at the Java REST API code[445], to see how it is implemented. The code typically has required parameters, optional parameters, and indicates the type of data that will be responded.

There was no central ProviderRegistry that you have to declare your path. Instead, the code is driven by annotations, here is a list of annotations used in the code for CommunitiesResource.java:

- @Path("/communities"), which then allows it to be routed to http://localhost:8080/communities, this is then the base path for all the requests within this class.
- @GET, which indicates that this method responds to GET http requests
- @POST, which indicates that this method responds to POST http requests
- @PUT, which indicates that this method responds to PUT http requests
- @DELETE, which indicates that this method responds to DELETE http requests
- @Path("/{community_id}"), the path is appended to the class level @Path above, this one uses a variable {community_id}. The total endpoint would be http://localhost:8080/rest/communities/123, where 123 is the ID.
- @Consumes({ MediaType.APPLICATION_JSON, MediaType.APPLICATION_XML }), this indicates that this request expects input of either JSON or XML. Another endpoint accepts HTML input.
- @PathParam("community_id") Integer communityId, this maps the path placeholder variable {community_id} to Java int communityID

445 https://github.com/DSpace/DSpace/tree/master/dspace-rest/src/main/java/org/dspace/rest

- @QueryParam("userIP") String user_ip, this maps a query param like ?userIP=8.8.4.4 to Java String user_id variable, and user_id == "8.8.4.4"

6.3.3 Configuration for DSpace REST

Property	rest.stats
Example Value	true
Informational Note	Boolean value indicates whether statistics should be recorded for access via the REST API; Defaults to 'false'.

6.3.4 Recording Proxy Access by Tools

For the purpose of more accurate statistics, a web-based tool may specify who is using it, by adding parameters to the request:

```
http://localhost:8080/rest/items/:ID?userIP=ip&userAgent=userAgent&xforwardedfor=xforwardedfor
```

If no parameters are given, the details of the HTTP request's sender are used in statistics. This enables tools to record the details of their user rather than themselves.

6.3.5 Additional Information

Additional information can be found in the README for dspace-rest[446], and in the GitHub Pull Request for DSpace REST (Jersey)[447].

Usage examples can be found at: https://github.com/BrunoNZ/dspace-rest-requests

6.3.6 REST Based Quality Control Reports

446 https://github.com/DSpace/DSpace/tree/master/dspace-rest
447 https://github.com/DSpace/DSpace/pull/323

- Enable User Authentication (Password AuthN only) for REST reports(see page 522)
- Configure the database-specific format for a regex expression(see page 522)
- Configure the sets of filters of interest to your repository managers(see page 523)
- Other filter configuration settings(see page 523)
- Enabling Sort-able Report Tables(see page 523)
- Installing in DSpace 5(see page 524)

DSpace 7.0 only supports this when using the older, deprecated REST API v6

In DSpace 7.0, REST Quality Control Reports are currently only supported if you also install the old REST API v6 (deprecated)(see page 506) webapp. Tentative plans to migrate these reports to support the new REST API(see page 502) have begun in https://jira.lyrasis.org/browse/DS-4301.

6.3.6.1 Tutorial

DSpace REST Report Tool Tutorial

The following repository contains a tutorial demonstrating the usage of the REST Base Report Tools: https://github.com/terrywbrady/restReportTutorial/blob/master/README.md

6.3.6.2 Summary

These reports utilize the DSpace REST API to provide a Collection Manager with

- an overview of their collections
- a tool to query metadata for consistency

When deploying the DSpace REST API, and institution may choose to make the API publicly accessible or to restrict access to the API.

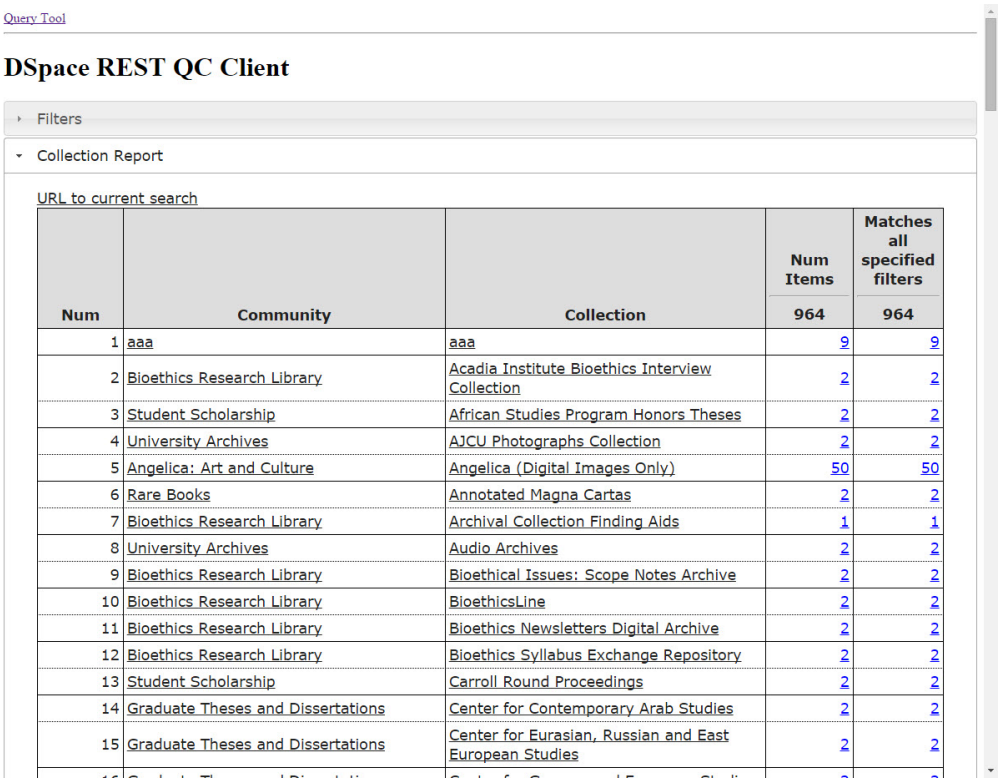
If these reports are deployed in a protected manner, the reporting tools can be configured to bypass DSpace authorization when reporting on collections and items.

6.3.6.3 API Calls Used in these Reports

REST Reports - Summary of API Calls(see page 537)

6.3.6.4 Report Screen Shots

Collection QC Report



Query Tool

DSpace REST QC Client

Filters

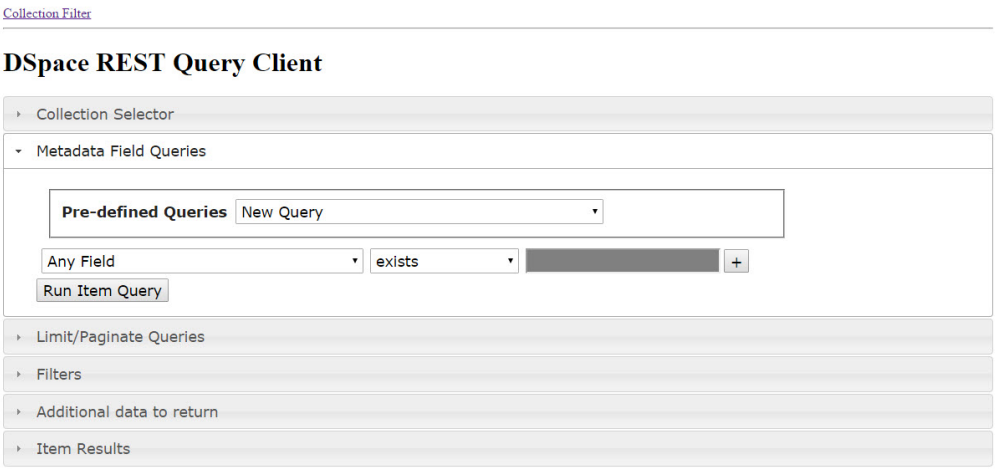
Collection Report

URL to current search

Num	Community	Collection	Num Items	Matches all specified filters
			964	964
1	aaa	aaa	9	9
2	Bioethics Research Library	Acadia Institute Bioethics Interview Collection	2	2
3	Student Scholarship	African Studies Program Honors Theses	2	2
4	University Archives	AJCU Photographs Collection	2	2
5	Angelica: Art and Culture	Angelica (Digital Images Only)	50	50
6	Rare Books	Annotated Magna Cartas	2	2
7	Bioethics Research Library	Archival Collection Finding Aids	1	1
8	University Archives	Audio Archives	2	2
9	Bioethics Research Library	Bioethical Issues: Scope Notes Archive	2	2
10	Bioethics Research Library	BioethicsLine	2	2
11	Bioethics Research Library	Bioethics Newsletters Digital Archive	2	2
12	Bioethics Research Library	Bioethics Syllabus Exchange Repository	2	2
13	Student Scholarship	Carroll Round Proceedings	2	2
14	Graduate Theses and Dissertations	Center for Contemporary Arab Studies	2	2
15	Graduate Theses and Dissertations	Center for Eurasian, Russian and East European Studies	2	2

REST Reports - Collection Report Screenshots with Annotated API Calls(see page 525)

Metadata Query Report



Collection Filter

DSpace REST Query Client

Collection Selector

Metadata Field Queries

Pre-defined Queries New Query

Any Field exists +

Run Item Query

Limit/Paginate Queries

Filters

Additional data to return

Item Results

REST Reports - Metadata Query Screenshots with Annotated API Calls(see page 532)

6.3.6.5 Installation and Configuration

Installing in DSpace 6

This code is part of the DSpace 6 code base.

Disabling the REST Reports

The REST reports will be enabled by default in DSpace 6. To disable the execution of these reports, remove the following line from dspace-rest/src/main/webapp/WEB-INF/web.xml

Enable/disable report resources in the REST API

```
<servlet-mapping>
        <servlet-name>default</servlet-name>
        <url-pattern>/static/*</url-pattern>
    </servlet-mapping>
```

Configuring Access of the Reporting Tools

The reports can be configured with anonymous access or the reports can be configured to bypass authorization checks.

Bypassing authorization checks allows collection owners to view the status of all items in the repository without authenticating through the REST API. This option is recommended **if you have secured access to your REST API**.

If your REST API is publicly accessible, deploy the reports with anonymous access and consider providing an authorization token for access to the report calls.

Configure Authorization for REST Reports

```
# Enable/disable authorization for the reporting tools.
# By default, the DSpace REST API will only return communities/collections/items that are accessible to a
particular user.
# If the REST API has been deployed in a protected manner, the reporting tools can be configured to bypass
authorization checks.
# This will allow all items/collections/communities to be returned to the report user.
# Set the rest-reporting-authenticate option to false to bypass authorization
rest.reporting-authenticate = false
```

Configure the REST Reports that can be requested by name

```
# Configure the report pages that can be requested by name
# Create a map of named reports that are available to a report tool user
# Each map entry should be prefixed with rest-report-url
#   The map key is a name for a report
#   The map value is a URL to a report page
# A list of available reports will be available with the call /rest/reports.
# If a request is sent to /rest/reports/[report key], the request will be re-directed to the specified URL
#
# This project currently contains 2 sample reports.  Eventually, additional reports could be introduced
through this mechanism.
rest.report-url.collections = /rest/static/index.html
rest.report-url.item-query = /rest/static/query.html
```

Configure Item handle resolution

Enable the appropriate path to use to resolve an item handle **restReport.js**. (Depends on https://github.com/DSpace/DSpace/pull/1366/files)

Item Handle Resolution

```
this.ROOTPATH = "/handle/"
```

Enable User Authentication (Password AuthN only) for REST reports

Override the following function in your report file to enable/disable password AuthN for the REST reports. (Depends on https://github.com/DSpace/DSpace/pull/1369)

This setting can be found in **restReport.js**

Enable/Disable Password AuthN

```
//disable this setting if Password Authentication is not supported
this.makeAuthLink = function(){return true;};
```

Configure the database-specific format for a regex expression

```
# The REST Report Tools may pass a regular expression test to the database.
# The following configuration setting will construct a SQL regular expression test appropriate to your
database engine
rest.regex-clause = text_value ~ ?
```

Configure the sets of filters of interest to your repository managers

```
# A filter contains a set of tests that will be applied to an item to determine its inclusion in a
particular report.
# Private items and withdrawn items are frequently excluded from DSpace reports.
# Additional filters can be configured to examine other item properties.
# For instance, items containing an image bitstream often have different requirements from a item
containing a PDF.
# The DSpace REST reports come with a variety of filters that examine item properties, item bitstream
properties,
# and item authorization policies.  The existing filters can be used as an example to construct institution
specific filters
# that will test conformity to a set of institutional policies.
# plugin.sequence.org.dspace.rest.filter points to a list of classes that contain available filters.
# Each class must implement the ItemFilterList interface.
#   ItemFilterDefs:     Filters that examine simple item and bitstream type properties
#   ItemFilterDefsMisc: Filters that examine bitstream mime types and dependencies between bitstreams
#   ItemFilterDefsMeta: Filters that examine metadata properties
#   ItemFilterDefsPerm: Filters that examine item and bitstream authorization policies
plugin.sequence.org.dspace.rest.filter.ItemFilterList = \
        org.dspace.rest.filter.ItemFilterDefs,\
        org.dspace.rest.filter.ItemFilterDefsMisc,\
        org.dspace.rest.filter.ItemFilterDefsPerm

#     org.dspace.rest.filter.ItemFilterDefsMeta,\
```

Other filter configuration settings

The configuration file contains other settings that will control the behavior of the filters that you have enabled.

Enabling Sort-able Report Tables

1. Install sortable.js http://www.kryogenix.org/code/browser/sorttable/
2. Add to /dspace/modules/rest/src/main/webapp/static/reports
3. Include sortable.js in index.html and query.html

Uncomment the following in index.html and query.html

```
<!-- <script src="sorttable.js"></script> -->
```

4. Enable sortable in the report code in restCollReport.js and restQueryReport.js

CHANGE

```
var CollReport = function() {
    Report.call(this);
    //If sortable.js is included, uncomment the following
    //this.hasSorttable = function(){return true;}

var QueryReport = function() {
    Report.call(this);
    //If sortable.js is included, uncomment the following
    //this.hasSorttable = function(){return true;}
```

CHANGE TO

```
var CollReport = function() {
    Report.call(this);
    //If sortable.js is included, uncomment the following
    this.hasSorttable = function(){return true;}

var QueryReport = function() {
    Report.call(this);
    //If sortable.js is included, uncomment the following
    this.hasSorttable = function(){return true;}
```

Installing in DSpace 5

This feature is not a part of the DSpace 5 code base. Please see the following notes to enable a DSpace 5 compatible version of these reports.

1. Install https://github.com/DSpace/DSpace/pull/1568
2. Change the following code into restCollReport.js and restQuery.js to pull the correct id for each DSpace Object

Change the following in restCollReport.js and restQuery.js

```
var CollReport = function() {
Report.call(this);
var QueryReport = function() {
Report.call(this);
```

Change TO

```
var CollReport = function() {
Report.call(this);
this.getId = function(obj) {return obj.id;}
var QueryReport = function() {
Report.call(this);
this.getId = function(obj) {return obj.id;}
```

6.3.6.6 REST Reports - Collection Report Screenshots with Annotated API Calls

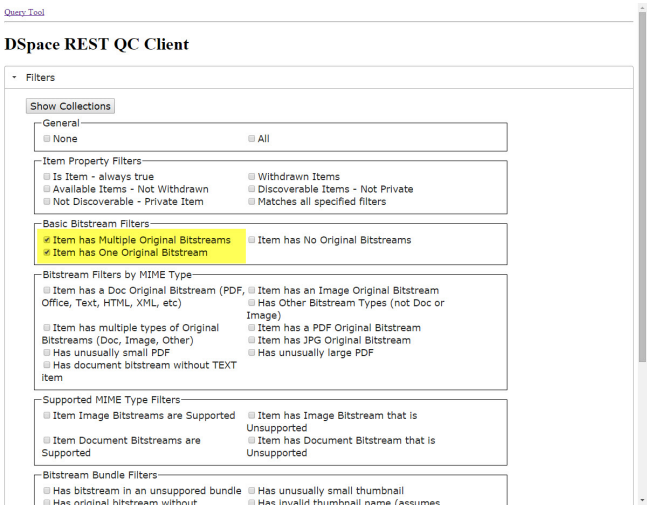
DSpace 7.0 only supports this when using the older, deprecated REST API v6

In DSpace 7.0, REST Quality Control Reports are currently only supported if you also install the old REST API v6 (deprecated)(see page 506) webapp. Tentative plans to migrate these reports to support the new REST API(see page 502) have begun in https://jira.lyrasis.org/browse/DS-4301.

Initial Report Display

API Call

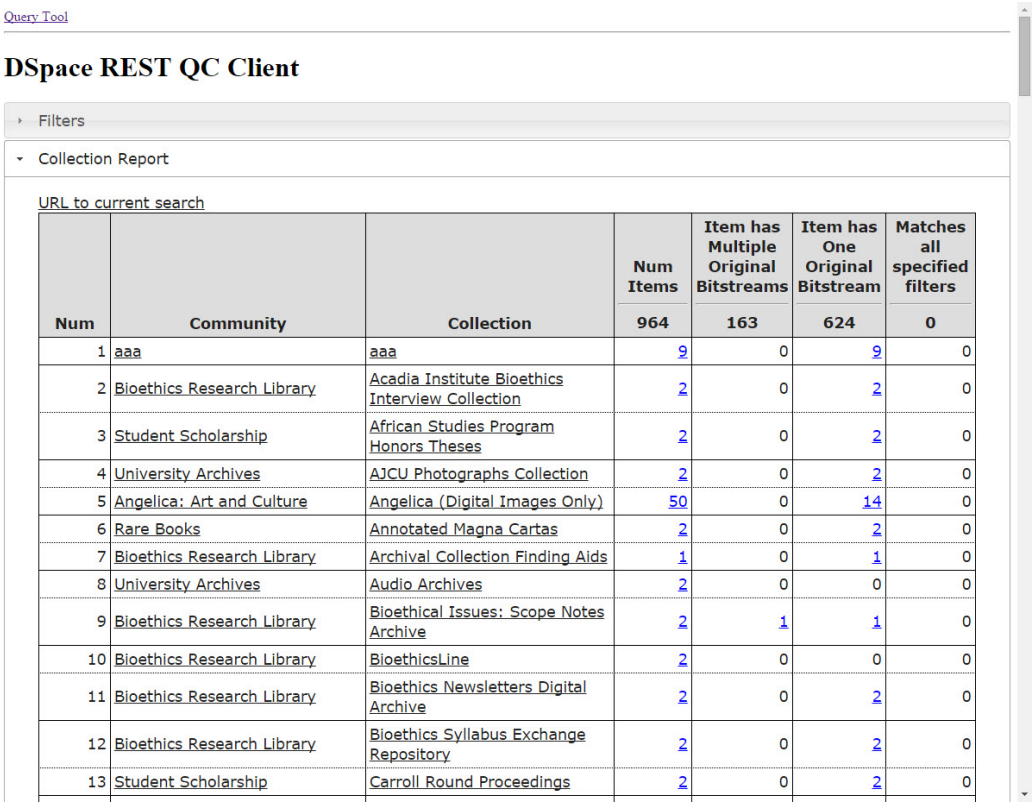
/rest/filtered-collections?limit=25&expand=topCommunity&offset=0



Select Filters of Interest

API Call

/rest/filters



Query Tool

DSpace REST QC Client

Filters

Collection Report

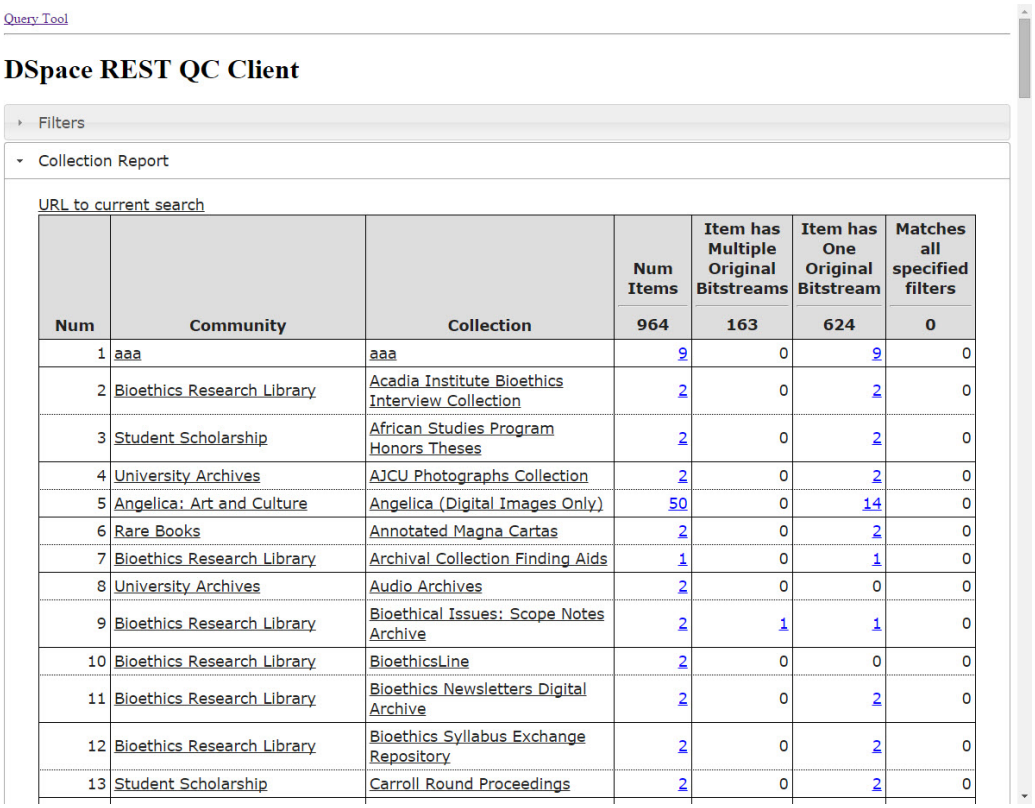
URL to current search

Num	Community	Collection	Num Items	Item has Multiple Original Bitstreams	Item has One Original Bitstream	Matches all specified filters
			964	163	624	0
1	aaa	aaa	9	0	9	0
2	Bioethics Research Library	Acadia Institute Bioethics Interview Collection	2	0	2	0
3	Student Scholarship	African Studies Program Honors Theses	2	0	2	0
4	University Archives	AJCU Photographs Collection	2	0	2	0
5	Angelica: Art and Culture	Angelica (Digital Images Only)	50	0	14	0
6	Rare Books	Annotated Magna Cartas	2	0	2	0
7	Bioethics Research Library	Archival Collection Finding Aids	1	0	1	0
8	University Archives	Audio Archives	2	0	0	0
9	Bioethics Research Library	Bioethical Issues: Scope Notes Archive	2	1	1	0
10	Bioethics Research Library	BioethicsLine	2	0	0	0
11	Bioethics Research Library	Bioethics Newsletters Digital Archive	2	0	2	0
12	Bioethics Research Library	Bioethics Syllabus Exchange Repository	2	0	2	0
13	Student Scholarship	Carroll Round Proceedings	2	0	2	0

View Filtered Counts

API Call

/rest/filtered-collections/{collection_id}?limit=500&filters=has_multiple_originals.has_one_original



Query Tool

DSpace REST QC Client

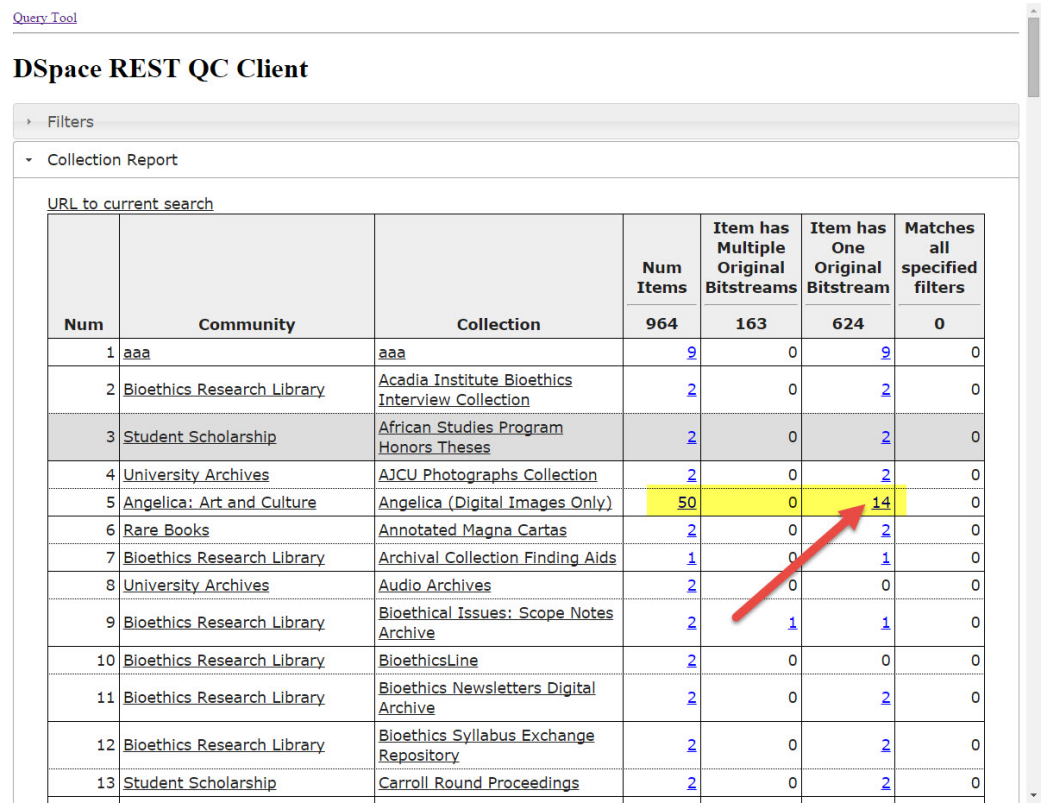
Filters

Collection Report

URL to current search

Num	Community	Collection	Num Items	Item has Multiple Original Bitstreams	Item has One Original Bitstream	Matches all specified filters
			964	163	624	0
1	aaa	aaa	9	0	9	0
2	Bioethics Research Library	Acadia Institute Bioethics Interview Collection	2	0	2	0
3	Student Scholarship	African Studies Program Honors Theses	2	0	2	0
4	University Archives	AJCU Photographs Collection	2	0	2	0
5	Angelica: Art and Culture	Angelica (Digital Images Only)	50	0	14	0
6	Rare Books	Annotated Magna Cartas	2	0	2	0
7	Bioethics Research Library	Archival Collection Finding Aids	1	0	1	0
8	University Archives	Audio Archives	2	0	0	0
9	Bioethics Research Library	Bioethical Issues: Scope Notes Archive	2	1	1	0
10	Bioethics Research Library	BioethicsLine	2	0	0	0
11	Bioethics Research Library	Bioethics Newsletters Digital Archive	2	0	2	0
12	Bioethics Research Library	Bioethics Syllabus Exchange Repository	2	0	2	0
13	Student Scholarship	Carroll Round Proceedings	2	0	2	0

View Items of Interest



Query Tool

DSpace REST QC Client

Filters

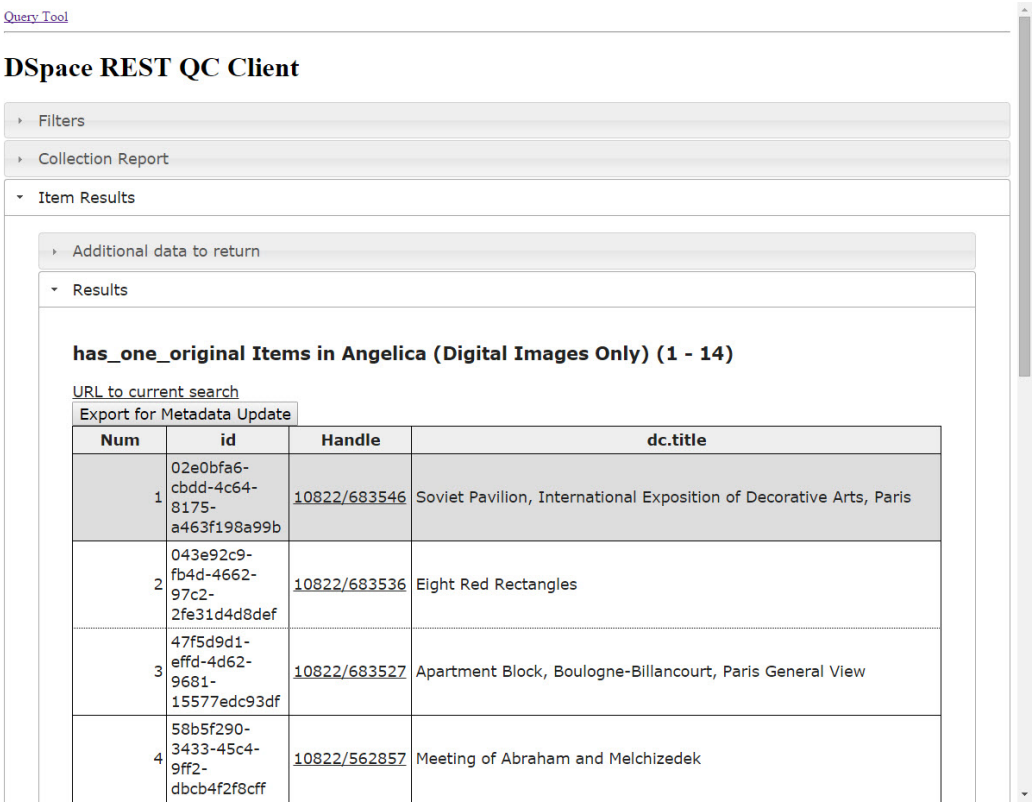
Collection Report

URL to current search

Num	Community	Collection	Num Items	Item has Multiple Original Bitstreams	Item has One Original Bitstream	Matches all specified filters
			964	163	624	0
1	aaa	aaa	9	0	9	0
2	Bioethics Research Library	Acadia Institute Bioethics Interview Collection	2	0	2	0
3	Student Scholarship	African Studies Program Honors Theses	2	0	2	0
4	University Archives	AJCU Photographs Collection	2	0	2	0
5	Angelica: Art and Culture	Angelica (Digital Images Only)	50	0	14	0
6	Rare Books	Annotated Magna Cartas	2	0	2	0
7	Bioethics Research Library	Archival Collection Finding Aids	1	0	1	0
8	University Archives	Audio Archives	2	0	0	0
9	Bioethics Research Library	Bioethical Issues: Scope Notes Archive	2	1	1	0
10	Bioethics Research Library	BioethicsLine	2	0	0	0
11	Bioethics Research Library	Bioethics Newsletters Digital Archive	2	0	2	0
12	Bioethics Research Library	Bioethics Syllabus Exchange Repository	2	0	2	0
13	Student Scholarship	Carroll Round Proceedings	2	0	2	0

API Call

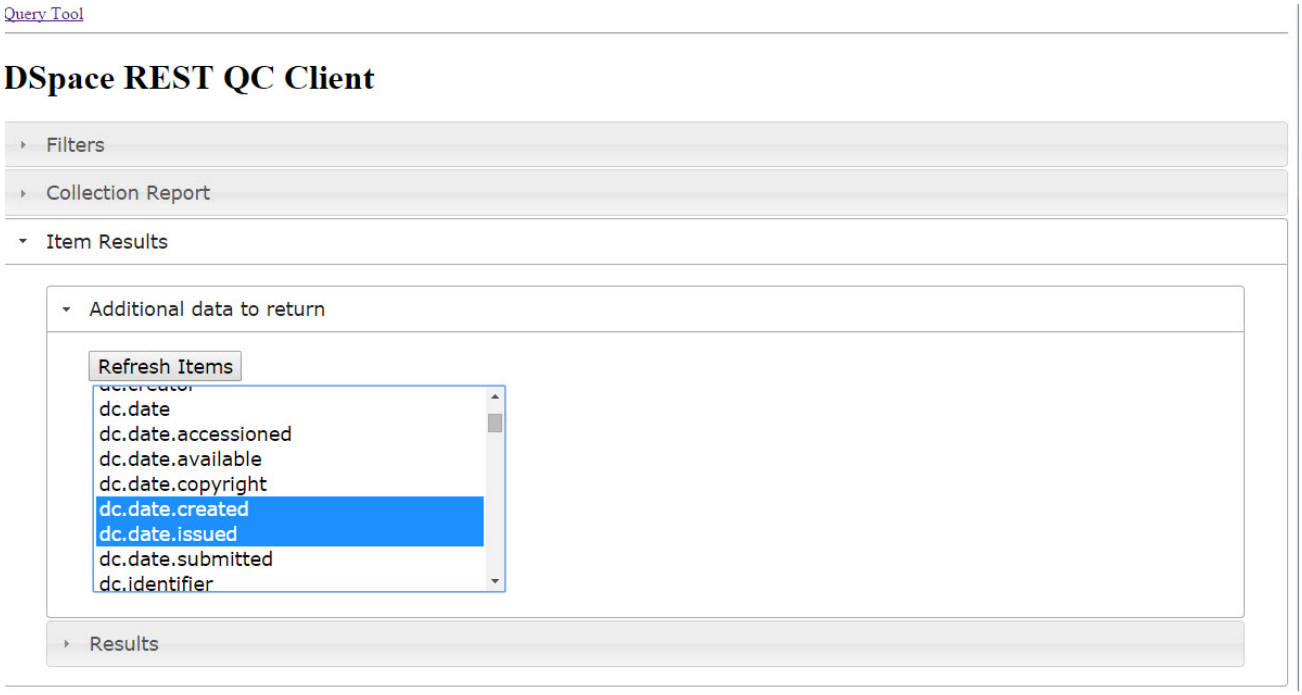
/rest/filtered-collections/{collection_id}?expand=items&limit=100&filters=has_one_original&offset=0



Select additional fields to display

API Call

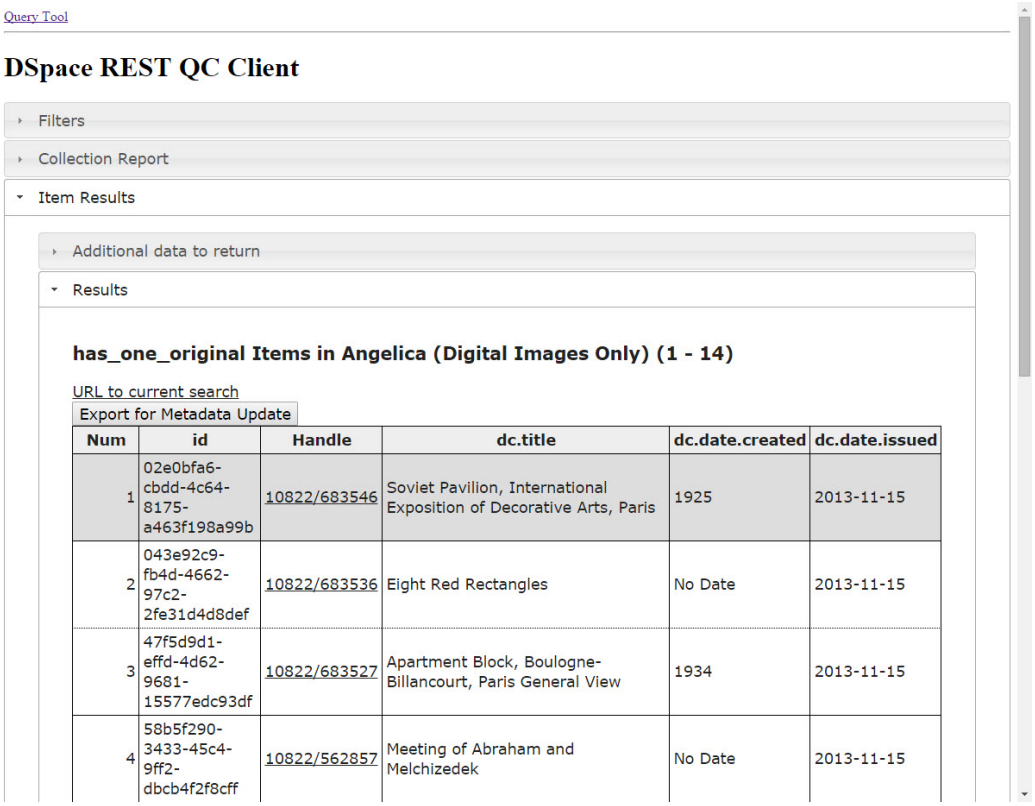
/rest/registries/schema



View Updated Results

API Call

/rest/filtered-collections/{collection_id}?expand=items,metadata&limit=100&filters=has_one_original&offset=0&show_fields[]=dc.date.created&show_fields[]=dc.date.issued



Query Tool

DSpace REST QC Client

- Filters
- Collection Report
- Item Results
  - Additional data to return
  - Results

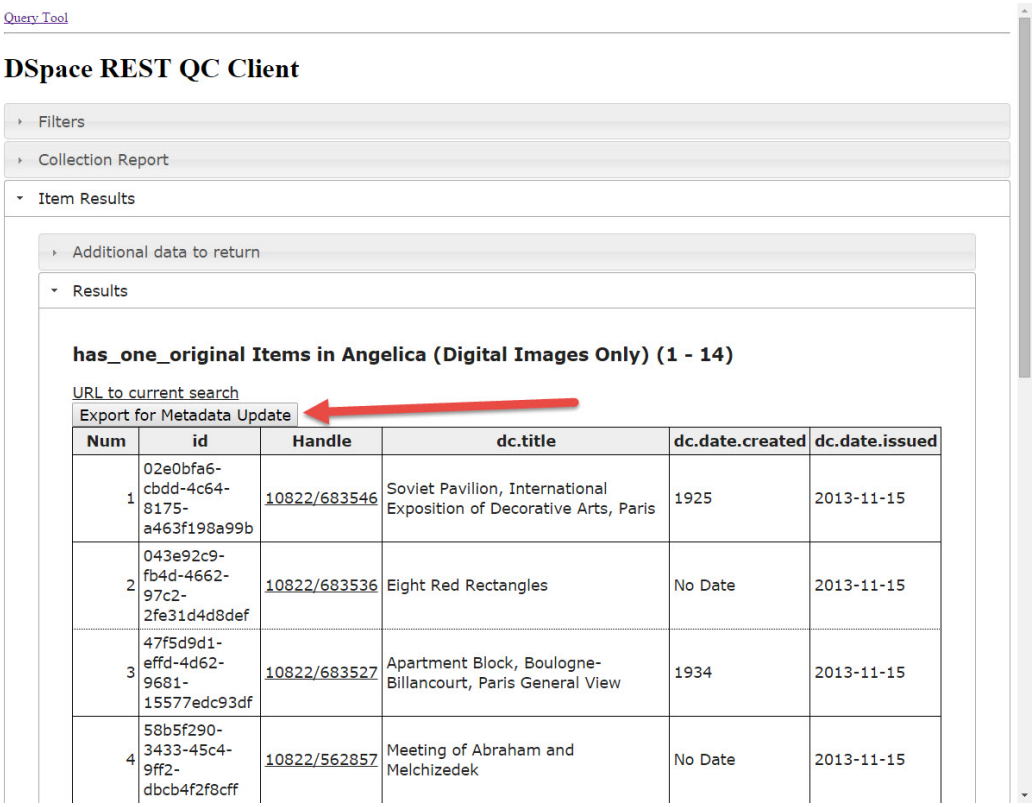
has_one_original Items in Angelica (Digital Images Only) (1 - 14)

URL to current search

Export for Metadata Update

Num	id	Handle	dc.title	dc.date.created	dc.date.issued
1	02e0bfa6-cbdd-4c64-8175-a463f198a99b	10822/683546	Soviet Pavilion, International Exposition of Decorative Arts, Paris	1925	2013-11-15
2	043e92c9-fb4d-4662-97c2-2fe31d4d8def	10822/683536	Eight Red Rectangles	No Date	2013-11-15
3	47f5d9d1-effd-4d62-9681-15577edc93df	10822/683527	Apartment Block, Boulogne-Billancourt, Paris General View	1934	2013-11-15
4	58b5f290-3433-45c4-9ff2-dbcb4f2f8cff	10822/562857	Meeting of Abraham and Melchizedek	No Date	2013-11-15

Download CSV File for Metadata Update



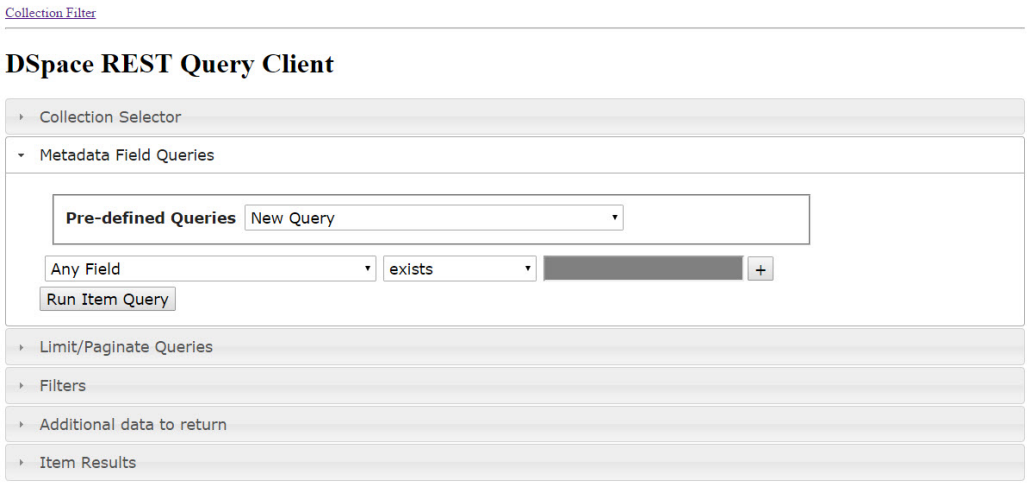
CSV File ready compatible with DSpace Metadata Update

```
"id","dc.title","dc.date.created","dc.date.issued"
"02e0bfa6-cbdd-4c64-8175-a463f198a99b","Soviet Pavilion, International Exposition of Decorative Arts, Paris","1925","2013-11-15"
"043e92c9-fb4d-4662-97c2-2fe31d4d8def","Eight Red Rectangles","No Date","2013-11-15"
"47f5d9d1-effd-4d62-9681-15577edc93df","Apartment Block, Boulogne-Billancourt, Paris General View","1934","2013-11-15"
"58b5f290-3433-45c4-9ff2-dbcb4f2f8cff","Meeting of Abraham and Melchizedek","No Date","2013-11-15"
"600a441e-fcd8-477b-b2a0-1b11eca97d35","Art Institute of Chicago Detail: Wing Detail: Wall MODEL","2005","2013-11-15"
"6401d1d0-3b67-46ab-892a-49549cd84fe3","Ascension","1801","2013-11-15"
"97c920e8-eebd-4548-95af-3c676d0f5c57","DuBarry Pavillion PLAN","1780","2013-11-15"
"a0e75f49-aa3a-4be7-9b13-48bad53a7a51","Saint Stefano Rotondo Detail: Entrance Portico","No Date","2013-11-15"
"b725a94f-5b86-472a-9a1c-58204f660703","Abbey du Thoronet PLAN","1146","2013-11-15"
"cd7dd096-5207-44e1-b480-f114a8ce8458","Soviet Pavilion, International Exposition of Decorative Arts, Paris","1925","2013-11-15"
"d44f90df-b20e-4ba5-a9c5-cd731868e79b","Mata Hari","1906","2013-11-15"
"dd74c102-4617-470f-ab42-c61caf230e88","Sistine Chapel Detail: Interior c1500","1473","2013-11-15"
"e5538842-a919-4ecc-ad16-2cf66993c031","Rheims Cathedral UNDER RESTORATION","1250","2013-11-15"
"eeeb25b9-463d-417b-9f2c-c791f78a69c9","Amiens Cathedral Notre-Dame PLAN","1220","2013-11-15"
```

6.3.6.7 REST Reports - Metadata Query Screenshots with Annotated API Calls

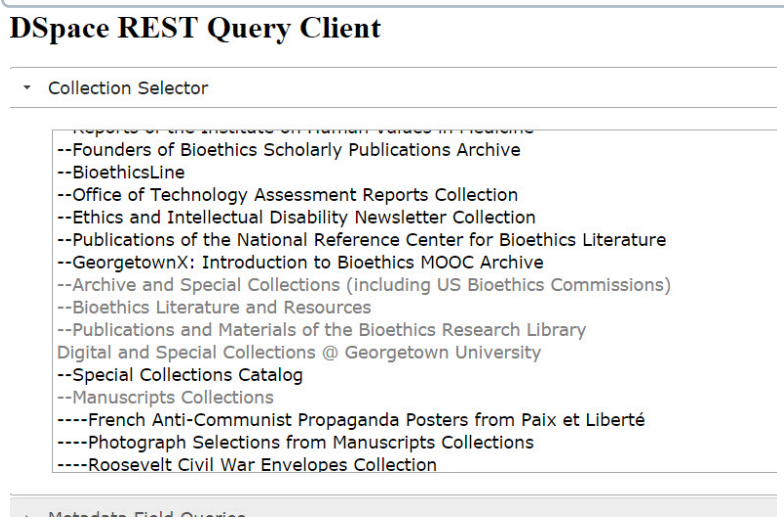
DSpace 7.0 only supports this when using the older, deprecated REST API v6

In DSpace 7.0, REST Quality Control Reports are currently only supported if you also install the old REST API v6 (deprecated)(see page 506) webapp. Tentative plans to migrate these reports to support the new REST API(see page 502) have begun in https://jira.lyrasis.org/browse/DS-4301.

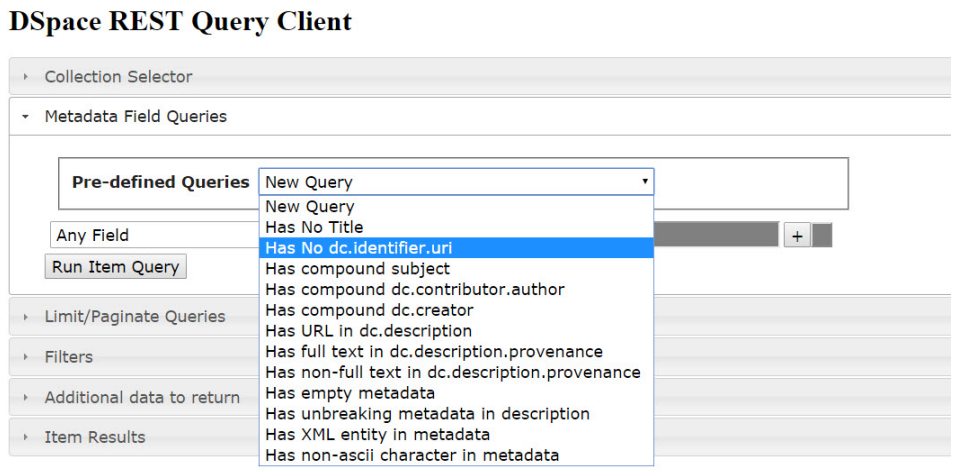


Set collections to Query

> **API Call**
>
> /rest/hierarchy

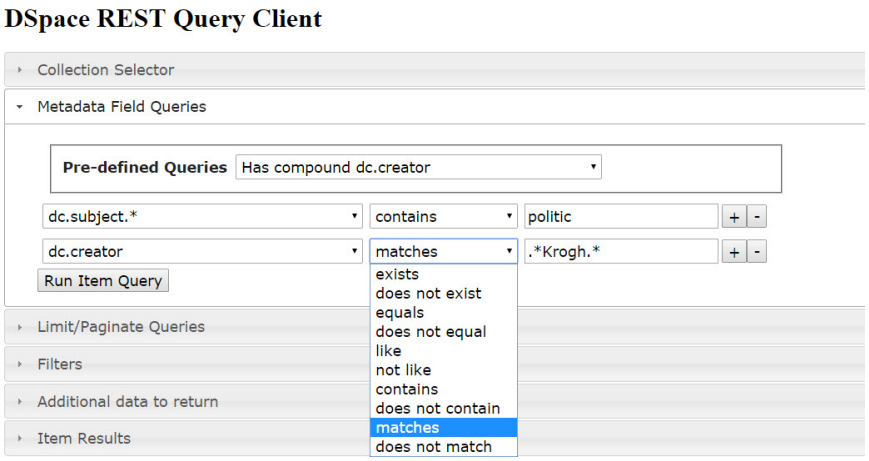


Pre-defined Queries are Available



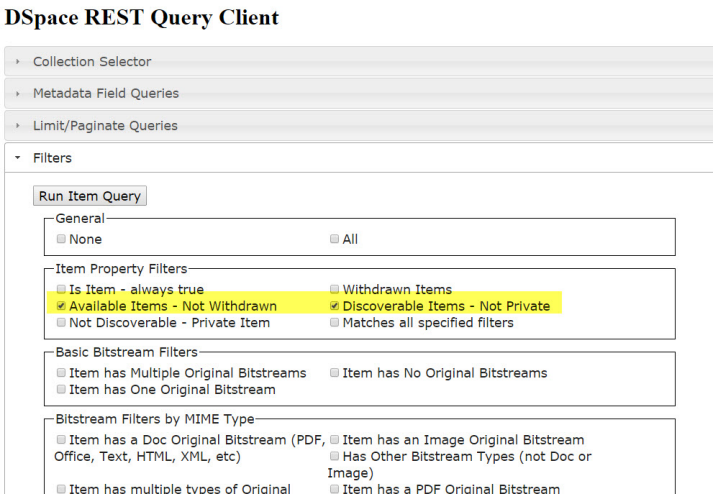
Multiple Metadata Fields can be Queried

> **API Call**
>
> /rest/registries/schema



Apply Filters if Desired

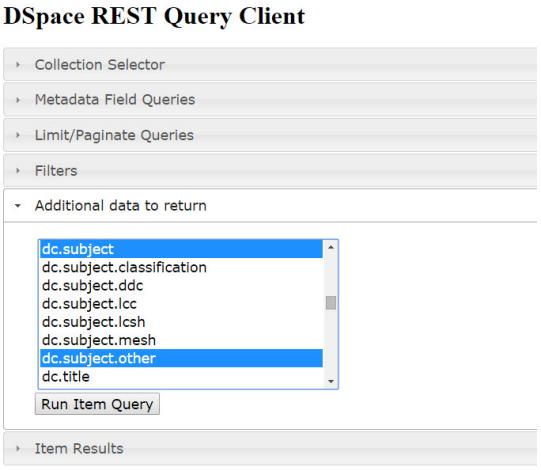
> **API Call**
>
> /rest/filters



Select Additional Fields to Display

API Call

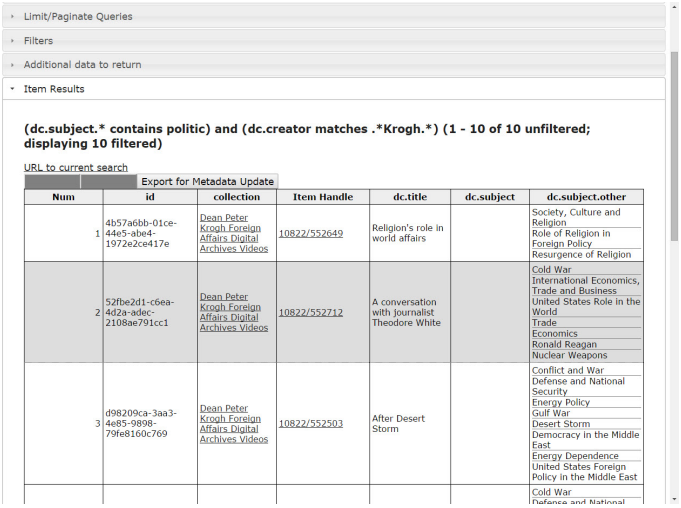
/rest/registries/schema



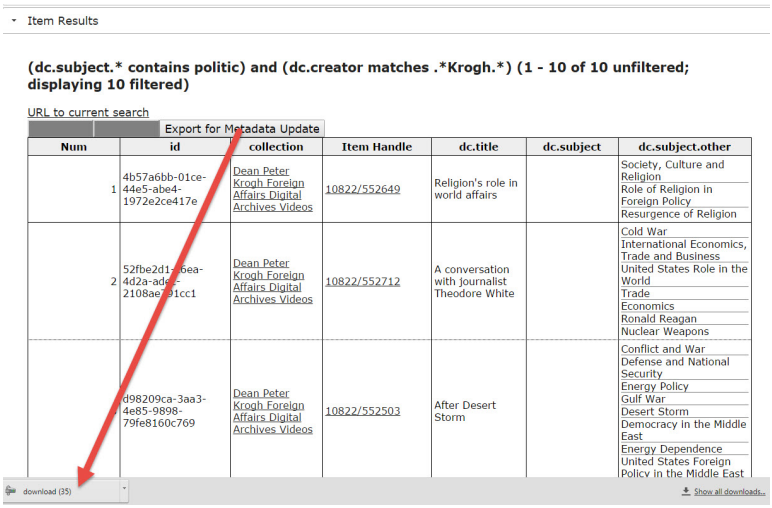
View Results

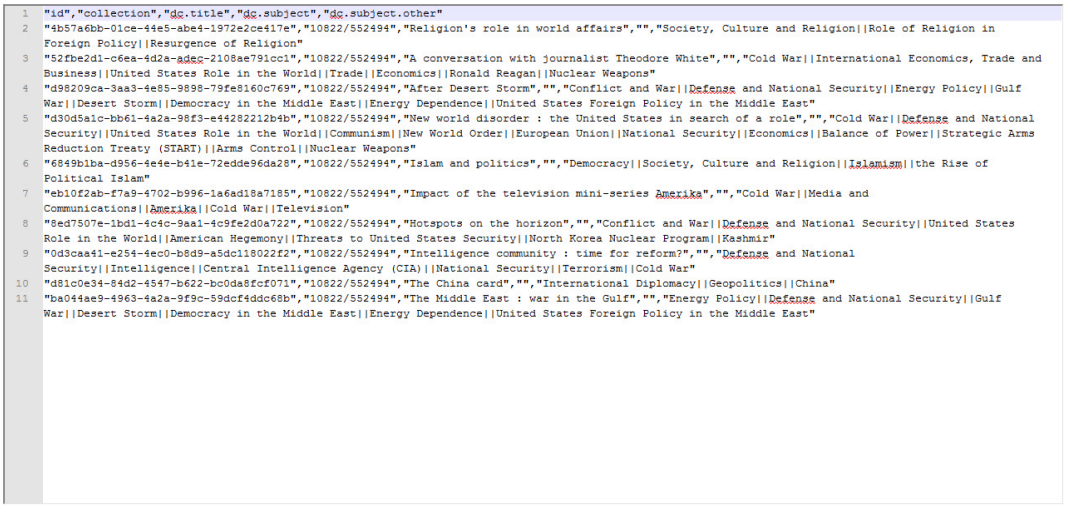
API Call

```
rest/filtered-items?

query_field[]=dc.subject.*&query_field[]=dc.creator&query_op[]=contains&query_op[]=matches&query_val[]=politic&query_val[]=.*Krogh.*

&collSel[]=

&limit=100&offset=0

&expand=parentCollection,metadata

&filters=is_withdrawn,is_discoverable

&show_fields[]=dc.subject&show_fields[]=dc.subject.other
```



Export as CSV for DSpace Metadata Update Process





```
"id","collection","dc.title","dc.subject","dc.subject.other"
"4b57a6bb-01ce-44e5-abe4-1972e2ce417e","10822/552494","Religion's role in world affairs","","Society, Culture and Religion||Role of Religion in Foreign Policy||Resurgence of Religion"
"52fbe2d1-c6ea-4d2a-adec-2108ae791cc1","10822/552494","A conversation with journalist Theodore White","","Cold War||International Economics, Trade and Business||United States Role in the World||Trade||Economics||Ronald Reagan||Nuclear Weapons"
"d98209ca-3aa3-4e85-9898-79fe8160c769","10822/552494","After Desert Storm","","Conflict and War||Defense and National Security||Energy Policy||Gulf War||Desert Storm||Democracy in the Middle East||Energy Dependence||United States Foreign Policy in the Middle East"
"d30d5a1c-bb61-4a2a-98f3-e44282212b4b","10822/552494","New world disorder : the United States in search of a role","","Cold War||Defense and National Security||United States Role in the World||Communism||New World Order||European Union||National Security||Economics||Balance of Power||Strategic Arms Reduction Treaty (START)||Arms Control||Nuclear Weapons"
"6849b1ba-d956-4e4e-b41e-72edde96da28","10822/552494","Islam and politics","","Democracy||Society, Culture and Religion||Islamism||the Rise of Political Islam"
"eb10f2ab-f7a9-4702-b996-1a6ad18a7185","10822/552494","Impact of the television mini-series Amerika","","Cold War||Media and Communications||Amerika||Cold War||Television"
"8ed7507e-1bd1-4c4c-9aa1-4c9fe2d0a722","10822/552494","Hotspots on the horizon","","Conflict and War||Defense and National Security||United States Role in the World||American Hegemony||Threats to United States Security||North Korea Nuclear Program||Kashmir"
"0d3caa41-e254-4ec0-b8d9-a5dc118022f2","10822/552494","Intelligence community : time for reform?","","Defense and National Security||Intelligence||Central Intelligence Agency (CIA)||National Security||Terrorism||Cold War"
"d81c0e34-84d2-4547-b622-bc0da8fcf071","10822/552494","The China card","","International Diplomacy||Geopolitics||China"
"ba044ae9-4963-4a2a-9f9c-59dcf4ddc68b","10822/552494","The Middle East : war in the Gulf","","Energy Policy||Defense and National Security||Gulf War||Desert Storm||Democracy in the Middle East||Energy Dependence||United States Foreign Policy in the Middle East"
```

6.3.6.8 REST Reports - Summary of API Calls

- GET /rest - Summary of API Calls(see page 537)
- GET /rest/reports - List of Available Reports(see page 538)
- GET /rest/reports/[report name] - Redirect to a Specific Report(see page 538)
- GET /rest/filters - Return filters to apply to a list of items(see page 538)
- GET /rest/filtered-collections - Return collections and item counts based on pre-defined filters(see page 538)
- GET /rest/filtered-collections/{collection_id} - Return items and item counts for a collection based on pre-defined filters(see page 539)
- GET /rest/filtered-items - Retrieve a set of items based on a metadata query and a set of filters(see page 539)

DSpace 7.0 only supports this when using the older, deprecated REST API v6

In DSpace 7.0, REST Quality Control Reports are currently only supported if you also install the old REST API v6 (deprecated)(see page 506) webapp. Tentative plans to migrate these reports to support the new REST API(see page 502) have begun in https://jira.lyrasis.org/browse/DS-4301.

GET /rest - Summary of API Calls

The response from this call includes the set of REST report calls that are available.

- GET /reports - Return a list of report tools built on the rest api
- GET /reports/{nickname} - Return a redirect to a specific report
- GET /filters - Return a list of use case filters available for quality control reporting
- GET /filtered-collections - Return collections and item counts based on pre-defined filters
- GET /filtered-collections/{collection_id} - Return items and item counts for a collection based on pre-defined filters
- GET /filtered-items - Retrieve a set of items based on a metadata query and a set of filters

GET /rest/reports - List of Available Reports

The response from this call includes the set of reports that are available

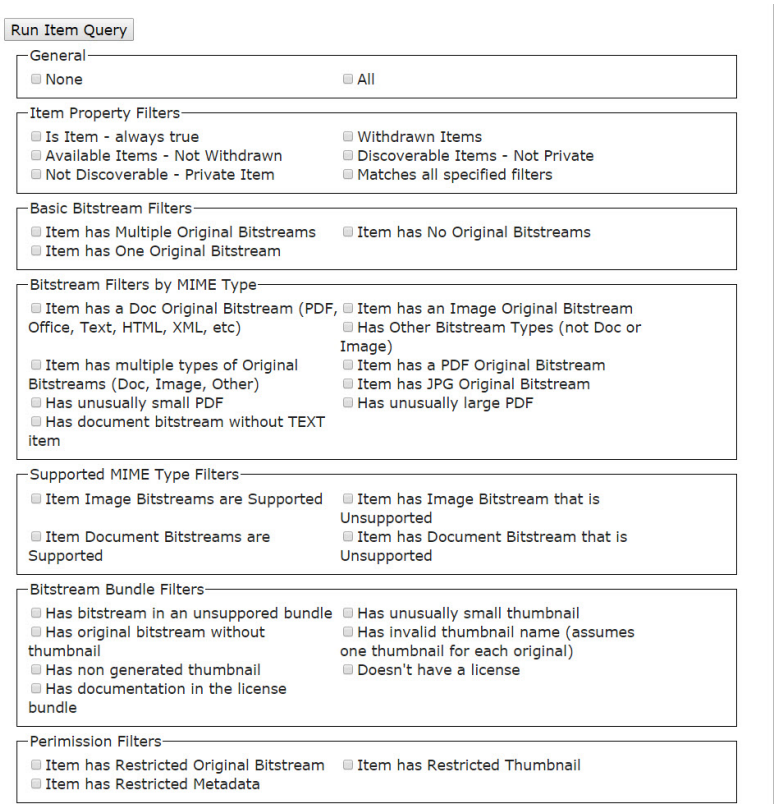
- **collection**: /rest/static/index.html
- **item-query**: /rest/static/query.html

GET /rest/reports/[report name] - Redirect to a Specific Report

This will re-direct to the configured report

GET /rest/filters - Return filters to apply to a list of items

The response will return the list of available filters. These have been configured in rest.cfg.



GET /rest/filtered-collections - Return collections and item counts based on pre-defined filters

This request is similar to the call /rest/collections except that it allows the user to supply a comma separated list of filters to apply to the collection.

The response will contain

- The total count of items for each collection
- The count of items that match each filter
- The count of items that match all filters combined

GET /rest/filtered-collections/{collection_id} - Return items and item counts for a collection based on pre-defined filters

This request is similar to the call /rest/collections/{collection_id} except that it allows the user to supply a comma separated list of filters to apply to the collection.

The response will contain

- The total count of items for the collection
- The count of items that match each filter
- The count of items that match all filters combined

When combined with the expand=items parameter, this call will return the set of items that match a filter or set of filters. It may be necessary to paginate through these results.

GET /rest/filtered-items - Retrieve a set of items based on a metadata query and a set of filters

This request allows a collection owner to construct a complex metadata query against specific metadata fields applying a number of comparison operators.

The search features of DSpace allow an end user to discover items via search. This command allows the collection owner to audit and enforce metadata consistency within a collection.

The query will consist of the following components

- Metadata Field Query
  - Field(s) to be searched
  - Search operator
  - Search value (if applicable)
- Collection scope (optional)
  - Comma separated list of collections to search
- Filters
  - A comma separated list of item filters that will be applied to all results.
  - It may be necessary to paginate through results when applying a highly selective filter

6.4 Curation Tasks

6.4.1 Writing your own tasks

A task is just a java class that can contain arbitrary code, but it must have 2 properties:

First, it must provide a no argument constructor, so it can be loaded by the PluginManager. Thus, all tasks are 'named' plugins, with the taskname being the plugin name.

Second, it must implement the interface `org.dspace.curate.CurationTask`

The `CurationTask` interface is almost a "tagging" interface, and only requires a few very high-level methods be implemented. The most significant is:

```
int perform(DSpaceObject dso);
```

The return value should be a code describing one of 4 conditions:

- 0 : SUCCESS the task completed successfully
- 1 : FAIL the task failed (it is up to the task to decide what 'counts' as failure - an example might be that the virus scan finds an infected file)
- 2 : SKIPPED the task could not be performed on the object, perhaps because it was not applicable
- -1 : ERROR the task could not be completed due to an error

If a task extends the `AbstractCurationTask` class, that is the only method it needs to define.

6.4.2 Task Output and Reporting

Few assumptions are made by CS about what the 'outcome' of a task may be (if any) - it. could e.g. produce a report to a temporary file, it could modify DSpace content silently, etc. But the CS runtime does provide a few pieces of information whenever a task is performed:

6.4.2.1 Status Code

This is returned to CS by any of a task's `perform` methods. The complete list of values, defined in `Curator`, is:

value	symbol	meaning
-3	CURATE_NOTASK	CS could not find the requested task
-2	CURATE_UNSET	task did not return a status code because it has not yet run
-1	CURATE_ERROR	task could not be performed
0	CURATE_SUCCESS	task performed successfully
1	CURATE_FAIL	task performed, but failed
2	CURATE_SKIP	task not performed due to object not being eligible

In the administrative UI, this code is translated into the word or phrase configured by the *ui.statusmessages* property (discussed in Curation System(see page 143)) for display.

6.4.2.2 Result String

The task may set a string indicating details of the outcome:

```
curator.setResult("Item " + item.getID() + " was painted " + color);
```

CS does not interpret or assign result strings; the task does it. A task may choose not to assign a result, but the "best practice" for tasks is to assign one whenever possible. Code which invokes `Curator.getResult()` may use the result string for display or any other purpose.

6.4.2.3 Reporting Stream

For very fine-grained information, a task may write to a *reporting* stream. Unlike the result string, there is no limit to the amount of data that may be pushed to this stream.

```
curator.report("Lorem ipsum dolor sit amet,\n");
curator.report("consectetur adipiscing elit,\n");
curator.report("sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.\n");
```

6.4.2.4 Accessing task output in calling code

The status code, reporting stream, and the result string are accessed (or set) by methods on the Curator object:

```
Curator curator = new Curator();
curator.setReporter(new OutputStreamWriter(System.out));
curator.addTask("vscan").curate(coll);
int status = curator.getStatus("vscan");
String result = curator.getResult("vscan");
```

6.4.3 Task Properties

Task code may configure itself using ConfigurationService in the normal manner, or by the use of "task properties". See Curation System - Task Properties(see page 143) for discussion of the issues for which task properties were invented. Any code which extends AbstractCurationTask has access to its configured task properties.

The entire "API" for task properties is:

```
public String taskProperty(String name);
public int taskIntProperty(String name, int defaultValue);
public long taskLongProperty(String name, long defaultValue);
public boolean taskBooleanProperty(String name, boolean default);
```

6.4.4 Task Annotations

CS looks for, and will use, certain java annotations in the task Class definition that can help it invoke tasks more intelligently. An example may explain best. Since tasks operate on DSOs that can either be simple (Items) or containers (Collections, and Communities), there is a fundamental problem or ambiguity in how a task is invoked: if the DSO is a collection, should the CS invoke the task on each member of the collection, or does the task "know" how to do that itself? The decision is made by looking for the @Distributive annotation: if present, CS assumes that the task will manage the details, otherwise CS will walk the collection, and invoke the task on each member. The java class would be defined:

```
@Distributive
public class MyTask implements CurationTask
```

A related issue concerns how non-distributive tasks report their status and results: the status will normally reflect only the last invocation of the task in the container, so important outcomes could be lost. If a task declares itself @Suspendable, however, the CS will cease processing when it encounters a FAIL status. When used in the UI, for example, this would mean that if our virus scan is running over a collection, it would stop and return status (and result) to the scene on the first infected item it encounters. You can even tune @Supendable tasks more precisely by annotating what invocations you want to suspend on. For example:

```
@Suspendable(invoked=Curator.Invoked.INTERACTIVE)
public class MyTask implements CurationTask
```

would mean that the task would suspend if invoked in the UI, but would run to completion if run on the command-line.

Only a few annotation types have been defined so far, but as the number of tasks grow, we can look for common behavior that can be signaled by annotation. For example, there is a @Mutative type: that tells CS that the task may alter (mutate) the object it is working on.

6.4.5 Scripted Tasks

DSpace 1.8 introduced limited (and somewhat experimental) support for deploying and running tasks written in languages other than Java. Since version 6, Java has provided a standard way (API) to invoke so-called scripting or dynamic language code that runs on the java virtual machine (JVM). Scripted tasks are those written in a language accessible from this API. See Curation System - Scripted Tasks(see page 143) for information on configuring and running scripted tasks.

6.4.5.1 Interface

Scripted tasks must implement a slightly different interface than the CurationTask[448] interface used for Java tasks. The appropriate interface for scripting tasks is ScriptedTask[449] and has the following methods:

448 https://github.com/DSpace/DSpace/blob/dspace-3_x/dspace-api/src/main/java/org/dspace/curate/CurationTask.java
449 https://github.com/DSpace/DSpace/blob/dspace-3_x/dspace-api/src/main/java/org/dspace/curate/ScriptedTask.java

```
public void init(Curator curator, String taskId) throws IOException;
public int performDso(DSpaceObject dso) throws IOException;
public int performId(Context ctx, String id) throws IOException;
```

The difference is that `ScriptedTask` has separate `perform` methods for DSO and identifier. The reason for that is that some scripting languages (e.g. Ruby) don't support method overloading.

performDso() vs. performId()

You may have noticed that the `ScriptedTask` interface has both `performDso()` and `performId()` methods, but only performDso is ever called when curator is launched from command line.

There are a class of use-cases in which we want to construct or create new DSOs (DSpaceObject) given an identifier in a task. In these cases, there may be no live DSO to pass to the task.
You actually **can** get curation system to call `performId()` if you queue a task then process the queue - when reading the queue all CLI has is the handle to pass to the task.

6.4.6 Curation tasks in Jython

As mentioned in the "Scripted Tasks" chapter of Curation Tasks(see page 539), you can write your curation tasks in several languages, including Jython (a flavour of Python running on JVM).

⚠ Instructions are outdated and unproven in DSpace 7.x

6.4.6.1 Setting up scripted tasks in Jython

1. Download the latest Jython installer jar (e.g. jython-installer-2.7.1.jar[450]) from http://www.jython.org/downloads.html
2. Get `jython.jar` and the `Lib` directory.
   a. either unzip the installer jar:
      ```
      unzip -d [dspace]/lib/ jython-installer-2.7.1.jar jython.jar Lib/
      unzip -d [dspace]/webapps/server/WEB-INF/lib/ jython-installer-2.7.1.jar
      jython.jar Lib/
      ```
   b. or use it to install Jython:
      ```
      java -jar jython-installer-2.7.1.jar --console
      ```
      Note: Installation location doesn't matter, this is not necessary for DSpace. You can safely delete it after you retrieve `jython.jar` and `Lib`.
3. Install Jython to DSpace classpaths (step 2a already did this for you):
   a. The goal is to put `jython.jar` and the jython `Lib/` directory into **every** DSpace classpath you intend to use, so it must be installed in **both** `[dspace]/lib` and the webapp that deploys to Tomcat (if you want to run from the UI) - `[dspace]/webapps/server/WEB-INF/lib/`. There are no special maven/pom extensions - just copy in the jar and `Lib/`.
   b. You **can** use symlinks if you wish as long as allowLinking (Tomcat <=7[451], Tomcat 8[452]) is set to true in that context's configuration. However, be warned that Tomcat documentation lists allowLinking="true" as a possible security concern[453].

450 http://search.maven.org/remotecontent?filepath=org/python/jython-installer/2.7.1/jython-installer-2.7.1.jar
451 http://tomcat.apache.org/tomcat-7.0-doc/config/context.html#Standard_Implementation
452 http://tomcat.apache.org/tomcat-8.0-doc/config/resources.html
453 https://tomcat.apache.org/tomcat-8.0-doc/security-howto.html#Context

   c. Note: Older versions of Jython mention the need for jython-engine.jar to implement JSR-223. Don't worry about that, new Jython versions, e.g. 2.7.1 don't require this.
4. Configure the curation framework to be aware of your new task(s):
   a. set up the location of scripted tasks in the curation system. This means simply adding a property to `[dspace]/config/modules/curate.cfg`:
   `script.dir=${dspace.dir}/ctscripts`
   b. in this directory, create a text file named "`task.catalog`". This is a Java properties file where lines beginning with '#' are comments. Add a line for each task you write. The syntax is following:

```
# logical task name = script engine name|file name|constructor invocation
mytask=python|mytask.py|MyTask()
```

   Notes:

   - **don't** put spaces around the pipe character or you'll get an error similar to this one: `ERROR org.dspace.curate.TaskResolver @ Script engine: 'python ' is not installed`
   - The "script engine name" is whatever name (or alias) jython registers in the JVM. You can use both "python" and "jython" as engine name (tested on jython 2.7.1).
   - The logical task name can't conflict with existing (java) task names, but otherwise any single-word token can be used.
   - The file name is just the script file name in the `script.dir` directory
   - "constructor invocation" is the language specific way to create an object that implements the task interface - it's ClassName() for Python

   c. If you want pretty names in the UI, configure other curate.cfg properties - see "`ui.tasknames`" (or groups etc)
5. Write your task.
   In the directory configured above, create your task (with the name configured in "`task.catalog`").
   The basic requirement of any scripted task is that it implements the ScriptedTask[454] Java interface.
   So for our example, the `mytask.py` file might look like this:

```
from org.dspace.curate import ScriptedTask

class MyTask(ScriptedTask):
        def init(self, curator, taskName):
                print "initializing with Jython"

        def performDso(self, dso):
                print "perform on dso"
                return 0

        def performId(self, context, id):
                print "perform on id %s" % (id)
                return 0
```

6. Invoke the task.
   You can do this the same way you would invoke any task (from command line, in the admin UI, etc). The advantage of scripting is that you do not need to restart your servlet container to test changes; each task's source code is reloaded when you launch the task, so you can just put the updated script in place.
   Example of invocation from command line:

454 https://github.com/DSpace/DSpace/blob/dspace-3_x/dspace-api/src/main/java/org/dspace/curate/ScriptedTask.java

```
[dspace]/bin/dspace curate -t mytask -i 123456789/123 -r -
```

Note: "`-r -`" means that the script's standard output will be directed to the console. You can read more details in the "On the command line" chapter of the Curation Tasks(see page 539) page.

6.4.6.2 See also

- Curation Tasks(see page 539) page in the official documentation
- Nailgun[455] - for speeding up repeated runs of a dspace command from the command line
  Note: since DSpace 4.0, there's a solution for running dspace CLI commands in batch: Executing streams of commands[456]
- Jython webapp for DSpace[457] - general purpose (not curation task) webapp written in Jython, optionally with access to DSpace API

6.5 Development Tools Provided by DSpace

6.5.1 Date parser tester

Some parts of DSpace use a custom date/time parser (`org.dspace.util.MultiFormatDateParser`) which is driven by a table of regular expressions, so it can match any of a variety of formats. The table is found in `config/spring/api/discovery-solr.xml`. To test new and altered rules, you can use the DSpace command line tool's `validate-date` command. You can simply pass it a date/time string on the command line (`dspace validate-date` 01-01-2015). You can pipe a stream of strings to be validated, one per line (`dspace validate-date < test.data`). Or you can have it prompt you for each string to be tested (`dspace validate-date`).

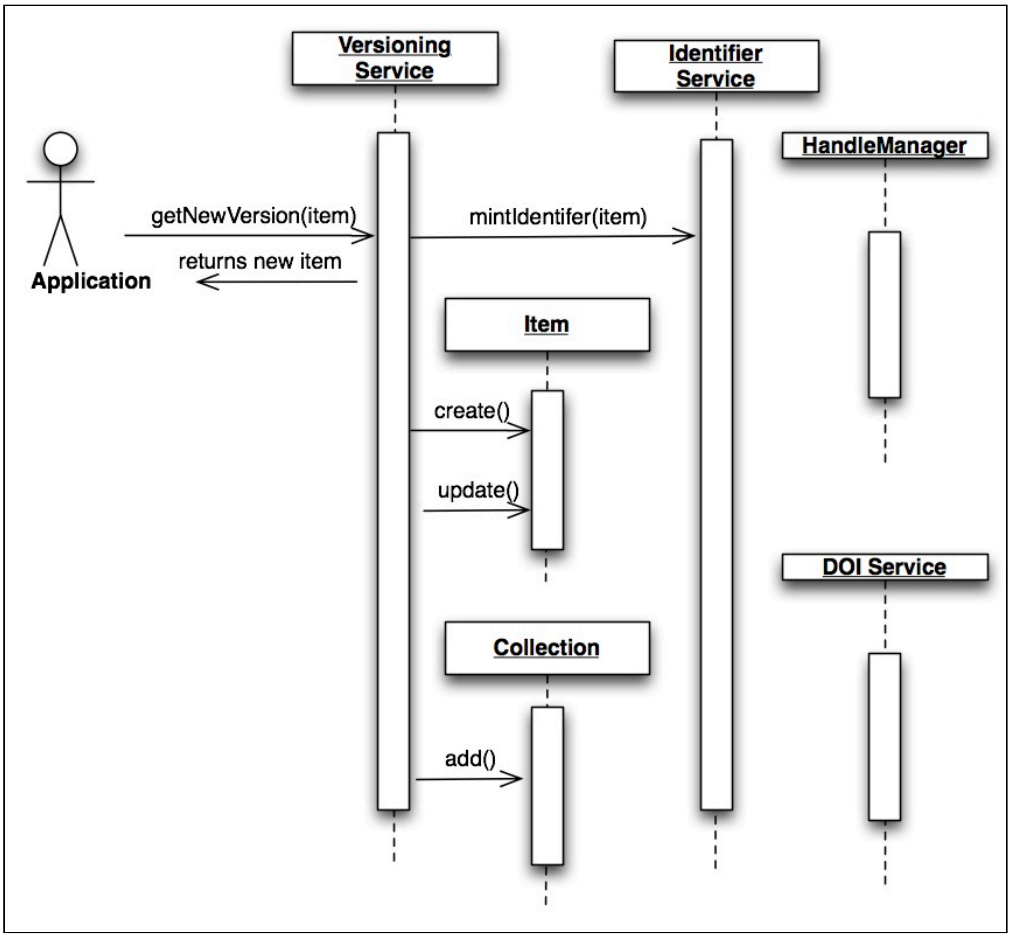
6.6 Services to support Alternative Identifiers

Together with the Item Level Versioning(see page 307) an Identifier Service was introduced that make it possible to integrate new Identifiers. Currently the Identifier Service is used for Items only, but this may be changed in future versions of DSpace. Identifiers used for different versions are an very important point as part of an versioning strategy. The following documentation describes the Identifier Service in the context of Item Level Versioning, nevertheless the Identifier Service is also used for Items when the Item Level Versioning is switched off.

6.6.1 Versioning and Identifier Service

DSpace Item Versioning is encapsulated as an Extensible Service that may be reimplemented by the local repository maintainers to produce alternate versioning behaviors and Identifier Schemes. Versioning Services layer on top of IdentifierServices dedicated to Encoding, Resolution, Minting and Registration of Identifiers for specific DSpace Items. It is through this highly extensible layering of functionality where local developers can alter the versioning behavior and introduce their own local enhancements. The DSpace Service Manager, based on the Spring Framework, provides the key leverage for this flexibility.

455 https://wiki.lyrasis.org/display/DSPACE/Nailgun
456 https://wiki.duraspace.org/display/DSDOC4x/Executing+streams+of+commands
457 https://wiki.duraspace.org/display/DSPACE/Jython+webapp+for+DSpace



6.6.1.1 Versioning Service

The *Versioning* Service will be responsible for the replication of one or more Items when a new version is requested. The new version will not yet be preserved in the Repository, it will be preserved when the databases transactional window is completed, thus when errors arise in the *versioning* process, the database will be properly kept in its original state and the application will alert that an exception has occurred that is in need of correction.

The *Versioning* Service will rely on a generic *IdentifierService* that is described below for minting and registering any identifiers that are required to track the revision history of the Items.

```
public interface VersioningService {

    Version createNewVersion(Context c, int itemId);

    Version createNewVersion(Context c, int itemId, String summary);

    void removeVersion(Context c, int versionID);

    void removeVersion(Context c, Item item);

    Version getVersion(Context c, int versionID);

    Version restoreVersion(Context c, int versionID);

    Version restoreVersion(Context c, int versionID, String summary);

    VersionHistory findVersionHistory(Context c, int itemId);

    Version updateVersion(Context c, int itemId, String summary);

    Version getVersion(Context c, Item item);
}
```

6.6.1.2 Identifier Service

The Identifier Service maintains an extensible set of *IdentifierProvider* services that are responsible for two important activities in Identifier management:

1. Resolution: *IdentifierService* act in a manner similar to the existing HandleManager in DSpace, allowing for resolution of DSpace Items from provided identifiers.
2. Minting: Minting is the act of reserving and returning an identifier that may be used with a specific DSpaceObject.
3. Registering: Registering is the act of recording the existence of a minted identifier with an external persistent resolver service. These services may reside on the local machine (HandleManager) or exist as external services (PURL or EZID DOI registration services)

```
public interface IdentifierService {

    /**
     *
     * @param context
     * @param dso
     * @param identifier
     * @return
     */
    String lookup(Context context, DSpaceObject dso, Class<? extends Identifier> identifier);

    /**
     *
     * This will resolve a DSpaceObject based on a provided Identifier.  The Service will
interrogate the providers in
     * no particular order and return the first successful result discovered.  If no resolution is
successful,
     * the method will return null if no object is found.
     *
     * TODO: Verify null is returned.
     *
     * @param context
     * @param identifier
     * @return
     * @throws IdentifierNotFoundException
     * @throws IdentifierNotResolvableException
     */
    DSpaceObject resolve(Context context, String identifier) throws IdentifierNotFoundException,
IdentifierNotResolvableException;

    /**
     *
     * Reserves any identifiers necessary based on the capabilities of all providers in the service.
     *
     * @param context
     * @param dso
     * @throws org.dspace.authorize.AuthorizeException
     * @throws java.sql.SQLException
     * @throws IdentifierException
     */
    void reserve(Context context, DSpaceObject dso) throws AuthorizeException, SQLException,
IdentifierException;

    /**
     *
     * Used to Reserve a Specific Identifier (for example a Handle,  hdl:1234.5/6) The provider is
responsible for
     * Detecting and Processing the appropriate identifier, all Providers are interrogated, multiple
providers
     * can process the same identifier.
     *
     * @param context
     * @param dso
     * @param identifier
     * @throws org.dspace.authorize.AuthorizeException
```

```
     * @throws java.sql.SQLException
     * @throws IdentifierException
     */
    void reserve(Context context, DSpaceObject dso, String identifier) throws AuthorizeException,
SQLException, IdentifierException;

    /**
     *
     * @param context
     * @param dso
     * @return
     * @throws org.dspace.authorize.AuthorizeException
     * @throws java.sql.SQLException
     * @throws IdentifierException
     */
    void register(Context context, DSpaceObject dso) throws AuthorizeException, SQLException,
IdentifierException;

    /**
     *
     * Used to Register a Specific Identifier (for example a Handle,  hdl:1234.5/6) The provider is
responsible for
     * Detecting and Processing the appropriate identifier, all Providers are interrogated, multiple
providers
     * can process the same identifier.
     *
     * @param context
     * @param dso
     * @param identifier
     * @return
     * @throws org.dspace.authorize.AuthorizeException
     * @throws java.sql.SQLException
     * @throws IdentifierException
     */
    void register(Context context, DSpaceObject dso, String identifier) throws AuthorizeException,
SQLException, IdentifierException;

    /**
     * Delete (Unbind) all identifiers registered for a specific DSpace item. Identifiers are
"unbound" across
     * all providers in no particular order.
     *
     * @param context
     * @param dso
     * @throws org.dspace.authorize.AuthorizeException
     * @throws java.sql.SQLException
     * @throws IdentifierException
     */
    void delete(Context context, DSpaceObject dso) throws AuthorizeException, SQLException,
IdentifierException;

    /**
     * Used to Delete a Specific Identifier (for example a Handle,  hdl:1234.5/6) The provider is
responsible for
     * Detecting and Processing the appropriate identifier, all Providers are interrogated, multiple
providers
```

```
     * can process the same identifier.
     *
     * @param context
     * @param dso
     * @param identifier
     * @throws org.dspace.authorize.AuthorizeException
     * @throws java.sql.SQLException
     * @throws IdentifierException
     */
    void delete(Context context, DSpaceObject dso, String identifier) throws AuthorizeException,
SQLException, IdentifierException;

}
```

6.7 Batch Processing

In the current DSpace design, the database transactions are in most of the cases relatively long: from Context creation to the moment the Context is completed. Especially when doing batch processing, that transaction can become very long. The new data access layer introduced in DSpace 6 which is based on Hibernate has built-in cache and auto-update mechanisms. But these mechanisms do not work well with long transactions and even have an exponentially adverse-effect on performance.

Therefore we added a new method `enableBatchMode()` to the DSpace Context class which tells our database connection that we are going to do some batch processing. The database connection (Hibernate in our case) can then optimize itself to deal with a large number of inserts, updates and deletes. Hibernate will then not postpone update statements anymore which is better in the case of batch processing. The method `isBatchModeEnabled()` lets you check if the current Context is in "batch mode".

When dealing with a lot of records, it is also important to deal with the size of the (Hibernate) cache. A large cache can also lead to decreased performance and eventually to "out of memory" exceptions. To help developers to better manage the cache, a method `getCacheSize()` was added to the DSpace Context class that will give you the number of database records currently cached by the database connection. Another new method `uncacheEntity( ReloadableEntity entity)` will allow you to clear the cache (of a single object) and free up (heap) memory. The `uncacheEntity()` method may be used to immediately remove an object from heap memory once the batch processing is finished with it. Besides the `uncacheEntity()` method, the `commit()` method in the DSpace Context class will also clear the cache, flush all pending changes to the database and commit the current database transaction. The database changes will then be visible to other threads.

BUT `uncacheEntity()` and `commit()` come at a price. After calling this method all previously fetched entities (hibernate terminology for database record) are "detached" (pending changes are not tracked anymore) and cannot be combined with "attached" entities. If you change a value in a detached entity, Hibernate will not automatically push that change to the database. If you still want to change a value of a detached entity or if you want to use that entity in combination with attached entities (e.g. adding a bitstream to an item) after you have cleared the cache, you first have to reload that entity. Reloading means asking the database connection to re-add the entity from the database to the cache and get a new object reference to the required entity. From then on, it is important that you use that new object reference. To simplify the process of reloading detached entities, we've added a `reloadEntity(ReloadableEntity entity)` method to the DSpace Context class with a new interface `ReloadableEntity`. This method will give the user a new "attached" reference to the requested entity. All DSpace Objects and some extra classes implement the `ReloadableEntity` interface so that they can be easily reloaded.

Examples on how to use these new methods can be found in the `IndexClient` class. But to summarize, when batch processing it is important that:

1. You put the Context into batch processing mode using the method:

```
boolean originalMode = context.isBatchModeEnabled();
context.enableBatchMode(true);
```

2. Perform necessary batch operations, being careful to call `uncacheEntity()` whenever you complete operations on each object. Alternatively, you can `commit()` the context once the object cache reaches a particular size (see `getCacheSize()`). Remember, once an object is "uncached", you will have to reload it (see `reloadEntity()`) before you can work with it again:

```
final Iterator<Item> itemIterator = itemService.findByCollection(context, collection);

// Loop through all items
while (itemIterator.hasNext()) {
  // Get access to next Item
  Item item = itemIterator.next();

  ... do something with Item ...

  // To prevent memory issues, discard Item from the cache after processing
  context.uncacheEntity(item);
}

// Remember: calling commit() will decache all objects
context.commit();

// So, if you need to reuse your Collection *post* commit(), you'd have to reload it
Collection collection = context.reloadEntity(collection);
```

3. When you're finished with processing the records, you put the context back into its original mode:

```
context.enableBatchMode(originalMode);
```

7 DSpace Reference

7.1 Configuration Reference

There are a numbers of ways in which DSpace may be configured and/or customized. This chapter of the documentation will discuss the configuration of the software and will also reference customizations that may be performed in the chapter following.

For ease of use, the Configuration documentation is broken into several parts:

- General Configuration(see page 554) - addresses general conventions used with configuring the `local.cfg` file, `dspace.cfg` and other configuration files which use similar conventions.
- The local.cfg Configuration Properties File(see page 560) - describes how to use the `local.cfg` file to store all your locally customized configurations
- The dspace.cfg Configuration Properties File(see page 563) - specifies the basic `dspace.cfg` file settings (these settings specify the default configuration for DSpace)
- Optional or Advanced Configuration Settings(see page 631) - contain other more advanced settings that are optional in the `dspace.cfg` configuration file.

The full table of contents follows:

7.1.1 General Configuration

In the following sections you will learn about the different configuration files that you will need to edit so that you may make your DSpace installation work.

DSpace provides a number of textual configuration files which may be used to configure your site based on local needs. These include:

- `[dspace]/config/dspace.cfg` : The primary configuration file, which contains the main configurations for DSpace.
- `[dspace]/config/modules/*.cfg` : Module configuration files, which are specific to various modules/features within DSpace.
- `[dspace]/config/local.cfg` : A (optional, but highly recommended) localized copy of configurations/settings specific to your DSpace (see The local.cfg Configuration Properties File(see page 560) below)
- Additional feature-specific configuration files also exist under `[dspace]/config/`, some of these include:

    - `default.license` : the default deposit license used by DSpace during the submission process (see Submission User Interface(see page 260) documentation)
    - `hibernate.cfg.xml` : The Hibernate class configuration for the DSpace database (almost never requires changing)
    - `item-submission.xml` : the default item submission process for DSpace (see Submission User Interface(see page 260) documentation)
    - `launcher.xml` : The configuration of the DSpace command-line "launcher" (`[dspace]/bin/dspace` , see the DSpace Command Launcher(see page 0) documentation)
    - `log4j2.xml` : The default logging settings for DSpace log files (usually placed in `[dspace]/log`)
    - `submission-forms.xml`: The default deposit forms for DSpace, used by `item-submission.xml` (see Submission User Interface(see page 260) documentation)

As most of these configurations are detailed in other areas of the DSpace documentation (see links above), this section concentrates primarily on the "*.cfg" configuration files (namely `dspace.cfg` and `local.cfg`).

7.1.1.1 Configuration File Syntax

We will use the `dspace.cfg` as our example for input conventions used throughout the system. These same input conventions apply to all DSpace *.cfg files.

All DSpace *.cfg files use the Apache Commons Configuration properties file syntax[458]. This syntax is very similar to a standard Java properties file, with a few notable enhancements described below.

- Comments all start with a "#" symbol. These lines are ignored by DSpace.
- Other settings appear as property/value pairs of the form: `property.name = property value`
- Certain special characters (namely commas) MUST BE escaped. See the "Special Characters" section below
- Values assigned in the same *.cfg file are "additive", and result in an array of values. See "Specifying Multiple Values for Properties" below.

Some property defaults are "commented out". That is, they have a "#" preceding them, and the DSpace software ignores the config property. This may cause the feature not to be enabled, or, cause a default property to be used.

The property value may contain references to other configuration properties, in the form `${property.name}`. A property may not refer to itself. Examples:

```
dspace.dir = /path/to/dspace
dspace.name = My DSpace

# property.name will be equal to "My DSpace is great!"
property.name = ${dspace.name} is great!

# property2.name will be equal to "/path/to/dspace/rest/of/path"
property2.name = ${dspace.dir}/rest/of/path

# However, this will result in an ERROR, as the property cannot reference itself
property3.name = ${property3.name}
```

Special Characters

Certain characters in `*.cfg` files are considered special characters, and **must** be escaped in any values. The most notable of these special characters include:

- Commas (,) : as they represent lists or arrays of values (see "Specifying Multiple Values for Properties" below)
- Backslashes (\) : as this is the escape character

This means that if a particular setting needs to use one of these special characters in its value, it must be escaped. Here's a few examples:

458 https://commons.apache.org/proper/commons-configuration/userguide/howto_properties.html

```
# WRONG SETTING
# This setting is INVALID. DSpace is expecting your site name to be a single value,
# But, this setting would create an array of two values: "DSpace" and "My Institution"
dspace.name = DSpace, My Institution

# CORRECT SETTING (commas is escaped)
# Instead, if the name of your DSpace includes a comma, you need to escape it with "\,"
dspace.name = DSpace\, My Institution

# WRONG SETTING
# As the backslash is the escape character, this won't work
property.name = \some\path

# CORRECT SETTING
# If you want a literal backslash, you need to escape it with "\\"
# So, the below value will be returned as "\some\path"
property.name = \\some\\path
```

Additional examples of escaping special characters are provided in the documentation of the Apache Commons Configuration properties file syntax[459].

Specifying Multiple Values for Properties

Because DSpace supports the Apache Commons Configuration properties file syntax[460], it is much easier to specify multiple values for a single setting. All you have to do is repeat the same property name multiple times in the same `*.cfg` file.

For example:

```
# The below settings define *two* AuthenticationMethods that will be enabled, LDAP and Password
authentication
# Notice how the same property name is simply repeated, and passed different values.
plugin.sequence.org.dspace.authenticate.AuthenticationMethod = org.dspace.authenticate.LDAPAuthentication
plugin.sequence.org.dspace.authenticate.AuthenticationMethod =
org.dspace.authenticate.PasswordAuthentication

# Alternatively, you can also define them as a comma-separated list
# (In this scenario, you would NOT escape the comma, as you want them to be considered multiple values)
# So, this single line is exactly equavalent to the settings above:
plugin.sequence.org.dspace.authenticate.AuthenticationMethod = org.dspace.authenticate.LDAPAuthentication,
org.dspace.authenticate.PasswordAuthentication
```

Please be aware that this ONLY works if you are reusing the exact same configuration in the same configuration file. This causes the values to be "additive" (i.e they are appended to the same list).

However, as you'll see below, the `local.cfg` file always *overrides* settings elsewhere. So, if the above "AuthenticationMethod" plugin was specified in both your `authentication.cfg` and your `local.cfg`, the value(s) in your `local.cfg` would *override* the defaults in your `authentication.cfg` (more on that below).

459 https://commons.apache.org/proper/commons-configuration/userguide/howto_properties.html
460 https://commons.apache.org/proper/commons-configuration/userguide/howto_properties.html

Additional examples of creating lists or arrays of values are provided in the documentation of the Apache Commons Configuration properties file syntax[461].

Including other Property Files

Because DSpace supports the Apache Commons Configuration properties file syntax[462], it also can include/embed property files within other property files by using the "`include=`" setting.

For example, the `dspace.cfg` includes/embeds all of the default `config/modules/*.cfg` files via a series of "`include=`" settings near the bottom of the `dspace.cfg`. As an example, here's a small subset of those include calls:

```
# defines our modules subdirectory
module_dir = modules

# The following lines include specific "authentication*.cfg" files inside your dspace.cfg
# This essentially "embeds" their configurations into your dspace.cfg,
# treating them as if they were a single configuration file.
include = ${module_dir}/authentication.cfg
include = ${module_dir}/authentication-ip.cfg
include = ${module_dir}/authentication-ldap.cfg
include = ${module_dir}/authentication-password.cfg
include = ${module_dir}/authentication-shibboleth.cfg
```

This ability to include properties files within others is very powerful, as it allows you to inherit settings from other files, or subdivide large configuration files. Be aware that this essentially causes DSpace to treat all included configurations *as if they were part of the parent file*. This means that, in the above example, as far as DSpace is concerned, all the settings contained within the `authentication*.cfg` files "appear" as though they are specified in the main `dspace.cfg`.

This ability to include other files is also possible with the `local.cfg` file, should you want to subdivide your localized settings into several locally specific configuration files.

7.1.1.2 Configuration Scheme for Reloading and Overriding

> **Known limitations to reloadable configurations**
>
> While the DSpace API supports dynamically reloading configurations, the user or machine interfaces *may* still cache some configuration settings. This means that while the API layer may reload a new value, that new value may not always affect/change the behavior of your user interface (until you restart Tomcat).
>
> Also, please be aware that all DSpace configuration values loaded into Spring beans (for example configurations that appear in Spring XML configuration files or in @Value annotations) **are cached by Spring**. This means that they will not be reloadable within Spring beans until Tomcat is restarted.

Because DSpace supports the Apache Commons Configuration[463], its configurations can now be reloaded without restarting your servlet container (e.g. Tomcat). By default, DSpace checks for changes to any of its runtime

461 https://commons.apache.org/proper/commons-configuration/userguide/howto_properties.html
462 https://commons.apache.org/proper/commons-configuration/userguide/howto_properties.html
463 http://commons.apache.org/proper/commons-configuration/

configuration files every 5 seconds. If a change has been made, the configuration file is reloaded. The 5 second interval is configurable in the `config-definition.xml` (which defines the configuration scheme DSpace uses).

Additionally, DSpace provides the ability to easily override default configuration settings (in dspace.cfg or modules/*.cfg) using a local.cfg file (see The local.cfg Configuration Properties File(see page 560)) or using System Properties / Environment Varilables.

Both of these features are defined in DSpace's default "configuration scheme" or "configuration definition" in the `[dspace]/config/config-definition.xml` file. This file defines the Apache Commons Configuration settings that DSpace utilizes by default. It is a valid "configuration definition" file as defined by Apache Commons Configuration. See their Configuration Definition File Documentation[464] for more details.

You are welcome to customize the `config-definition.xml` to customize your local configuration scheme as you see fit. Any customizations to this file will require restarting your servlet container (e.g. Tomcat).

By default, the DSpace `config-definition.xml` file defines the following configuration scheme:

- *Configuration File Syntax/Sources:* All DSpace configurations are loaded via Properties files (using the Configuration File Syntax(see page 554) detailed above)
  - Note: Apache Commons Configuration does support other configuration sources such as XML configurations or database configurations (see its Overview documentation[465]). At this time, DSpace does not utilize these other sorts of configurations by default. However, it would be possible to customize your local config-definition.xml to load settings from other locations.
- *Configuration Files/Sources:* By default, only two configuration files are loaded into Apache Commons Configuration for DSpace:
  - `local.cfg` (see The local.cfg Configuration Properties File(see page 560) documentation below)
  - `dspace.cfg` (NOTE: all `modules/*.cfg` are loaded by `dspace.cfg` via "`include=`" statements at the end of that configuration file. They are essentially treated as sub-configs which are embedded/included into the `dspace.cfg`)
- *Configuration Override Scheme:* The configuration override scheme is defined as follows. Configurations specified in earlier locations will automatically override any later values:
  - System Properties (`-D[setting]=[value]`) override all other options
  - Environment Variables
  - `local.cfg`
  - `dspace.cfg` (and all `modules/*.cfg` files) contain the default values for all settings.
- *Configuration Auto-Reload:* By default, all configuration files are automatically checked every 5 seconds for changes. If they have changed, they are automatically reloaded.

For more information on customizing our default config-definition.xml file, see the Apache Commons Configuration documentation on the configuration definition file[466]. Internally, DSpace simply uses the `DefaultConfigurationBuilder` class provided by Apache Commons Configuration to initialize our configuration scheme (and load all configuration files).

Customizing the default configuration scheme

Because the `config-definition.xml` file is just a Configuration Definition file for Apache Commons Configuration, you can also choose to customize the above configuration scheme based on your institution's local needs. This includes, but is not limited to, changing the name of "local.cfg", adding additional configuration files/sources, or modifying the override or auto-reload schemes. For more

464 https://commons.apache.org/proper/commons-configuration/userguide/howto_combinedbuilder.html#Configuration_definition_file_reference
465 https://commons.apache.org/proper/commons-configuration/userguide/overview.html
466 https://commons.apache.org/proper/commons-configuration/userguide/howto_combinedbuilder.html#Configuration_definition_file_reference

information, see the Configuration Definition File Documentation[467] from Apache Commons Configuration.

7.1.1.3 Why are there multiple copies of some config files?

It is important to remember that there are **multiple copies of each configuration files in an installation of DSpace**. The primary ones to be aware of are:

1. The "source" configuration file(s) are found under in `[dspace-source]/dspace/config/` or subdirectories. This also includes the `[dspace-source]/local.cfg`
2. The "runtime" configuration file(s) that are found in `[dspace]/config/`

The DSpace server (webapp) and command line programs only look at the *runtime* configuration file(s).

When you are revising/changing your configuration values, it may be tempting to *only edit the runtime file*. **DO NOT** do this. Whenever you rebuild DSpace, it will "reset" your runtime configuration to whatever is in your source directories (the previous runtime configuration is copied to a date suffixed file, should you ever need to restore it).

Instead, we recommend to **always make the same changes to the source version** of the configuration file in addition to the runtime file. In other words, the source and runtime files should always be identical / kept in sync.

One way to keep the two files in synchronization is to edit your files in `[dspace-source]/dspace/config/` and then run the following commands to rebuild DSpace and install the updated configs:

```
cd [dspace-source]/dspace/
mvn package
cd [dspace-source]/dspace/target/dspace-installer
ant update_configs
```

This will copy the source configuration files into the runtime ([dspace]/config) directory. Another option to manually sync the files by copying them to each directory.

Please note that there are additional "ant" commands to help with configuration management:

- "ant update_configs" ==> Moves existing configs in `[dspace]/config/` to *.old files and replaces them with what is in `[dspace-source]/dspace/config/`
- "ant -Doverwrite=false update_configs" ==> Leaves existing configs in `[dspace]/config/` intact. Just copies new configs from `[dspace-source]/dspace/config/` over to *.new files.

467 https://commons.apache.org/proper/commons-configuration/userguide/howto_combinedbuilder.html#Configuration_definition_file_reference

7.1.2 The `local.cfg` Configuration Properties File

build.properties has been replaced by local.cfg

As of DSpace 6 and above, the old "`build.properties`" configuration file has been replaced by this new "`local.cfg`" configuration file. For individuals who are familiar with the old `build.properties` file, this new `local.cfg` differs in a few key ways:

- Unlike `build.properties`, the `local.cfg` file can be used to override ANY setting in any other configuration file (`dspace.cfg` or `modules/*.cfg`). To override a default setting, simply copy the configuration into your `local.cfg` and change its value(s).
- Unlike `build.properties`, the `local.cfg` file is not utilized during the compilation process (e.g. mvn package). But, it is automatically copied alongside the final `dspace.cfg` into your installation location (`[dspace]/config/`), where it overrides default DSpace settings with your locally specific settings *at runtime*.
- Like `build.properties`, the `local.cfg` file is expected to be specified in the source directory by default (`[dspace-source]`). There is an example (`[dspace-source]/dspace/config/local.cfg.EXAMPLE`) provided which you can use to create a `[dspace-source]/dspace/config/local.cfg`.

Many configurations have changed names between DSpace 5 (and below) and DSpace 6 (and above)

If you are upgrading from an earlier version of DSpace, you will need to be aware that *many* configuration names/keys have changed. Because Apache Commons Configuration allows for auto-overriding of configurations, all configuration names/keys in different `*.cfg` files MUST be uniquely named (otherwise accidental, unintended overriding may occur).

In order to create this powerful ability to override configurations in your `local.cfg`, all `modules/*.cfg` files had their configurations **renamed** to be prepended with the module name. As a basic example, all the configuration settings within the `modules/oai.cfg` configuration now start with "`oai.`".

Additionally, while the `local.cfg` may look *similar* to the old `build.properties`, many of its configurations have slightly different names. *So, simply copying your build.properties into a local.cfg will NOT work*.

This means that DSpace 5.x (or below) configurations are NOT guaranteed compatible with DSpace 6. While you obviously can use your old configurations as a reference, you will need to start with fresh copy of all configuration files, and reapply any necessary configuration changes (this has always been the recommended procedure). However, as you'll see below, you'll likely want to do that anyways in order to take full advantage of the new `local.cfg` file.

It is possible to easily override default DSpace configurations (from `dspace.cfg` or `modules/*.cfg` files) in your own `local.cfg` configuration file.

A example `[dspace-source]/dspace/config/local.cfg.EXAMPLE` is provided with DSpace. The example only provides a few key configurations which most DSpace sites are likely to need to customize. However, you may add (or remove) any other configuration to your `local.cfg` to customize it as you see fit.

To get started, simply create your own `[dspace-source]/dspace/config/local.cfg` based on the example, e.g.

```
cd [dspace-source]/dspace/config/
cp local.cfg.EXAMPLE local.cfg
```

You can then begin to edit your `local.cfg` with your local settings for DSpace. There are a few key things to note about the `local.cfg` file:

- **Override any default configurations:** Any setting in your `local.cfg` will automatically OVERRIDE a setting of the same name in the `dspace.cfg` or any `modules/*.cfg` file. This also means that you can copy ANY configuration (from `dspace.cfg` or any `modules/*.cfg` file) into your `local.cfg` to specify a new value.
  - For example, specifying `dspace.url` in `local.cfg` will override the default value of `dspace.url` in `dspace.cfg`.
  - Also, specifying `oai.solr.url` in `local.cfg` will override the default value of `oai.solr.url` in `config/modules/oai.cfg`
- **Configuration Syntax:** The `local.cfg` file uses the Apache Commons Configuration Property file syntax (like all *.cfg files) . For more information see the section on Configuration File Syntax(see page 554) above.
  - This means the `local.cfg` also supports enhanced features like the ability to include other config files (via "`include=`" statements).
- **Override local.cfg via System Properties:** As needed, you also are able to OVERRIDE settings in your `local.cfg` by specifying them as System Properties or Environment Variables.
  - For example, if you wanted to change your `dspace.dir` in development/staging environment, you could specify it as a System Property (e.g. `-Ddspace.dir=[new-location]`). This new value will override any value in *both* `local.cfg` and `dspace.cfg`.

When you build DSpace (e.g. mvn package), this `local.cfg` file will be automatically copied to `[dspace]/config/local.cfg`. Similar to the `dspace.cfg`, the "runtime" configuration (used by DSpace) is the one in `[dspace]/config/local.cfg`. See the Why are there multiple copies of some config files?(see page 559) question above for more details on the runtime vs source configuration.

Here's a very basic example of settings you could place into your `local.cfg` file (with inline comments):

```
# This is a simple example local.cfg file which shows off options
# for creating your own local.cfg

# This overrides the default value of "dspace.dir" in dspace.cfg
dspace.dir = C:/dspace/

# This overrides the default value of "dspace.server.url" in dspace.cfg
dspace.server.url = https://dspace.myuniversity.edu/server

# The overrides the default "dspace.ui.url" setting it to the same value as my "baseUrl" above
dspace.ui.url = https://dspace.myuniversity.edu

# If our database settings are the same as the default ones in dspace.cfg,
# then, we may be able to simply customize the db.username and db.password
db.username = myuser
db.password = mypassword

# For DSpace, we want the LDAP and Password authentication plugins enabled
# This overrides the default AuthenticationMethod in /config/modules/authentication.cfg
# Since we specified the same key twice, these two values are appended (see Configuration File Syntax
above)
plugin.sequence.org.dspace.authenticate.AuthenticationMethod = org.dspace.authenticate.LDAPAuthentication
plugin.sequence.org.dspace.authenticate.AuthenticationMethod =
org.dspace.authenticate.PasswordAuthentication

# We also can reference other configurations in values.
# For instance, we can set the "mail.admin" and the "feedback.recipient" to be the same email
mail.admin = myemail@myuniversity.edu
feedback.recipient = ${mail.admin}

# For the example, we'll override the default oai.path in /config/modules/oai.cfg
# This puts our OAI-PMH interface at ${dspace.server.url}/oaipmh
oai.path = oaipmh

# We'll also override the default oai.solr.url in /config/modules/oai.cfg
# Notice here we're referencing a configuration (solr.server) that only exists in dspace.cfg
# This is allowed. Your local.cfg can reference configs from other *.cfg files.
oai.solr.url = ${solr.server}/oaipmh

# Finally, this local.cfg also supports adding "include=" statements, to include
# additional local configuration files.
# In this example, a local-rest.cfg and local-curate.cfg (in the same directory)
# will automatically be included as part of this local.cfg.
# This allows you to subdivide you local configs in as many (or few) config files
# as you desire.
include = local-rest.cfg
include = local-curate.cfg
```

7.1.3 The `dspace.cfg` Configuration Properties File

> **Any dspace.cfg setting can be overridden in your local.cfg**
>
> Remember, *any* of the below `dspace.cfg` settings can be copied into your local.cfg configuration file and overridden(see page 560). So, rather than editing the `dspace.cfg` (or any of the `modules/*.cfg`), it's recommended to simply override the default values in your `local.cfg`. That way, your `local.cfg` can serve as the record of which configurations you have actually tweaked in your DSpace, which may help to simplify future upgrades.

The `dspace.cfg` contains basic information about a DSpace installation, including system path information, network host information, and other like items. It is the default configuration file for DSpace, used by DSpace when it is actively running. However, as noted above, any of these default configurations may be overridden in your own `local.cfg` configuration file.

7.1.3.1 Main DSpace Configurations

Property:	`dspace.dir`
Example Value:	`/dspace`
Informational Note:	Root directory of DSpace installation. Omit the trailing slash '/'. Note that this setting is used by default in other settings, e.g. `assetstore.dir`. (*On Windows be sure to use forward slashes for the directory path!* For example: "C:/dspace" is a valid path for Window.)
Property:	`dspace.server.url`
Example Value:	http://dspacetest.myu.edu:8080
Informational Note:	Main URL at which DSpace backend ("server" webapp) is publicly available. If using port 80 (HTTP) or 443 (HTTPS), you may omit the port number. Otherwise the port number must be included. **Do not include a trailing slash ('/').** In Production, you must use HTTPS if you wish to access the REST API from a different server/domain.
Property:	`dspace.ui.url`
Example Value:	`dspace.ui.url = http://dspacetest.myu.edu:4000`

Informational note	Main URL at which the DSpace frontend (Angular User Interface) is publicly available. If using port 80 (HTTP) or 443 (HTTPS), you may omit the port number. Otherwise the port number must be included. **Do not include a trailing slash ('/').** In Production, you should be using HTTPS for security purposes. This URL should match the URL you type in the browser to access your User Interface. In the backend, this URL is primarily used to build UI-based URLs in sitemaps, email messages, etc. Therefore, it need not be set on initial installation, but it should be configured as soon as your user interface is installed. If you are not using the DSpace UI (and running the backend "headless"), this may be set to the URL of whatever you consider your primary "user interface".
Property:	`dspace.name`
Example Value:	`dspace.name = DSpace at My University`
Informational Note:	Short and sweet site name, used in e-mails, exports and machine interfaces (e.g. OAI-PMH). It is not currently used by the Angular UI.

7.1.3.2 DSpace Database Configuration

Many of the database configurations are software-dependent. That is, it will be based on the choice of database software being used. Currently, DSpace properly supports PostgreSQL and Oracle.

Property:	`db.url`
Example Value:	`db.url = jdbc:postgresql://localhost:5432/dspace`
Informational Note:	The above value is the default value when configuring with PostgreSQL. When using Oracle, use this value: `jbdc.oracle.thin:@//host:port/dspace`
Property:	`db.username`
Example Value:	`db.username = dspace`
Informational Note:	In the installation directions, the administrator is instructed to create the user "dspace" who will own the database "dspace".

Property:	`db.password`
Example Value:	`db.password = dspacepassword`
Informational Note:	This is the password that was prompted during the installation process (cf. 3.2.3. Installation)
Property:	`db.schema`
Example Value:	`db.schema = public`
Informational Note:	If your database contains multiple schemas, you can avoid problems with retrieving the definitions of duplicate objects by specifying the schema name here that is used for DSpace by uncommenting the entry. This property is optional. For PostgreSQL databases, this is often best set to "public" (default schema). For Oracle databases, the schema is usually equivalent to the username of your database account. So, for Oracle, this may be set to `${db.username}` in most scenarios.
Property:	`db.maxconnections`
Example Value:	`db.maxconnections = 30`
Informational Note:	Maximum number of Database connections in the connection pool
Property:	`db.maxwait`
Example Value:	`db.maxwait = 5000`
Informational Note:	Maximum time to wait before giving up if all connections in pool are busy (in milliseconds).
Property:	`db.maxidle`
Example Value:	`db.maxidle = -1`

Informational Note:	Maximum number of idle connections in pool. (*-1* = unlimited)
Property:	`db.cleanDisabled`
Example Value:	`db.cleanDisabled = true`
Informational Note:	This is a developer-based setting which determines whether you are allowed to run "./dspace database clean" to completely delete all content and tables in your database. This should always be set to "true" in Production to protect against accidentally deleting all your content by running that command. (Default is set to true)

To provide the database connection pool externally

Alternately, you may supply a configured connection pool out of JNDI. The object must be named `jdbc/dspace` (the full path is `java:comp/env/jdbc/dspace`). DSpace will always look up this name and, if found, will use the returned object as its database connection pool. If not found, the above `db.*` properties will be used to create the pool.

If you are using Tomcat, then the object might be defined using a `<Resource>` element, or connected to a `<Resource>` child of `<GlobalNamingResources>` using a `<ResourceLink>` element. See your Servlet container's documentation for details of configuring the JNDI initial context. For example, Tomcat provides a useful JNDI Datasource How-to[468]

Earlier releases of DSpace provided a configuration property `db.jndi` to specify the name to be looked up, but that has been removed. The name is specified in `config/spring/api/core-hibernate.xml` if you really need to change it.

7.1.3.3 DSpace Email Settings

The configuration of email is simple and provides a mechanism to alert the person(s) responsible for different features of the DSpace software.

DSpace will look up a javax.mail.Session object in JNDI and, if found, will use that to send email. Otherwise it will create a Session using some of the properties detailed below.

Property:	`mail.server`
Example Value:	`mail.server = smtp.my.edu`
Informational Note:	The address on which your outgoing SMTP email server can be reached.
Property:	`mail.server.username`

468 https://tomcat.apache.org/tomcat-9.0-doc/jndi-datasource-examples-howto.html

Example Value:	`mail.server.username = myusername`
Informational Note:	SMTP mail server authentication username, if required. This property is optional.
Property:	`mail.server.password`
Example Value:	`mail.server.password = mypassword`
Informational Note:	SMTP mail server authentication password, if required. This property is optional.
Property:	`mail.server.port`
Example Value:	`mail.server.port = 25`
Informational Note:	The port on which your SMTP mail server can be reached. By default, port 25 is used. Change this setting if your SMTP mailserver is running on another port. This property is optional.
Property:	`mail.from.address`
Example Value:	`mail.from.address = dspace-noreply@myu.edu`
Informational Note:	The "From" address for email. Change the 'myu.edu' to the site's host name.
Property:	`feedback.recipient`
Example Value:	`feedback.recipient = dspace-help@myu.edu`
Informational Note:	When a user clicks on the feedback link/feature, the information will be sent to the email address of choice. This configuration is currently limited to only one recipient. This is also the email address displayed on the contacts page.

Property:	`mail.admin`
Example Value:	`mail.admin = dspace-help@myu.edu`
Example Value:	Email address of the general site administrator (Webmaster). System notifications/reports and other sysadmin emails are sent to this email address.
Property:	`mail.admin.name`
Example Value:	`mail.admin.name = DSpace Administrator`
Example Value:	Name associated with the `mail.admin` email address.
Property:	`alert.recipient`
Example Value:	`alert.recipient = john.doe@myu.edu`
Informational Note:	Enter the recipient for server errors and alerts. This property is optional and defaults to the `${mail.admin}` setting
Property:	`registration.notify`
Example Value:	`registration.notify = mike.smith@myu.edu`
Informational Note:	Enter the recipient that will be notified when a new user registers on DSpace. This property is optional & defaults to no value.
Property:	`mail.charset`
Example Value:	`mail.charset = UTF-8`
Informational Note:	Set the default mail character set. This may be over-ridden by providing a line inside the email template '#set($charset = "encoding")'. Otherwise this default is used.
Property:	`mail.allowed.referrers`

Example Value:	`mail.allowed.referrers = localhost`
Informational Note:	A comma separated list of hostnames that are allowed to refer browsers to email forms. This property is optional. **UNSUPPORTED in DSpace 7.0**
Property:	`mail.extraproperties`
Example Value:	`mail.extraproperties = mail.smtp.socketFactory.port=465, \` `  mail.smtp.socketFactory.class=javax.net.ssl.SSLSocketFactory, \` `  mail.smtp.socketFactory.fallback=false`
Informational Note:	If you need to pass extra settings to the Java mail library. Comma separated, equals sign between the key and the value. This property is optional.
Property:	`mail.server.disabled`
Example Value:	`mail.server.disabled = false`
Informational Note:	An option is added to disable the mailserver. By default, this property is set to '`false`'. By setting value to '`true`', DSpace will not send out emails. It will instead log the subject of the email which should have been sent. This is especially useful for development and test environments where production data is used when testing functionality. This property is optional.
Property:	`mail.session.name`
Example Value:	`mail.session.name = myDSpace`
Informational Note:	Specifies the name of a javax.mail.Session object stored in JNDI under `java:comp/env/mail`. The default value is "Session".
Property:	`default.language`

Example Value:	`default.language = en_US`
Informational Note:	If no other language is explicitly stated in the *submission-forms.xml*, the default language will be attributed to the metadata values. See also Multilingual Support(see page 407)
Property:	mail.message.headers
Example Value:	mail.message.headers = subject mail.message.headers = charset
Informational Note:	When processing a message template, setting a Velocity variable whose name is one of the values of this configuration property will add or replace a message header of the same name, using the value of the variable as the header's value. See "Templates can set message headers".

Wording of E-mail Messages

Sometimes DSpace automatically sends e-mail messages to users, for example, to inform them of a new work flow task, or as a subscription e-mail alert. The wording of e-mails can be changed by editing the relevant file in `[dspace]/config/emails` . Each file is commented. Be careful to keep the right number 'placeholders' (e.g. *${params[2]}*) for the template's positional parameters.

Each file is a Velocity[469] template. You can use the full Velocity Template Language[470] to help you customize messages. There are two Velocity variables pre-defined by DSpace when processing an e-mail template:

- `params` is the array of message parameters provided by the DSpace code which is sending the message. These are indexed by number, starting at zero.
- `config` is the table of DSpace configuration properties (such as `dspace.name`). These are looked up using `config.get(property name)`.

Sample message template

```
## This is a comment.  It will not be part of the message.
This is the body of the message.  The code which sends it supplied two parameters:  ${params[0]} and $
{params[1]}.
The name of this DSpace instance is ${config.get('dspace.name')} and you can browse it at $
{config.get('dspace.url')}.
```

Also see the template `config/emails/register` for an example of each.

Note: You should replace the contact-information "`dspace-help@myu.edu`[471] `or call us at xxx-555-xxxx`" with your own contact details in:

469 https://velocity.apache.org/
470 https://velocity.apache.org/engine/1.7/vtl-reference.html
471 mailto:dspace-help@myu.edu

```
config/emails/change_password
config/emails/register
```

Templates can set message headers

A template can set specific message headers by defining Velocity variables which have been enabled for this use by naming them as values of the DSpace configuration property `mail.message.headers`. In most cases the name of the Velocity variable will become the header's name, and the value of the variable, the header's value. For example: `#set(My-Header, "Hello World!")` in a template will result in the message having a header `My-Header: Hello World!` if DSpace's mail.message.headers includes "My-Header".

A few Velocity variable names are special in DSpace email templates:

name	meaning
subject	supplies the Subject: header's value.
charset	sets the `charset` parameter of the `Content-Type:` header of the bodypart, when there is a single bodypart. It also causes the subject value to be treated as being encoded in this charset. If not set, the charset defaults to US-ASCII as specified in RFC 2046. If there are multiple bodyparts, all are assumed to be encoded in US-ASCII and `charset` has no effect on them.

Sample message template

```
## This is a comment.  It will not be part of the sent message.
#set($subject = "This will be the Subject: of the message")
This is the body of the message.
```

7.1.3.4 File Storage

> Beginning with DSpace 6, your file storage location (aka bitstore) is now defined in the `[dspace]/config/spring/api/bitstore.xml` Spring configuration file. By default it is defined as the `[dspace]/assetstore/`. More information on modifying your file storage location can be found at Configuring the Bitstream Store (see page 693)in the Storage Layer(see page 686) documentation.

DSpace supports multiple options for storing your repository bitstreams (uploaded files). The files are not stored in the database, instead they are provided via a configured "assetstore" or "bitstore".

By default, the assetstore is simply a directory on your server (`[dspace]/assetstore/`) under which bitstreams (files) are stored by DSpace.

At this time, DSpace supports two primary locations for storing your files:

1. Your local filesystem (used by default), specifically under the `[dspace]/assetstore/` directory
2. OR, Amazon S3[472] (requires your own Amazon S3 account)

472 https://aws.amazon.com/s3/

More information on configuring or customizing the storage location of your files can be found in the Storage Layer(see page 686) documentation.

7.1.3.5 Logging Configuration

> Logging configuration has now moved to `${dspace.dir}/config/log4j2.xml`

Property:	`log.init.config`
Example Value:	`log.init.config = ${dspace.dir}/config/log4j2.xml`
Informational Note:	This is where your logging configuration file is located. You may override the default log4j configuration by providing your own. Existing alternatives are: `log.init.config = ${dspace.dir}/config/log4j2.xml` `log.init.config = ${dspace.dir}/config/log4j2-console.xml`
Property:	`log.dir` (*defined in log4j2.xml*)
Example value:	`log.dir = ${log4j:configParentLocation}/../log`
Informational Note:	This is where to put the DSpace logs. The default setting (shown above) writes all DSpace logs to the ${dspace.dir}/log/ directory.
Property:	`loglevel.dspace` (*defined in log4j2.xml*)
Example value:	`loglevel.dspace = INFO`

Informational Note:	Log level for all DSpace-specific code (org.dspace.* packages). By default, DSpace only provides general INFO logs (in order to keep log sizes reasonable). As necessary, you can temporarily change this setting to any of the following (ordered for most information to least): DEBUG, INFO, WARN, ERROR, FATAL Please be aware we do not recommend running at the DEBUG level in Production for significant periods of time, as it will cause the logs to be extremely large in size.
Property:	`loglevel.other (`*`defined in log4j2.xml`*`)`
Example value:	`loglevel.other = INFO`
Informational Note:	Log level for other third-party tools/APIs used by DSpace (non-DSpace specific code). By default, DSpace only provides general INFO logs (in order to keep log sizes reasonable). As necessary, you can temporarily change this setting to any of the following (ordered for most information to least): DEBUG, INFO, WARN, ERROR, FATAL Please be aware we do not recommend running at the DEBUG level in Production for significant periods of time, as it will cause the logs to be extremely large in size.

7.1.3.6 General Plugin Configuration

Property:	`plugin.classpath`
Example Value:	/opt/dspace/plugins/aPlugin.jar:/opt/dspace/moreplugins
Informational Note:	Search path for third-party plugin classes. This is a colon-separated list of directories and JAR files, each of which will be searched for plugin classes after looking in all the places where DSpace classes are found. In this way you can designate one or more locations for plugin files which will not be affected by DSpace upgrades.

7.1.3.7 Configuring the Search Engine

DSpace's search module is also known as "Discovery" and utilizes Apache Solr for indexing. It provides up-to-date features, such as filtering/faceting, hit highlighting, search snippets, etc.

Detailed documentation is available for customization, see Discovery(see page 386).

7.1.3.8 Handle Server Configuration

The CNRI Handle system is a 3rd party service for maintaining persistent URL's. For a nominal fee, you can register a handle prefix for your repository. As a result, your repository items will be also available under the links http://handle.net/<<handle prefix>>/<<item id>>. As the base url of your repository might change or evolve, the persistent handle.net URL's secure the consistency of links to your repository items. For complete information regarding the Handle server, the user should consult Handle.Net Registry Support(see page 464).

Property:	`handle.canonical.prefix`
Example Value	`handle.canonical.prefix = http://hdl.handle.net/` `handle.canonical.prefix = ${dspace.ui.url}/handle/`
Informational Note:	Canonical Handle URL prefix. By default, DSpace is configured to use http://hdl.handle.net/ as the canonical URL prefix when generating `dc.identifier.uri` during submission, and in the 'identifier' displayed in item record pages. If you do not subscribe to CNRI's handle service, you can change this to match the persistent URL service you use, or you can force DSpace to use your site's URL, e.g. `handle.canonical.prefix = ${dspace.ui.url}/handle/`. Note that this will not alter `dc.identifer.uri` metadata for existing items (only for subsequent submissions).
Property:	`handle.prefix`
Example Value	`handle.prefix = 1234.56789`
Informational Note:	The default installed by DSpace is `123456789` but you will replace this upon receiving a handle from CNRI.
Property:	`handle.dir`
Example Value:	`handle.dir = ${dspace.dir}/handle-server`
Informational Note:	The default files, as shown in the Example Value is where DSpace will install the files used for the Handle Server.
Property	`handle.additional.prefixes`
Example Value	`handle.additional.prefixes = 1234.56789.0, 1234.56789.1, 987`

Informational Note:	List any additional prefixes that need to be managed by this handle server. For example, any handle prefixes that came from an external repository whose items have now been added to this DSpace. Multiple additional prefixes may be added in a comma separated list.

7.1.3.9 Delegation Administration: Authorization System Configuration

It is possible to delegate the administration of Communities and Collections. This functionality eliminates the need for an Administrator Superuser account for these purposes. An EPerson that will be attributed Delegate Admin rights for a certain community or collection will also "inherit" the rights for underlying collections and items. As a result, a community admin will also be collection admin for all underlying collections. Likewise, a collection admin will also gain admin rights for all the items owned by the collection.

Authorization to execute the functions that are allowed to user with WRITE permission on an object will be attributed to be the ADMIN of the object (e.g. community/collection/admin will be always allowed to edit metadata of the object). The default will be "*true*" for all the configurations.

Community Administration: Subcommunities and Collections	
Property:	`core.authorization.community-admin.create-subelement`
Example Value:	`core.authorization.community-admin.create-subelement = true`
Informational Note:	Authorization for a delegated community administrator to create subcommunities or collections.
Property:	`core.authorization.community-admin.delete-subelement`
Example Value:	`core.authorization.community-admin.delete-subelement = true`
Informational Note:	Authorization for a delegated community administrator to delete subcommunities or collections.
Community Administration: Policies and The group of administrators	
Property:	`core.authorization.community-admin.policies`

Example Value:	`core.authorization.community-admin.policies = true`
Informational Note:	Authorization for a delegated community administrator to administrate the community policies.
Property:	`core.authorization.community-admin.admin-group`
Example Value:	`core.authorization.community-admin.admin-group = true`
Informational Note:	Authorization for a delegated community administrator to edit the group of community admins.
Community Administration: Collections in the above Community	
Property:	`core.authorization.community-admin.collection.policies`
Example Value:	`core.authorization.community-admin.collection.policies = true`
Informational Note:	Authorization for a delegated community administrator to administrate the policies for underlying collections.
Property:	`core.authorization.community-admin.collection.template-item`
Example Value:	`core.authorization.community-admin.collection.template-item = true`
Informational Note:	Authorization for a delegated community administrator to administrate the item template for underlying collections.
Property:	`core.authorization.community-admin.collection.submitters`
Example Value:	`core.authorization.community-admin.collection.submitters = true`

Informational Note:	Authorization for a delegated community administrator to administrate the group of submitters for underlying collections.
Property:	`core.authorization.community-admin.collection.workflows`
Example Value:	`core.authorization.community-admin.collection.workflows = true`
Informational Note:	Authorization for a delegated community administrator to administrate the workflows for underlying collections.
Property:	`core.authorization.community-admin.collection.admin-group`
Example Value:	`core.authorization.community-admin.collection.admin-group = true`
Informational Note:	Authorization for a delegated community administrator to administrate the group of administrators for underlying collections.
Community Administration: Items Owned by Collections in the Above Community	
Property:	`core.authorization.community-admin.item.delete`
Example Value:	`core.authorization.community-admin.item.delete = true`
Informational Note:	Authorization for a delegated community administrator to delete items in underlying collections.
Property:	`core.authorization.community-admin.item.withdraw`
Example Value:	`core.authorization.community-admin.item.withdraw = true`
Informational Note:	Authorization for a delegated community administrator to withdraw items in underlying collections.

Property:	`core.authorization.community-admin.item.reinstate`
Example Value:	`core.authorization.community-admin.item.reinstate = true`
Informational Note:	Authorization for a delegated community administrator to reinstate items in underlying collections.
Property:	`core.authorization.community-admin.item.policies`
Example Value:	`core.authorization.community-admin.item.policies = true`
Informational Note:	Authorization for a delegated community administrator to administrate item policies in underlying collections.
Community Administration: Bundles of Bitstreams, related to items owned by collections in the above Community	
Property:	`core.authorization.community-admin.item.create-bitstream`
Example Value:	`core.authorization.community-admin.item.create-bitstream = true`
Informational Note:	Authorization for a delegated community administrator to create additional bitstreams in items in underlying collections.
Property:	`core.authorization.community-admin.item.delete-bitstream`
Example Value:	`core.authorization.community-admin.item.delete-bitstream = true`
Informational Note:	Authorization for a delegated community administrator to delete bitstreams from items in underlying collections.
Property:	`core.authorization.community-admin.item.cc-license`

Example Value:	`core.authorization.community-admin.item.cc-license = true`
Informational Note:	Authorization for a delegated community administrator to administer licenses from items in underlying collections.
Collection Administration: The properties for collection administrators work similar to those of community administrators, with respect to collection administration.	`core.authorization.collection-admin.policies` `core.authorization.collection-admin.template-item` `core.authorization.collection-admin.submitters` `core.authorization.collection-admin.workflows` `core.authorization.collection-admin.admin-group`

Collection Administration: Item owned by the above Collection. The properties for collection administrators work similar to those of community administrators, with respect to administration of items in underlying collections.	`core.authorization.collection-admin.item.delete` `core.authorization.collection-admin.item.withdraw` `core.authorization.collection-admin.item.reinstatiate` `core.authorization.collection-admin.item.policies`

Collection Administration: Bundles of bitstreams, related to items owned by collections in the above Community. The properties for collection administrators work similar to those of community administrators, with respect to administration of bitstreams related to items in underlying collections.	`core.authorization.collection-admin.item.create-bitstream` `core.authorization.collection-admin.item.delete-bitstream` `core.authorization.collection-admin.item-admin.cc-license`

Item Administration. The properties for item administrators work similar to those of community and collection administrators, with respect to administration of items in underlying collections.	`core.authorization.item-admin.policies`

Item Administration: Bundles of bitstreams, related to items owned by collections in the above Community. The properties for item administrators work similar to those of community and collection administrators, with respect to administration of bitstreams related to items in underlying collections.	`core.authorization.item-admin.create-bitstream` `core.authorization.item-admin.delete-bitstream` `core.authorization.item-admin.cc-license`

7.1.3.10 Login as feature

Property:	`webui.user.assumelogin`
Example Value:	`webui.user.assumelogin = true`
Informational Note:	Determine if super administrators (those whom are in the Administrators group) can login as another user from the "edit eperson" page. This is useful for debugging problems in a running DSpace instance, especially in the workflow process. The default value is false, i.e., no one may assume the login of another user.

7.1.3.11 Restricted Item Visibility Settings

By default RSS feeds and subscription emails will include ALL items regardless of permissions set on them. If you wish to only expose items through these channels where the ANONYMOUS user is granted READ permission, then set the following options to false.

Property:	`harvest.includerestricted.rss`
Example Value:	`harvest.includerestricted.rss = true`
Informational Note:	When set to 'true' (default), items that haven't got the READ permission for the ANONYMOUS user, will be included in RSS feeds anyway.
Property:	`harvest.includerestricted.subscription`
Example Value:	`harvest.includerestricted.subscription = true`
Informational Note:	When set to true (default), items that haven't got the READ permission for the ANONYMOUS user, will be included in Subscription emails anyway.

7.1.3.12 Proxy Settings

These settings for proxy are commented out by default. Uncomment and specify both properties if proxy server is required for external http requests. Use regular host name without port number.

Property:	`http.proxy.host`
Example Value	`http.proxy.host = proxy.myu.edu`
Informational Note	Enter the host name without the port number. *Only currently used for Creative Commons licensing feature (to contact their API), and Sitemap generation (to ping search server regarding updates)*
Property:	`http.proxy.port`
Example Value	`http.proxy.port = 2048`

Informational Note	Enter the port number for the proxy server. *Only currently used for Creative Commons licensing feature (to contact their API), and Sitemap generation (to ping search server regarding updates)*
Property	`useProxies`
Example Value:	`useProxies = true`
Informational Note:	As of DSpace 7 (and above), this setting defaults to **true**. If "useProxies" is enabled, the authentication and statistics logging code will read the X-Forwarded-For header in order to determine the correct client IP address. As the User Interface uses Angular Universal[473] (for SEO support), the proxy server that comes with Angular Universal is always enabled. By default, only your local server (127.0.0.1) and the public IP address of `dspace.ui.url` are "trusted" as a proxy. If your DSpace instance is protected by external proxy server, you may need to update the "proxies.trusted.ipranges" property below. This also affects IPAuthentication, and should be enabled for that to work properly if your installation uses a proxy server.
Property	proxies.trusted.ipranges
Example Value:	proxies.trusted.ipranges `= 127.0.0.1`
Informational Note:	By default, only proxies running on localhost (127.0.0.1) and the `dspace.ui.url` (public IP address) are "trusted". This allows our Angular User Interface to communicate with the REST API via a trusted proxy, which is required for Angular Universal[474] (for SEO support). You can specify a range by only listing the first three ip-address blocks, e.g. 128.177.243 You can list multiple IP addresses or ranges by comma-separating them.
Property	proxies.trusted.include_ui_ip
Example Value:	proxies.trusted.include_ui_ip = true

473 https://angular.io/guide/universal
474 https://angular.io/guide/universal

Informational Note:	This setting specifies whether to automatically trust IP address of the `dspace.ui.url` as a proxy. By default, this is always set to true to ensure the UI is fully trusted by the backend. However, if you are not using the Angular UI, you may choose to set this to "false" in order to only trust proxies running on localhost (127.0.0.1) by default.
Property	server.forward-headers-strategy
Example Value:	server.forward-headers-strategy = FRAMEWORK
Informational Note:	This is a Spring Boot setting which may be overridden/specified in your local.cfg. By default, Spring Boot does not automatically use X-Forwarded-* Headers when generating links (and similar) in the REST API. When using a proxy in front of the REST API, you may need to modify this setting: • NATIVE = allows your web server to natively support standard Forwarded headers • FRAMEWORK = enables Spring Framework's built in filter to manage these headers in Spring Boot. (This value may be useful to set for DSpace if you find that X-Forwarded headers are not working) • NONE = default value. Forwarded headers are ignored For more information see the Spring Boot docs at https://docs.spring.io/spring-boot/docs/current/reference/html/howto.html#howto-use-behind-a-proxy-server

7.1.3.13 Configuring Media Filters

Media or Format Filters are classes used to generate derivative or alternative versions of content or bitstreams within DSpace. For example, the PDF Media Filter will extract textual content from PDF bitstreams, the JPEG Media Filter can create thumbnails from image bitstreams.

Media Filters are configured as Named Plugins, with each filter also having a separate configuration setting (in *dspace.cfg*) indicating which formats it can process. The default configuration is shown below.

Property:	`filter.plugins`
Example Value:	`filter.plugins = PDF Text Extractor` `filter.plugins = Html Text Extractor` `filter.plugins = Word Text Extractor` `filter.plugins = JPEG Thumbnail`
Informational Note:	This setting lists the names of all enabled MediaFilter or FormatFilter plugins. To enable multiple plugins, list them on separate lines (as shown above) or provide a comma separated list.

Property:	`plugin.named.org.dspace.app.mediafilter.FormatFilter`
Example Value:	`plugin.named.org.dspace.app.mediafilter.FormatFilter = org.dspace.app.mediafilter.PDFFilter = PDF Text Extractor` `plugin.named.org.dspace.app.mediafilter.FormatFilter = org.dspace.app.mediafilter.HTMLFilter = HTML Text Extractor` `plugin.named.org.dspace.app.mediafilter.FormatFilter = org.dspace.app.mediafilter.WordFilter = Word Text Extractor` `plugin.named.org.dspace.app.mediafilter.FormatFilter = org.dspace.app.mediafilter.JPEGFilter = JPEG Thumbnail` `plugin.named.org.dspace.app.mediafilter.FormatFilter = org.dspace.app.mediafilter.BrandedPreviewJPEGFilter = Branded Preview JPEG`
Informational Note:	Assign "human-understandable" names to each filter. These names are used to enable/disable plugins using "filter.plugins" setting above. As with the previous setting, multiple plugins can be listed here on separate lines (or comma separated)
Property:	`filter.org.dspace.app.mediafilter.PDFFilter.inputFormats` `filter.org.dspace.app.mediafilter.HTMLFilter.inputFormats` `filter.org.dspace.app.mediafilter.WordFilter.inputFormats` `filter.org.dspace.app.mediafilter.JPEGFilter.inputFormats` `filter.org.dspace.app.mediafilter.BrandedPreviewJPEGFilter.inputFormats`
Example Value:	`filter.org.dspace.app.mediafilter.PDFFilter.inputFormats = Adobe PDF` `filter.org.dspace.app.mediafilter.HTMLFilter.inputFormats = HTML, Text` `filter.org.dspace.app.mediafilter.WordFilter.inputFormats = Microsoft Word` `filter.org.dspace.app.mediafilter.JPEGFilter.inputFormats = BMP, GIF, JPEG, \` `image/png` `filter.org.dspace.app.mediafilter.BrandedPreviewJPEGFilter.inputFormats = BMP, \` `GIF, JPEG, image/png`
Informational Note:	Configure each filter's input format(s). These must match format names in the DSpace file format registry.

Property:	filter.org.dspace.app.mediafilter.publicPermission
Example Value:	filter.org.dspace.app.mediafilter.publicPermission = JPEGFilter
	Optionally, configure filter(s) which should **always** create publicly accessible bitstreams (e.g. useful if you want thumbnails to always be publicly accessible). By default, any bitstreams created by a filter will inherit the same permissions as the original file (e.g. if original image is access restricted, then thumbnail will also be access restricted by default).
Property:	`pdffilter.largepdfs`
Example Value:	`pdffilter.largepdfs = true`
Informational Note:	It this value is set for "true", all PDF extractions are written to temp files as they are indexed. This is slower, but helps to ensure that PDFBox software DSpace uses does not eat up all your memory.
Property:	`pdffilter.skiponmemoryexception`
Example Value:	`pdffilter.skiponmemoryexception = true`
Informational Note:	If this value is set for "true", PDFs which still result in an "Out of Memory" error from PDFBox are skipped over. These problematic PDFs will never be indexed until memory usage can be decreased in the PDFBox software.

Names are assigned to each filter using the `plugin.named.org.dspace.app.mediafilter.FormatFilter` field (e.g. by default the PDFilter is named "PDF Text Extractor".

Finally, the appropriate `filter.<class path>.inputFormats` defines the valid input formats which each filter can be applied. These format names **must match** the `short description` field of the Bitstream Format Registry.

You can also implement more dynamic or configurable Media/Format Filters which extend `SelfNamedPlugin`.

> **More Information on MediaFilters**
>
> For more information on Media/Format Filters, see the section on Mediafilters for Transforming DSpace Content(see page 469).

7.1.3.14 Crosswalk and Packager Plugin Settings

The subsections below give configuration details based on the types of crosswalks and packager plugins you need to implement.

> **More Information on Packagers & Crosswalks**
>
> For more information on using Packagers and Crosswalks, see the section on Importing and Exporting Content via Packages(see page 244).

Configurable MODS Dissemination Crosswalk

The MODS crosswalk is a self-named plugin. To configure an instance of the MODS crosswalk, add a property to the DSpace configuration starting with "`crosswalk.mods.properties.`"; the final word of the property name becomes the plugin's name. For example, a property name `crosswalk.mods.properties.MODS` defines a crosswalk plugin named "`MODS`".

The value of this property is a path to a separate properties file containing the configuration for this crosswalk. The pathname is relative to the DSpace configuration directory, i.e. the `config` subdirectory of the DSpace install directory. Example from the `dspace.cfg` file:

Properties:	`crosswalk.mods.properties.MODS` `crosswalk.mods.properties.mods`
Example Values:	`crosswalk.mods.properties.MODS = crosswalks/mods.properties` `crosswalk.mods.properties.mods = crosswalks/mods.properties`
Informational Note:	This defines a crosswalk named MODS whose configuration comes from the file `[dspace]/config/crosswalks/mods.properties` . (In the above example, the lower-case name was added for OAI-PMH)

The MODS crosswalk properties file is a list of properties describing how DSpace metadata elements are to be turned into elements of the MODS XML output document. The property name is a concatenation of the metadata schema, element name, and optionally the qualifier. For example, the *contributor.author* element in the native Dublin Core schema would be: *dc.contributor.author*. The value of the property is a line containing two segments separated by the vertical bar ("|"_): The first part is an XML fragment which is copied into the output document. The second is an XPath expression describing where in that fragment to put the value of the metadata element. For example, in this property:

```
dc.contributor.author = <mods:name>
                            <mods:role>
                                <mods:roleTerm type="text">author</mods:roleTerm>
                            </mods:role>
                            <mods:namePart>%s</mods:namePart>
                        </mods:name>
```

Some of the examples include the string "`%s`" in the prototype XML where the text value is to be inserted, but don't pay any attention to it, it is an artifact that the crosswalk ignores. For example, given an author named *Jack Florey*, the crosswalk will insert

```
<mods:name>
    <mods:role>
        <mods:roleTerm type="text">author</mods:roleTerm>
    </mods:role>
    <mods:namePart>Jack Florey</mods:namePart>
</mods:name>
```

into the output document. Read the example configuration file for more details.

XSLT-based Crosswalks

The XSLT crosswalks use XSL stylesheet transformation (XSLT) to transform an XML-based external metadata format to or from DSpace's internal metadata. XSLT crosswalks are much more powerful and flexible than the configurable MODS and QDC crosswalks, but they demand some esoteric knowledge (XSL stylesheets). Given that, you can create all the crosswalks you need just by adding stylesheets and configuration lines, without touching any of the Java code.

The default settings in the `dspace.cfg` file for submission crosswalk:

Properties:	`crosswalk.submission.MODS.stylesheet`
Example Value:	`crosswalk.submission.MODS.stylesheet = crosswalks/mods-submission.xsl`
Informational Note:	Configuration XSLT-driven submission crosswalk for MODS

As shown above, there are three (3) parts that make up the properties "key":

```
crosswalk.submission.PluginName.stylesheet =
          1        2        3               4
```

`crosswalk` first part of the property key.
`submission` second part of the property key.
`PluginName` is the name of the plugin. The *path* value is the path to the file containing the crosswalk stylesheet (relative to `/[dspace]/config`).
Here is an example that configures a crosswalk named "LOM" using a stylesheet in `[dspace]/config/crosswalks/d-lom.xsl`:
`crosswalk.submission.LOM.stylesheet = crosswalks/d-lom.xsl`
A dissemination crosswalk can be configured by starting with the property key *crosswalk.dissemination*. Example:
`crosswalk.dissemination.PluginName.stylesheet = path`
The *PluginName* is the name of the plugin (!) . The *path* value is the path to the file containing the crosswalk stylesheet (relative to `/[dspace]/config`).

You can make two different plugin names point to the same crosswalk, by adding two configuration entries with the same path:

```
crosswalk.submission.MyFormat.stylesheet = crosswalks/myformat.xslt
     crosswalk.submission.almost_DC.stylesheet = crosswalks/myformat.xslt
```

The dissemination crosswalk must also be configured with an XML Namespace (including prefix and URI) and an XML schema for its output format. This is configured on additional properties in the DSpace configuration:

```
crosswalk.dissemination.PluginName.namespace.Prefix = namespace-URI
     crosswalk.dissemination.PluginName.schemaLocation = schemaLocation value
```

For example:

```
crosswalk.dissemination.qdc.namespace.dc = http://purl.org/dc/elements/1.1/
     crosswalk.dissemination.qdc.namespace.dcterms = http://purl.org/dc/terms/
     crosswalk.dissemination.qdc.schemalocation = http://purl.org/dc/elements/1.1/ \
     http://dublincore.org/schemas/xmls/qdc/2003/04/02/qualifieddc.xsd
```

If you remove all XSLTDisseminationCrosswalks please disable the XSLTDisseminationCrosswalk in the list of selfnamed plugins. If no XSLTDisseminationCrosswalks are configured but the plugin is loaded the PluginManager will log an error message ("Self-named plugin class "org.dspace.content.crosswalk.XSLTDisseminationCrosswalk" returned null or empty name list!").

Testing XSLT Crosswalks

The XSLT crosswalks will automatically reload an XSL stylesheet that has been modified, so you can edit and test stylesheets without restarting DSpace. You can test a crosswalk by using a command-line utitlity. To test a dissemination crosswalk you have to run:

```
[dspace]/bin/dspace dsrun org.dspace.content.crosswalk.XSLTDisseminationCrosswalk <plugin name> <handle>
[output-file]
```

For example, you can test the marc plugin on the handle 123456789/3 with:

```
[dspace]/bin/dspace dsrun org.dspace.content.crosswalk.XSLTDisseminationCrosswalk marc 123456789/3
```

Informations from the script will be printed to stderr while the XML output of the dissemination crosswalk will be printed to stdout. You can give a third parameter containing a filename to write the output into a file, but be careful: the file will be overwritten if it exists.

Testing a submission crosswalk works quite the same way. Use the following command-line utility, it calls the crosswalk plugin to translate an XML document you submit, and displays the resulting intermediate XML (DIM). Invoke it with:

```
[dspace]/bin/dspace dsrun
     org.dspace.content.crosswalk.XSLTIngestionCrosswalk [-l] <plugin name> <input-file>
```

where <*plugin name*> is the name of the crosswalk plugin to test (e.g. "LOM"), and <*input-file*> is a file containing an XML document of metadata in the appropriate format.

Add the `-l` option to pass the ingestion crosswalk a list of elements instead of a whole document, as if the List form of the ingest() method had been called. This is needed to test ingesters for formats like DC that get called with lists of elements instead of a root element.

Configurable Qualified Dublin Core (QDC) dissemination crosswalk

The QDC crosswalk is a self-named plugin. To configure an instance of the QDC crosswalk, add a property to the DSpace configuration starting with "`crosswalk.qdc.properties.`"; the final word of the property name becomes the plugin's name. For example, a property name `crosswalk.qdc.properties.QDC` defines a crosswalk plugin named "`QDC`".

The following is from *dspace.cfg* file:

Properties:	`crosswalk.qdc.namspace.qdc.dc`
Example Value:	`crosswalk.qdc.namspace.qdc.dc = http://purl.org/dc/elements/1.1_`
Properties:	`crosswalk.qdc.namspace.qdc.dcterms`
Example Value:	`crosswalk.qdc.namspace.qdc.dc = http://purl.org/dc/terms/_`
Properties:	`crosswalk.qdc.schemaLocation.QDC`
Example Value:	`crosswalk.qdc.schemaLocation.QDC = http://www.purl.org/dc/terms \` `http://dublincore.org/schemas/xmls/qdc/2006/01/06/dcterms.xsd \` `http://purl.org/dc/elements/1.1 \` `http://dublincore.org/schemas/xmls/qdc/2006/01/06/dc.xsd`
Properties:	`crosswalk.qdc.properties.QDC`
Example Value:	`crosswalk.qdc.properties.QDC = crosswalks/QDC.properties`
Informational Note:	Configuration of the QDC Crosswalk dissemination plugin for Qualified DC. *(Add lower-case name for OAI-PMH. That is, change QDC to qdc.)*}}

In the property key "`crosswalk.qdc.properties.QDC`" the value of this property is a path to a separate properties file containing the configuration for this crosswalk. The pathname is relative to the DSpace configuration directory `/[dspace]/config`. Referring back to the "Example Value" for this property key, one has `crosswalks/`

`qdc.properties` which defines a crosswalk named `QDC` whose configuration comes from the file `[dspace]/config/crosswalks/qdc.properties`.

You will also need to configure the namespaces and schema location strings for the XML output generated by this crosswalk. The namespaces properties names are formatted:

`crosswalk.qdc.namespace.`*prefix* = *uri*

where *prefix* is the namespace prefix and *uri* is the namespace URI. See the above Property and Example Value keys as the default dspace.cfg has been configured.

The QDC crosswalk properties file is a list of properties describing how DSpace metadata elements are to be turned into elements of the Qualified DC XML output document. The property name is a concatenation of the metadata schema, element name, and optionally the qualifier. For example, the `contributor.author` element in the native Dublin Core schema would be: `dc.contributor.author`. The value of the property is an XML fragment, the element whose value will be set to the value of the metadata field in the property key.

For example, in this property:

```
dc.coverage.temporal = <dcterms:temporal />
```

the generated XML in the output document would look like, e.g.:
`<dcterms:temporal>Fall, 2005</dcterms:temporal>`

Configuring Crosswalk Plugins

Ingestion crosswalk plugins are configured as named or self-named plugins for the interface `org.dspace.content.crosswalk.IngestionCrosswalk`. Dissemination crosswalk plugins are configured as named or self-named plugins for the interface `org.dspace.content.crosswalk.DisseminationCrosswalk`.

You can add names for existing crosswalks, add new plugin classes, and add new configurations for the configurable crosswalks as noted below.

Configuring Packager Plugins

Package ingester plugins are configured as named or self-named plugins for the interface `org.dspace.content.packager.PackageIngester`. Package disseminator plugins are configured as named or self-named plugins for the interface `org.dspace.content.packager.PackageDisseminator`.

You can add names for the existing plugins, and add new plugins, by altering these configuration properties. See the Plugin Manager architecture(see page 552) for more information about plugins.

7.1.3.15 Event System Configuration

If you are unfamiliar with the Event System in DSpace, and require additional information with terms like "Consumer" and "Dispatcher" please refer to EventSystemPrototype[475].

Property:	`event.dispatcher.default.class`
Example Value:	`event.dispatcher.default.class = org.dspace.event.BasicDispatcher`

475 https://wiki.lyrasis.org/display/DSPACE/EventSystemPrototype

Informational Note:	This is the default synchronous dispatcher (Same behavior as traditional DSpace).
Property:	`event.dispatcher.default.consumers`
Example Value:	`event.dispatcher.default.consumers = search, browse, eperson`
Informational Note:	This is the default synchronous dispatcher (Same behavior as traditional DSpace).
Property:	`event.dispatcher.noindex.class`
Example Value:	`event.dispatcher.noindex.class = org.dspace.event.BasicDispatcher`
Informational Note:	The noindex dispatcher will not create search or browse indexes (useful for batch item imports).
Property:	`event.dispatcher.noindex.consumers`
Example Value:	`event.dispatcher.noindex.consumers = eperson`
Informational Note:	The noindex dispatcher will not create search or browse indexes (useful for batch item imports).
Property:	`event.consumer.discovery.class`
Example Value:	`event.consumer.discovery.class = org.dspace.discovery.IndexEventConsumer`
Informational Note:	Consumer to maintain the search/browse (Discovery) index.
Property:	`event.consumer.discovery.filters`

Example Value:	`event.consumer.discovery.filters = Community | Collection | Item | Bundle | Site+Add | Create | Modify | Modify_Metadata | Delete | Remove`
Informational Note:	Consumer to maintain the search/browse (Discovery) index.
Property:	`event.consumer.eperson.class`
Example Value:	`event.consumer.eperson.class = org.dspace.eperson.EPersonConsumer`
Informational Note:	Consumer related to EPerson changes
Property:	`event.consumer.eperson.filters`
Example Value:	`event.consumer.eperson.filters = EPerson+Create`
Informational Note:	Consumer related to EPerson changes
Property:	`event.consumer.test.class`
Example Value:	`event.consumer.test.class = org.dspace.event.TestConsumer`
Informational Note:	Test consumer for debugging and monitoring. **Commented out by default.**
Property:	`event.consumer.test.filters`
Example Value:	`event.consumer.test.filters = All+All`
Informational Note:	Test consumer for debugging and monitoring. **Commented out by default.**
Property:	`testConsumer.verbose`

Example Value:	`testConsumer.verbose = true`
Informational Note:	Set this to true to enable testConsumer messages to standard output. **Commented out by default.**

7.1.3.16 Embargo

DSpace embargoes utilize standard metadata fields to hold both the "terms" and the "lift date". Which fields you use are configurable, and no specific metadata element is dedicated or predefined for use in embargo. Rather, you specify exactly what field you want the embargo system to examine when it needs to find the terms or assign the lift date.

Property:	`embargo.field.terms`
Example Value:	`embargo.field.terms = SCHEMA.ELEMENT.QUALIFIER`
Informational Note:	Embargo terms will be stored in the item metadata. This property determines in which metadata field these terms will be stored. An example could be dc.embargo.terms
Property:	`embargo.field.lift`
Example Value:	`embargo.field.lift = SCHEMA.ELEMENT.QUALIFIER`
Informational Note:	The Embargo lift date will be stored in the item metadata. This property determines in which metadata field the computed embargo lift date will be stored. You may need to create a DC metadata field in your Metadata Format Registry if it does not already exist. An example could be dc.embargo.liftdate
Property:	`embargo.terms.open`
Example Value:	`embargo.terms.open = forever`
Informational Note:	You can determine your own values for the embargo.field.terms property (see above). This property determines what the string value will be for indefinite embargos. The string in terms field to indicate indefinite embargo.

Property:	`plugin.single.org.dspace.embargo.EmbargoSetter`
Example Value:	`plugin.single.org.dspace.embargo.EmbargoSetter = org.dspace.embargo.DefaultEmbargoSetter`
Informational Note:	To implement the business logic to set your embargos, you need to override the EmbargoSetter class. If you use the value DefaultEmbargoSetter, the default implementation will be used.
Property:	`plugin.single.org.dspace.embargo.EmbargoLifter`
Example Value:	`plugin.single.org.dspace.embargo.EmbargoLifter = org.dspace.embargo.DefaultEmbargoLifter`
Informational Note:	To implement the business logic to lift your embargos, you need to override the EmbargoLifter class. If you use the value DefaultEmbargoLifter, the default implementation will be used.

More Embargo Details

More details on Embargo configuration, including specific examples can be found in the Embargo(see page 113) section of the documentation.

7.1.3.17 Checksum Checker Settings

DSpace comes with a Checksum Checker script (`[dspace]/bin/dspace checker`) which can be scheduled to verify the checksum of every item within DSpace. Since DSpace calculates and records the checksum of every file submitted to it, this script is able to determine whether or not a file has been changed (either manually or by some sort of corruption or virus). The idea being that the earlier you can identify a file has changed, the more likely you'd be able to recover it (assuming it was not a wanted change).

Property:	`plugin.single.org.dspace.checker.BitstreamDispatcher`
Example Value:	`plugin.single.org.dspace.checker.BitstreamDispatcher = org.dspace.checker.SimpleDispatcher`
Informational Note:	The Default dispatcher is case non is specified.
Property:	`checker.retention.default`

Example Value:	`checker.retention.default = 10y`
Informational Note:	This option specifies the default time frame after which all checksum checks are removed from the database (defaults to 10 years). This means that after 10 years, all successful or unsuccessful matches are removed from the database.
Property:	`checker.retention.CHECKSUM_MATCH`
Example Value:	`checker.retention.CHECKSUM_MATCH = 8w`
Informational Note:	This option specifies the time frame after which a successful match will be removed from your DSpace database (defaults to 8 weeks). This means that after 8 weeks, all successful matches are automatically deleted from your database (in order to keep that database table from growing too large).

More Checksum Checking Details

For more information on using DSpace's built-in Checksum verification system, see the section on Validating CheckSums of Bitstreams(see page 493).

7.1.3.18 Item Export and Download Settings

It is possible for an authorized user to request a complete export and download of a DSpace item in a compressed zip file. This zip file may contain the following:
dublin_core.xml
license.txt
contents *(listing of the contents)*
handle *file itself and the extract file if available*

The configuration settings control several aspects of this feature:

Property:	`org.dspace.app.itemexport.work.dir`
Example Value:	`org.dspace.app.itemexport.work.dir = ${dspace.dir}/exports`
Informational Note:	The directory where the exports will be done and compressed.
Property:	`org.dspace.app.itemexport.download.dir`

Example Value:	`org.dspace.app.itemexport.download.dir = ${dspace.dir}/exports/download`
Informational Note	The directory where the compressed files will reside and be read by the downloader.
Property:	`org.dspace.app.itemexport.life.span.hours`
Example Value:	`org.dspace.app.itemexport.life.span.hours = 48`
Informational Note	The length of time in hours each archive should live for. When new archives are created this entry is used to delete old ones.
Property:	`org.dspace.app.itemexport.max.size`
Example Value:	`org.dspace.app.itemexport.max.size = 200`
Informational Note	The maximum size in Megabytes (Mb) that the export should be. This is enforced before the compression. Each bitstream's size in each item being exported is added up, if their cumulative sizes are more than this entry the export is not kicked off.

7.1.3.19 Subscription Emails

DSpace, through some advanced installation and setup, is able to send out an email to collections that a user has subscribed. The user who is subscribed to a collection is emailed each time an item id added or modified. The following property key controls whether or not a user should be notified of a modification.

Property:	`eperson.subscription.onlynew`
Example Value:	`eperson.subscription.onlynew = true`
Informational Note:	For backwards compatibility, the subscription emails by default include any modified items. The property key is COMMENTED OUT by default.

7.1.3.20 Hiding Metadata

It is now possible to hide metadata from public consumption that is only available to the Administrator.

Property:	`metadata.hide.dc.description.provenance`
Example Value:	`metadata.hide.dc.description.provenance = true`
Informational Note:	Hides the metadata in the property key above except to the administrator. Fields named here are hidden in the following places UNLESS the logged-in user is an Administrator: 1. REST API (and therefore the UI) 2. RDF (everywhere as there is currently no possibility to authenticate) 3. OAI-PMH server (everywhere as there is currently no possibility to authenticate) To designate a field as hidden, add a property here in the form: `metadata.hide.SCHEMA.ELEMENT.QUALIFIER = true`. This default configuration hides the `dc.description.provenance` field, since that usually contains email addresses which ought to be kept private and is mainly of interest to administrators.

7.1.3.21 Settings for the Submission Process

Property:	`webui.submit.upload.required`
Example Value:	`webui.submit.upload.required = true`
Informational Note:	Whether or not a file is **required** to be uploaded during the "Upload" step in the submission process. The default is true. If set to "false", then the submitter (human being) has the option to skip the uploading of a file.

7.1.3.22 Configuring the Sherpa/RoMEO Integration

DSpace has integration with the Sherpa/RoMEO API[476] in order to allow importing data from Sherpa/RoMEO during the submission You must register for a free API key (see below for details).

Property:	sherpa.romeo.url
Example Value:	sherpa.romeo.url = https://v2.sherpa.ac.uk/cgi/retrieve

476 https://v2.sherpa.ac.uk/romeo/

Informational Note:	The Sherpa/RoMEO endpoint.
Property:	sherpa.romeo.apikey
Example Value:	`sherpa.romeo.apikey = YOUR-API-KEY`
Informational Note:	Allow to use a specific API key to raise the usage limit (500 calls/day for unregistred user). You MUST register for a free api access key at https://v2.sherpa.ac.uk/api/

The functionality rely on understanding to which Journal (ISSN) is related the submitting item. This is done out of box looking to some item metadata but a different strategy can be used as for example look to a metadata authority in the case that the Sherpa/RoMEO autocomplete for Journal is used (see AuthorityControlSettings(see page 625))

The strategy used to discover the Journal related to the submission item is defined in the spring file **/config/spring/api/sherpa.xml**

```
<bean class="org.dspace.app.sherpa.submit.SHERPASubmitConfigurationService"
        id="org.dspace.app.sherpa.submit.SHERPASubmitConfigurationService">
        <property name="issnItemExtractors">
            <list>
                <bean class="org.dspace.app.sherpa.submit.MetadataValueISSNExtractor">
                    <property name="metadataList">
                        <list>
                            <value>dc.identifier.issn</value>
                        </list>
                    </property>
                </bean>
                <!-- Use the follow if you have the SHERPARoMEOJournalTitle enabled
                <bean class="org.dspace.app.sherpa.submit.MetadataAuthorityISSNExtractor">
                    <property name="metadataList">
                        <list>
                            <value>dc.title.alternative</value>
                        </list>
                    </property>
                </bean>  -->
            </list>
        </property>
    </bean>
```

7.1.3.23 Configuring Creative Commons License

The following configurations are for the Creative Commons license step in the submission process. Submitters are given an opportunity to select a Creative Common license to accompany the item. Creative Commons licenses

govern the use of the content. For further details, refer to the Creative Commons website at http://creativecommons.org[477] .

Creative Commons licensing is optionally available and may be configured for any given collection that has a defined submission sequence, or be part of the "default" submission process. This process is described in the Submission User Interface(see page 260) section of this manual. There is a Creative Commons step already defined, but it is commented out, so enabling Creative Commons licensing is typically just a matter of uncommenting that step.

When enabled, the Creative Commons public API is utilized. This allows DSpace to store metadata references to the selected CC license, while also storing the CC License as a bitstream. The following CC License information are captured:

- The URL of the CC License is stored in the "dc.rights.uri" metadata field (or whatever field is configured in the "cc.license.uri" setting below)
- The name of the CC License is stored in the "dc.rights" metadata field (or whatever field is configured in the "cc.license.name" setting below). This only occurs if "cc.submit.setname=true" (default value)
- The RDF version of the CC License is stored in a bitstream named "license_rdf" in the CC-LICENSE bundle (as long as "cc.submit.addbitstream=true", which is the default value)

The following configurations (in dspace.cfg) relate to the Creative Commons license process:

Property:	`cc.api.rooturl`
Example Value:	`cc.api.rooturl = http://api.creativecommons.org/rest/1.5`
Informational Note:	Generally will never have to assign a different value - this is the base URL of the Creative Commons service API.
Property:	`cc.license.uri`
Example Value:	`cc.license.uri = dc.rights.uri`
Informational Note:	The field that holds the Creative Commons license URI.
Property:	`cc.license.name`
Example Value:	`cc.license.name = dc.rights`
Informational Note:	The field that holds the Creative Commons license Name.

477 http://creativecommons.org/

Property:	`cc.submit.setname`
Example Value:	`cc.submit.setname = true`
Informational Note:	If true, the license assignment will add the field configured with the "cc.license.name" with the name of the CC license; if false, only "cc.license.uri" field is added.
Property:	`cc.submit.addbitstream`
Example Value:	`cc.submit.addbitstream = true`
Informational Note:	If true, the license assignment will add a bitstream with the CC license RDF; if false, only metadata field(s) are added.
Property:	`cc.license.classfilter`
Example Value:	`cc.license.classfilter = recombo,mark`
Informational Note:	This list defines the values that will be excluded from the license (class) selection list, as defined by the web service at the URL: http://api.creativecommons.org/rest/1.5/classes
Property:	`cc.license.jurisdiction`
Example Value:	`cc.license.jurisdiction = nz`
Informational Note:	Should a jurisdiction be used? If so, which one? See http://creativecommons.org/international/ for a list of possible codes (e.g. nz = New Zealand, uk = England and Wales, jp = Japan) Commenting out this field will cause DSpace to select the latest, unported CC license (currently version 4.0). However, as Creative Commons 4.0 does not provide jurisdiction specific licenses, if you specify this setting, your DSpace will continue to use older, Creative Commons 3.0 jurisdiction licenses.
Property	`cc.license.locale`

Example Value:	`cc.license.locale` = en
Informational Note:	Locale to be used (in the form: language or language_country), e.g. "en" or "en_US" If no default locale is defined the Creative Commons default locale will be used.

7.1.3.24 WEB User Interface Configurations

General Web User Interface Configurations

Property:	`webui.licence_bundle.show`
Example Value:	`webui.licence_bundle.show = false`
Informational Note:	Sets whether to display the contents of the license bundle (often just the deposit license in the standard DSpace installation). **UNSUPPORTED in DSpace 7.0**
Property:	`thumbnail.maxwidth`
Example Value:	`thumbnail.maxwidth = 80`
Informational Note:	This property sets the maximum width of generated thumbnails that are being displayed on item pages.
Property:	`thumbnail.maxheight`
Example Value:	`thumbnail.maxheight = 80`
Informational Note:	This property sets the maximum height of generated thumbnails that are being displayed on item pages.
Property:	`webui.preview.maxwidth`
Example Value:	`webui.preview.maxwidth = 600`

Informational Note:	This property sets the maximum width for the preview image. Only used for BrandedPreviewJPEGFilter
Property:	`webui.preview.maxheight`
Example Value:	`webui.preview.maxheight = 600`
Informational Note:	This property sets the maximum height for the preview image. Only used for BrandedPreviewJPEGFilter
Property:	`webui.preview.brand`
Example Value:	`webui.preview.brand = My Institution Name`
Informational Note:	This is the brand text that will appear with the image. Only used for BrandedPreviewJPEGFilter
Property:	`webui.preview.brand.abbrev`
Example Value:	`webui.preview.brand.abbrev = MyOrg`
Informational Note:	An abbreviated form of the full Branded Name. This will be used when the preview image cannot fit the normal text. Only used for BrandedPreviewJPEGFilter
Property:	`webui.preview.brand.height`
Example Value:	`webui.preview.brand.height = 20`
Informational Note:	The height (in px) of the brand. Only used for BrandedPreviewJPEGFilter
Property:	`webui.preview.brand.font`
Example Value:	`webui.preview.brand.font = SansSerif`

Informational Note:	This property sets the font for your Brand text that appears with the image. Only used for BrandedPreviewJPEGFilter
Property:	`webui.preview.brand.fontpoint`
Example Value:	`webui.preview.brand.fontpoint = 12`
Informational Note:	This property sets the font point (size) for your Brand text that appears with the image. Only used for BrandedPreviewJPEGFilter
Property:	`webui.preview.dc`
Example Value:	`webui.preview.dc = rights`
Informational Note:	The Dublin Core field that will display along with the preview. This field is optional. Only used for BrandedPreviewJPEGFilter
Property:	`webui.strengths.cache`
Example Value:	`webui.strengths.cache = false`
Informational Note:	When showing the strengths (i.e. item counts), should they be counted in real time, or fetched from the cache. Counts fetched in real time will perform an actual count of the index contents every time a page with this feature is requested, which may not scale. If you set the property key is set to cache ("true"), the counts will be cached on first load. **UNSUPPORTED in DSpace 7.0**

7.1.3.25 Browse Index Configuration

The browse indexes for DSpace can be extensively configured. These configurations are used by Discovery[478]. This section of the configuration allows you to take control of the indexes you wish to browse, and how you wish to present the results. The configuration is broken into several parts: defining the indexes, defining the fields upon which users can sort results, defining truncation for potentially long fields (e.g. authors), setting cross-links between different browse contexts (e.g. from an author's name to a complete list of their items), how many recent submissions to display, and configuration for item mapping browse.

Property:	`webui.browse.index.<n>`

478 https://wiki.lyrasis.org/display/DSDOC5x/Discovery

Example Value:	`webui.browse.index.1 = dateissued:item:dateissued` `webui.browse.index.2 = author:metadata:dc.contributor.*,dc.creator:text`
Informational Note:	This is an example of how one "Defines the Indexes". See "Defining the Indexes(see page 607)" in the next sub-section.
Property:	`webui.itemlist.sort-option.<n>`
Example Value:	`webui.itemlist.sort-option.1 = title:dc.title:title`
Informational Note:	This is an example of how one "Defines the Sort Options". See "Defining Sort Options(see page 610)" in the following sub-section.

Defining the storage of the Browse Data

> Optionally, you may configure a custom implementation use for the Browse DAOs both for read operations (create/update operations are handled by Event Consumers). However, as of DSpace 6, DSpace only includes one out-of-the-box option:
>
> - SOLR Browse Engine (SOLR DAOs), default since DSpace 4.0 - This enables Apache Solr to be utilized as a backend for all browsing of DSpace. This option requires that you have Discovery(see page 386) (Solr search/browse engine) enabled in your DSpace.

Property:	`browseDAO.class`
Example Value:	browseDAO.class = org.dspace.browse.SolrBrowseDAO
Informational Note:	This property configures the Java class that is used for READ operations by the Browse System. **You need to have** Discovery(see page 386) **enabled (this is the default since DSpace 4.0) to use the Solr Browse DAOs**

Defining the Indexes

> If you make changes in this section be sure to update your SOLR indexes running the Discovery Maintenance Script, see Discovery(see page 386)

DSpace comes with four default indexes pre-defined: author, title, date issued, and subjects. Users may also define additional indexes or re-configure the current indexes for different levels of specificity. For example, the default entries that appear in the dspace.cfg as default installation:

```
webui.browse.index.1 = dateissued:item:dateissued
webui.browse.index.2 = author:metadata:dc.contributor.*,dc.creator:text
webui.browse.index.3 = title:item:title
webui.browse.index.4 = subject:metadata:dc.subject.*:text
#webui.browse.index.5 = dateaccessioned:item:dateaccessioned
```

There are two types of indexes which are provided in this default integration:

- "`item`" indexes which have a format of `webui.browse.index.<n> = <index-name> : item : <sort-type> : (asc | desc)`
- "`metadata`" indexes which have a format of `webui.browse.index.<n> = <index-name> : metadata : <comma-separated-list-of-metadata-fields> : (date | text) : (asc | dec) : <sort-type>`

Please notice that the punctuation is paramount in typing this property key in the `dspace.cfg` file. The following table explains each element:

Element	Definition and Options (if available)
`webui.browse.index.<n>`	*n* is the index number. The index numbers must start from 1 and increment continuously by 1 thereafter. Deviation from this will cause an error during install or a configuration update. So anytime you add a new browse index, remember to increase the number. (Commented out index numbers may be used over again).
`<index-name>`	The name by which the index will be identified. In order for the DSpace UI to display human-friendly description for this index, you'll need to update the UI's language packs (e.g. `src/assets/i18n/en.json5`) to include a key using this index name, for example: • `browse.metadata.<index-name> = "MyField",` • `browse.metadata.<index-name>.breadcrumbs = "Browse by MyField",`

Element	Definition and Options (if available)
(metadata|item)	Only two options are available: "metadata" or "item" • "metadata" indexes allow you to index all items based on one or more metadata fields. The list of fields should be provided as part of the "metadata" configuration. Only items which have values for these fields will appear in this index (e.g. if you have a "metadata" index for "dc.subject.*", an item will not appear in that browse/search if it doesn't have a "dc.subject.*" value). The browse index will have to parts: first it lists all values of the specified metadata fields. If the user select one of these values the index lists all items in which the specified metadata field is assigned with the selected value. • Note: If you set a <sort-type> to be used, this sort type is not used on the values of the metadata fields but on the order of the items when listing all items that have a specific value of the metadata field. • "item" indexes provide you with a browseable list of ALL items in the site, sorted by a particular metadata field. The field this index is sorted by is referenced by <sort-option-name> (which should refer to a corresponding "webui.itemlist.sort-option.<n>" setting... see Defining Sort Options(see page 610) below for more information)
<schema-prefix>	(Only for "metadata" indexes) The schema used for the field to be index. The default is dc (for Dublin Core).
<element>	(Only for "metadata" indexes) The schema element. In Dublin Core, for example, the author element is referred to as "Contributor". The user should consult the default Dublin Core Metadata Registry table in Appendix A.
<qualifier>	(Only for "metadata" indexes) This is the qualifier to the <element> component. The user has two choices: an asterisk "**" or a proper qualifier of the element. The asterisk is a wildcard and causes DSpace to index all types of the schema element. For example, if you have the element "contributor" and the qualifier "**" then you would index all contributor data regardless of the qualifier. Another example, you have the element "subject" and the qualifier "lcsh" would cause the indexing of only those fields that have the qualifier "lcsh". (This means you would only index Library of Congress Subject Headings and not all data elements that are subjects.

Element	Definition and Options (if available)
<sort-type>	(Optional, should be set for "item" indexes) This refers to the sort type / data type of the field: • date the index type will be treated as a date object and sorted as such • text the index type will be treated as plain text and sorted as such • (any other value refers to a custom <sort-type> which should be defined in a corresponding webui.itemlist.sort-option.<n> setting. See Defining Sort Options(see page 610) below for more information.)
<sort-order>	(Optional) The default sort order. Choose asc (ascending) or desc (descending). Ascending is the default value, but descending may be useful for date-based indexes (e.g. to display most recent submissions first)

Defining Sort Options

> If you make changes in this section be sure to update your SOLR indexes running the Discovery Maintenance Script, see Discovery(see page 386)

Sort options/types will be available when browsing a list of items (either on "item" index type above or after selecting a specific value for "metadata" indexes). You can define an arbitrary number of fields to sort on. For example, the default entries that appear in the *dspace.cfg* as default installation:

```
webui.itemlist.sort-option.1 = title:dc.title:title
webui.itemlist.sort-option.2 = dateissued:dc.date.issued:date
webui.itemlist.sort-option.3 = dateaccessioned:dc.date.accessioned:date
```

The format of each entry is web.browse.sort-option.<n> = <sort-type-name>:<schema-prefix>.<element>.<qualifier>:<datatype>. Please notice the punctuation used between the different elements. The following table explains the each element:

Element	Definition and Options (if available)
webui.itemlist.sort-option.<n>	*n* is an arbitrary number you choose.

Element	Definition and Options (if available)
<sort-type-name>	The name by which the sort option will be identified. This is the name by which it is referred in the "webui.browse.index" settings (see Defining the Indexes(see page 607)).
<schema-prefix>	The schema used for the field to be sorted on in the index. The default is dc (for Dublin Core).
<element>	The schema element. In Dublin Core, for example, the author element is referred to as "Contributor". The user should consult the default Dublin Core Metadata Registry table in Appendix A.
<qualifier>	This is the qualifier to the <element> component. The user has two choices: an asterisk "*" or a proper qualifier of the element.
<datatype>	This refers to the datatype of the field: `date` the sort field will be treated as a date object `text` the sort field will be treated as plain text. `title` the sort field will be treated like a title, which will include a link to the item page

Other Browse Options

We set other browse values in the following section.

Property:	`webui.browse.metadata.show-freq.< n >`
Example Value:	`webui.browse.metadata.show-freq.1 = false`
Informational Note:	This enable/disable the show of frequencies (count) in metadata browse <n> refers to the browse configuration. As default frequencies are shown for all metadata browse
Property:	`plugin.named.org.dspace.sort.OrderFormatDelegate`
Example Value:	`plugin.named.org.dspace.sort.OrderFormatDelegate = \` `org.dspace.sort.OrderFormatTitleMarc21=title`

Informational Note:	This sets the option for how the indexes are sorted. All sort normalizations are carried out by the OrderFormatDelegate. The plugin manager can be used to specify your own delegates for each datatype. The default datatypes (and delegates) are: `author = org.dspace.sort.OrderFormatAuthor` `title = org.dspace.sort.OrderFormatTitle` `text = org.dspace.sort.OrderFormatText` If you redefine a default datatype here, the configuration will be used in preferences to the default. However, if you do not explicitly redefine a datatype, then the default will still be used in addition to the datatypes you do specify. As of DSpace release 1.5.2, the multi-lingual MARC21 title ordering is configured as default, as shown in the example above. To use the previous title ordering (before release 1.5.2), comment out the configuration in your *dspace.cfg* file.

Browse Index Authority Control Configuration

Property:	`webui.browse.index.<n>`
Example Value:	`webui.browse.index.5 = lcAuthor:metadataAuthority:dc.contributor.author:authority`
Informational Note:	

Tag cloud

Apart from the single (type=metadata) and full (type=item) browse pages, tag cloud is a new way to display the unique values of a metadata field.

To enable “tag cloud” browsing for a specific index you need to declare it in the dspace.cfg configuration file using the following option:

Property:	webui.browse.index.tagcloud.<*n*>
Example Value:	webui.browse.index.tagcloud.1 = true

Informational Note:	Enable/Disable tag cloud in browsing for a specific index. 'n' is the index number of the specific index which needs to be of type 'metadata'. Possible values: true, false Default value is false. If no option exists for a specific index, it is assumed to be false. You do not have to re-index discovery when you change this configuration **UNSUPPORTED in DSpace 7.0**

Tag cloud configuration

The appearance configuration for the tag cloud is located in the Discovery xml configuration file (*dspace/config/spring/api/discovery.xml*). Without configuring the appearance, the default one will be applied to the tag cloud

In this file, there must be a bean named "*browseTagCloudConfiguration*" of class "*org.dspace.discovery.configuration.TagCloudConfiguration*". This bean can have any of the following properties. If some is missing, the default value will be applied.

displayScore	Should display the score of each tag next to it? Default: false
shouldCenter	Should display the tag as center aligned in the page or left aligned? Possible values: true | false. Default: true
totalTags	How many tags will be shown. Value -1 means all of them. Default: -1
cloudCase	The letter case of the tags. Possible values: Case.LOWER | Case.UPPER | Case.CAPITALIZATION | Case.PRESERVE_CASE | Case.CASE_SENSITIVE Default: Case.PRESERVE_CASE
randomColors	If the 3 css classes of the tag cloud should be independent of score (random=yes) or based on the score. Possible values: true | false . Default: true
fontFrom	The font size (in em) for the tag with the lowest score. Possible values: any decimal. Default: 1.1

fontTo	The font size (in em) for the tag with the lowest score. Possible values: any decimal. Default: 3.2
cuttingLevel	The score that tags with lower than that will not appear in the rag cloud. Possible values: any integer from 1 to infinity. Default: 0
ordering	The ordering of the tags (based either on the name or the score of the tag) Possible values: Tag.NameComparatorAsc | Tag.NameComparatorDesc | Tag.ScoreComparatorAsc | Tag.ScoreComparatorDesc Default: Tag.GreekNameComparatorAsc

When tagCloud is rendered there are some CSS classes that you can change in order to change the tagcloud appearance.

Class	Note
tagcloud	General class for the whole tagcloud
tagcloud_1	Specific tag class for tag of type 1 (based on score)
tagcloud_2	Specific tag class for tag of type 2 (based on score)
tagcloud_3	Specific tag class for tag of type 3 (based on score)

7.1.3.26 Links to Other Browse Contexts

We can define which fields link to other browse listings. This is useful, for example, to link an author's name to a list of just that author's items. The effect this has is to create links to browse views for the item clicked on. If it is a "single" type, it will link to a view of all the items which share that metadata element in common (i.e. all the papers by a single author). If it is a "full" type, it will link to a view of the standard full browse page, starting with the value of the link clicked on.

Property:	`webui.browse.link.<n>`
Example Value:	`webui.browse.link.1 = author:dc.contributor.*`
Informational Note:	This is used to configure which fields should link to other browse listings. This should be associated with the name of one of the browse indexes (`webui.browse.index.n`) with a metadata field listed in `webui.itemlist.columns` above. If this condition is not fulfilled, cross-linking will not work. Note also that crosslinking only works for metadata fields not tagged as `title` in `webui.itemlist.columns`.

The format of the property key is *webui.browse.link.<n> = <index name>:<display column metadata>* Please notice the punctuation used between the elements.

Element	Definition and Options (if available)
`webui.browse.link.` *n*	{{*n* is an arbitrary number you choose
`<index name>`	This need to match your entry for the index name from *webui.browse.index* property key.
`<display column metadata>`	Use the DC element (and qualifier)

Examples of some browse links used in a real DSpace installation instance:

`webui.browse.link.1 = author:dc.contributor.*`

Creates a link for all types of contributors (authors, editors, illustrators, others, etc.)

`webui.browse.link.2 = subject:dc.subject.lcsh`

Creates a link to subjects that are Library of Congress only. In this case, you have a browse index that contains only LC Subject Headings

`webui.browse.link.3 = series:dc.relation.ispartofseries`

Creates a link for the browse index "Series". Please note this is again, a customized browse index and not part of the DSpace distributed release.

7.1.3.27 Submission License Substitution Variables

Property:	`plugin.named.org.dspace.content.license.` `LicenseArgumentFormatter` (property key broken up for display purposes only)
Example Value:	`plugin.named.org.dspace.content.license.LicenseArgumentFormatter = \` `org.dspace.content.license.SimpleDSpaceObjectLicenseFormatter = collection, \` `org.dspace.content.license.SimpleDSpaceObjectLicenseFormatter = item, \` `org.dspace.content.license.SimpleDSpaceObjectLicenseFormatter = eperson`

Informational Note:	It is possible include contextual information in the submission license using substitution variables. The text substitution is driven by a plugin implementation.

7.1.3.28 Syndication Feed (RSS) Settings

> ⚠ **UNSUPPORTED in 7.0.** Will be added in a later release

This will enable syndication feeds, links display on community and collection home pages.

Property:	`webui.feed.enable`
Example Value:	`webui.feed.enable = true`
Informational Note:	By default, RSS feeds are set to true (on) . Change key to "false" to disable.
Property:	`webui.feed.items`
Example Value:	`webui.feed.items = 4`
Informational Note:	Defines the number of DSpace items per feed (the most recent submissions)
Property:	`webui.feed.cache.size`
Example Value:	`webui.feed.cache.size = 100`
Informational Note:	Defines the maximum number of feeds in memory cache. Value of "`0`" will disable caching.
Property:	`webui.feed.cache.age`
Example Value:	`webui.feed.cache.age = 48`

Informational Note:	Defines the number of hours to keep cached feeds before checking currency. The value of "*0*" will force a check with each request.
Property:	`webui.feed.formats`
Example Value:	`webui.feed.formats = rss_1.0,rss_2.0,atom_1.0`
Informational Note:	Defines which syndication formats to offer. You can use more than one; use a comma-separated list. The following list are the available values: rss_0.90, rss_0.91, rss_0.92, rss_0.93, rss_0.94, rss_1.0, rss_2.0, atom_1.0.
Property:	`webui.feed.localresolve`
Example Value:	`webui.feed.localresolve = false`
Informational Note:	By default, (set to false), URLs returned by the feed will point at the global handle resolver (e.g. http://hdl.handle.net/123456789/1). If set to ***true*** the local server URLs are used (e.g. http://myserver.myorg/handle/123456789/1).
Property:	`webui.feed.item.title`
Example Value:	`webui.feed.item.title = dc.title`
Informational Note:	This property customizes each single-value field displayed in the feed information for each item. Each of the fields takes a **single** metadata field. The form of the key is <scheme prefix>.<element>.<qualifier> In place of the qualifier, one may leave it blank to exclude any qualifiers or use the wildcard "*" to include all qualifiers for a particular element.
Property:	`webui.feed.item.date`
Example Value:	`webui.feed.item.date = dc.date.issued`

Informational Note:	This property customizes each single-value field displayed in the feed information for each item. Each of the fields takes a **single** metadata field. The form of the key is <scheme prefix>.<element>.<qualifier> In place of the qualifier, one may leave it blank to exclude any qualifiers or use the wildcard "*" to include all qualifiers for a particular element.
Property:	`webui.feed.item.description`
Example Value:	`webui.feed.item.description = dc.title, dc.contributor.author, \` `dc.contributor.editor, dc.description.abstract, \` `dc.description`
Informational Note:	One can customize the metadata fields to show in the feed for each item's description. Elements are displayed in the order they are specified in *dspace.cfg*.Like other property keys, the format of this property key is: *webui.feed.item.description = <scheme prefix>.<element>.<qualifier>*. In place of the qualifier, one may leave it blank to exclude any qualifiers or use the wildcard "*" to include all qualifiers for a particular element.
Property:	`webui.feed.item.author`
Example Value:	`webui.feed.item.author = dc.contributor.author`
Informational Note:	The name of field to use for authors (Atom only); repeatable.
Property:	`webui.feed.logo.url`
Example Value:	`webui.feed.logo.url = ${dspace.url}/themes/mysite/images/mysite-logo.png`
Informational Note:	Customize the image icon included with the site-wide feeds. This must be an absolute URL.
Property:	`webui.feed.item.dc.creator`

Example Value:	`webui.feed.item.dc.creator = dc.contributor.author`
Informational Note:	This optional property adds *structured* DC elements as XML elements to the feed description. They are not the same thing as, for example, *webui.feed.item.description*. Useful when a program or stylesheet will be transforming a feed and wants separate author, description, date, etc.
Property:	`webui.feed.item.dc.date`
Example Value:	`webui.feed.item.dc.date = dc.date.issued`
Informational Note:	This optional property adds *structured* DC elements as XML elements to the feed description. They are not the same thing as, for example, *webui.feed.item.description*. Useful when a program or stylesheet will be transforming a feed and wants separate author, description, date, etc.
Property:	`webui.feed.item.dc.description`
Example Value:	`webui.feed.item.dc.description = dc.description.abstract`
Informational Note:	This optional property adds *structured* DC elements as XML elements to the feed description. They are not the same thing as, for example, *webui.feed.item.description*. Useful when a program or stylesheet will be transforming a feed and wants separate author, description, date, etc.
Property:	`webui.feed.podcast.collections`
Example Value:	`webui.feed.podcast.collections = 1811/45183,1811/47223`
Informational Note:	This optional property enables Podcast Support on the RSS feed for the specified collection handles. The podcast is iTunes compatible and will expose the bitstreams in the items for viewing and download by the podcast reader. Multiple values are separated by commas. For more on using/enabling Media RSS Feeds to share content via iTunesU, see: Enable Media RSS Feeds[479]
Property:	`webui.feed.podcast.communities`

479 https://wiki.lyrasis.org/display/DSPACE/Enable+Media+RSS+Feeds

Example Value:	`webui.feed.podcast.communities = 1811/47223`
Informational Note:	This optional property enables Podcast Support on the RSS feed for the specified community handles. The podcast is iTunes compatible and will expose the bitstreams in the items for viewing and download by the podcast reader. Multiple values are separated by commas. For more on using/enabling Media RSS Feeds to share content via iTunesU, see: Enable Media RSS Feeds[480]
Property:	`webui.feed.podcast.mimetypes`
Example Value:	`webui.feed.podcast.mimetypes = audio/x-mpeg,application/pdf`
Informational Note:	This optional property for Podcast Support, allows you to choose which MIME types of bitstreams are to be enclosed in the podcast feed. Multiple values are separated by commas. For more on using/enabling Media RSS Feeds to share content via iTunesU, see: Enable Media RSS Feeds[481]
Property:	`webui.feed.podcast.sourceuri`
Example Value:	`webui.feed.podcast.sourceuri = dc.source.uri`
Informational Note:	This optional property for the Podcast Support will allow you to use a value for a metadata field as a replacement for actual bitstreams to be enclosed in the RSS feed. A use case for specifying the external sourceuri would be if you have a non-DSpace media streaming server that has a copy of your media file that you would prefer to have the media streamed from. For more on using/enabling Media RSS Feeds to share content via iTunesU, see: Enable Media RSS Feeds[482]

7.1.3.29 OpenSearch Support

OpenSearch is a small set of conventions and documents for describing and using "search engines", meaning any service that returns a set of results for a query. See extensive description in the *Business Layer section* of the documentation.

Please note that for result data formatting, OpenSearch uses Syndication Feed Settings (RSS). So, even if Syndication Feeds **are not** enable, they **must** be configured to enable OpenSearch. OpenSearch uses all the

480 https://wiki.lyrasis.org/display/DSPACE/Enable+Media+RSS+Feeds
481 https://wiki.lyrasis.org/display/DSPACE/Enable+Media+RSS+Feeds
482 https://wiki.lyrasis.org/display/DSPACE/Enable+Media+RSS+Feeds

configuration properties for DSpace RSS to determine the mapping of metadata fields to feed fields. Note that a new field for authors has been added (used in Atom format only).

Property:	`websvc.opensearch.enable`
Example Value:	`websvc.opensearch.enable = false`
Informational Note:	Whether or not OpenSearch is enabled. By default, the feature is disabled. Change the property key to "true" to enable.
Property:	`websvc.opensearch.uicontext`
Example Value:	`websvc.opensearch.uicontext = simple-search`
Informational Note:	Context for HTML request URLs. Change only for non-standard servlet mapping.
Property:	`websvc.opensearch.autolink`
Example Value:	`websvc.opensearch.autolink = true`
Informational Note:	Present autodiscovery link in every page head.
Property:	`websvc.opensearch.validity`
Example Value:	`websvc.opensearch.validity = 48`
Informational Note:	Number of hours to retain results before recalculating. This applies to the Manakin interface only.
Property:	`websvc.opensearch.shortname`
Example Value:	`websvc.opensearch.shortname = DSpace`
Informational Note:	A short name used in browsers for search service. It should be sixteen (16) or fewer characters.

Property:	`websvc.opensearch.longname`
Example Value:	`websvc.opensearch.longname = ${dspace.name}`
Informational Note:	A longer name up to 48 characters.
Property:	`websvc.opensearch.description`
Example Value:	`websvc.opensearch.description = ${dspace.name} DSpace repository`
Informational Note:	`Brief service description`
Property:	`websvc.opensearch.faviconurl`
Example Value:	_websvc.opensearch.faviconurl = http://www.dspace.org/images/favicon.ico_
Informational Note:	Location of favicon for service, if any. They must by 16 x 16 pixels. You can provide your own local favicon instead of the default.
Property:	`websvc.opensearch.samplequery`
Example Value:	`websvc.opensearch.samplequery = photosynthesis`
Informational Note:	Sample query. This should return results. You can replace the sample query with search terms that should actually yield results in your repository.
Property:	`websvc.opensearch.tags`
Example Value:	`websc.opensearch.tags = IR DSpace`
Informational Note:	Tags used to describe search service.

Property:	`websvc.opensearch.formats`
Example Value:	`websvc.opensearch.formats = html,atom,rss`
Informational Note:	Result formats offered. Use one or more comma-separated from the list: html, atom, rss. Please note that html is required for auto discovery in browsers to function, and must be the first in the list if present.

7.1.3.30 Content Inline Disposition Threshold

The following configuration is used to change the disposition behavior of the browser. That is, when the browser will attempt to open the file or download it to the user-specified location. For example, the default size is 8MB. When an item being viewed is larger than 8MB, the browser will download the file to the desktop (or wherever you have it set to download) and the user will have to open it manually.

Property:	`webui.content_disposition_threshold`
Example value:	`webui.content_disposition_threshold = 8388608`
Informational Note:	The default value is set to 8MB. This property key applies to the REST API.

Other values are possible:
4 MB = 41943048 MB = 838860816 MB = 16777216

7.1.3.31 Multi-file HTML Document/Site Settings

The setting is used to configure the "depth" of request for html documents bearing the same name.

Property:	`webui.html.max-depth-guess`
Example Value:	`webui.html.max-depth-guess = 3`

Informational Note:	When serving up composite HTML items in the UI, how deep can the request be for us to serve up a file with the same name? For example, if one receives a request for "*foo/bar/index.html*" and one has a bitstream called just "*index.html*", DSpace will serve up the former bitstream (*foo/bar/index.html*) for the request if *webui.html.max-depth-guess* is 2 or greater. If *webui.html.max-depth-guess* is 1 or less, then DSpace would not serve that bitstream, as the depth of the file is greater. If webui.html.max-depth-guess is zero, the request filename and path must always exactly match the bitstream name. The default is set to 3. **UNSUPPORTED IN DSpace 7.0**

7.1.3.32 Sitemap Settings

To aid web crawlers index the content within your repository, you can make use of sitemaps. For best SEO, Sitemaps are enabled by default and update automatically (see cron setting).

Property:	`sitemap.dir`
Example Value:	`sitemap.dir = ${dspace.dir}/sitemaps`
Informational Note:	The directory where the generate sitemaps are stored.
Property:	`sitemap.engineurls`
Example Value:	sitemap.engineurls = http://www.google.com/webmasters/sitemaps/ping?sitemap=[483]
Informational Note:	Comma-separated list of search engine URLs to "ping" when a new Sitemap has been created. Include everything except the Sitemap UL itself (which will be URL-encoded and appended to form the actual URL "pinged").Add the following to the above parameter if you have an application ID with Yahoo: http://search.yahooapis.com/SiteExplorererService/V1/updateNotification?appid=REPLACE_ME?url=_ . *(Replace the component _REPLACE_ME* with your application ID). There is no known "ping" URL for MSN/Live search.
Property:	`sitemap.cron`

483 http://www.google.com/webmasters/sitemaps/ping?sitemap=_

Example Value:	`sitemap.cron = 0 15 1 * * ?`
Informational Note:	The DSpace sitemaps are regenerated on a regular basis based on the Cron syntax provided in this configuration. By default, sitemaps are updated daily at 1:15am local time. Cron syntax is defined at https://www.quartz-scheduler.org/api/2.3.0/org/quartz/CronTrigger.html Remove (comment out) this config to disable the sitemap scheduler. Sitemap scheduler can also be disabled by setting to "-" (single dash) in local.cfg.

7.1.3.33 Authority Control Settings

Two features fall under the header of Authority Control: Choice Management and Authority Control of Item ("DC") metadata values. Authority control is a fully optional feature in DSpace 1.6. Implemented out of the box are the Library of Congress Names service, and the Sherpa Romeo authority plugin.

For an in-depth description of this feature, please consult: Authority Control of Metadata Values[484]

Property:	`plugin.named.org.dspace.content.authority.ChoiceAuthority`
Example Value:	`plugin.named.org.dspace.content.authority.ChoiceAuthority = \` `org.dspace.content.authority.SampleAuthority = Sample, \` `org.dspace.content.authority.SHERPARoMEOPublisher = SRPublisher, \` `org.dspace.content.authority.SHERPARoMEOJournalTitle = SRJournalTitle, \` `org.dspace.content.authority.SolrAuthority = SolrAuthorAuthority`
Informational Note:	List of all enabled authority control plugins
Property:	`plugin.selfnamed.org.dspace.content.authority.ChoiceAuthority`
Example Value:	`plugin.selfnamed.org.dspace.content.authority.ChoiceAuthority = \` `org.dspace.content.authority.DCInputAuthority`
Property:	`lcname.url`

484 https://wiki.lyrasis.org/display/DSPACE/Authority+Control+of+Metadata+Values

Example Value:	`lcname.url = http://alcme.oclc.org/srw/search/lcnaf_`
Informational Note:	Location (URL) of the Library of Congress Name Service
Property:	`sherpa.romeo.url / sherpa.romeo.apikey`
Informational Note:	Please refers to the Sherpa/RoMEO Publishers Policy Database Integration section for details about such properties. See Configuring the Sherpa/RoMEO Publishers Policy Database Integration(see page 0)
Property:	`orcid.api.url`
Example Value:	`orcid.api.url = https://pub.orcid.org/v3.0`
Informational Note:	Location (URL) of the ORCID v3 Public API
Property:	`authority.minconfidence`
Example Value:	`authority.minconfidence = ambiguous`
Informational Note:	This sets the default lowest confidence level at which a metadata value is included in an authority-controlled browse (and search) index. It is a symbolic keyword, one of the following values (listed in descending order): accepted, uncertain, ambiguous, notfound, failed, rejected, novalue, unset. See `org.dspace.content.authority.Choices` source for descriptions.

7.1.3.34 Configuring Multilingual Support

See Multilingual Support(see page 407) *for more details/examples.*

Setting the Default Language for the Application

Property:	`default.locale`
Example Value:	`default.locale = en`
Informational Note:	The default language for the application is set with this property key. This is a locale according to i18n and might consist of country, country_language or country_language_variant. If no default locale is defined, then the server default locale will be used. The format of a local specifier is described here: http://java.sun.com/j2se/1.4.2/docs/api/java/util/Locale.html

Supporting More Than One Language

Changes in dspace.cfg

Property:	`webui.supported.locales`
Example Value:	`webui.supported.locales = en, de`
`or perhaps`	`webui.supported.locales = en, en_ca, de`
Informational Note:	All the locales that are supported by this instance of DSpace. Comma separated list. **UNSUPPORTED IN DSpace 7.0.** *However, the DSpace 7 UI has a similar "languages" setting in environment.*.ts*

The table above, if needed and is used will result in:

- a language switch in the default header
- the user will be enabled to choose his/her preferred language, this will be part of his/her profile
- wording of emails
    - mails to registered users, e.g. alerting service will use the preferred language of the user
    - mails to unregistered users, e.g. suggest an item will use the language of the session
- according to the language selected for the session, using dspace-admin Edit News will edit the news file of the language according to session

Related Files

If you set webui.supported.locales make sure that all the related additional files for each language are available. *LOCALE* should correspond to the locale set in *webui.supported.locales*, e. g.: for webui.supported.locales = en, de, fr, there should be:

- `[dspace-source]/dspace/modules/server/src/main/resources/Messages.properties`
- `[dspace-source]/dspace/modules/server/src/main/resources/Messages_en.properties`
- `[dspace-source]/dspace/modules/server/src/main/resources/Messages_de.properties`
- `[dspace-source]/dspace/modules/server/src/main/resources/Messages_fr.properties`

Files to be localized:

- `[dspace-source]/dspace/modules/server/src/main/resources/Messages_LOCALE.properties`
- `[dspace-source]/dspace/config/submission-forms_LOCALE.xml`
- `[dspace-source]/dspace/config/default_LOCALE.license - should be pure ASCII`
- `[dspace-source]/dspace/config/emails/change_password_LOCALE`
- `[dspace-source]/dspace/config/emails/feedback_LOCALE`
- `[dspace-source]/dspace/config/emails/internal_error_LOCALE`
- `[dspace-source]/dspace/config/emails/register_LOCALE`
- `[dspace-source]/dspace/config/emails/submit_archive_LOCALE`
- `[dspace-source]/dspace/config/emails/submit_reject_LOCALE`
- `[dspace-source]/dspace/config/emails/submit_task_LOCALE`
- `[dspace-source]/dspace/config/emails/subscription_LOCALE`
- `[dspace-source]/dspace/config/emails/suggest_LOCALE`

7.1.3.35 Upload File Settings

Property:	`upload.temp.dir`
Example Value:	`upload.temp.dir = ${dspace.dir}/upload`
Informational Note:	This property sets where DSpace temporarily stores uploaded files.

7.1.3.36 SFX Server (OpenURL)

SFX Server is an OpenURL Resolver.

Property:	`sfx.server.url`
Example Value:	`sfx.server.url = http://sfx.myu.edu:8888/sfx?`
	`sfx.server.url = http://worldcatlibraries.org/registry/gateway?`

Informational Note:	SFX query is appended to this URL. If this property is commented out or omitted, SFX support is switched off.

All the parameters mapping are defined in `[dspace]/config/sfx.xml` file. The program will check the parameters in `sfx.xml` and retrieve the correct metadata of the item. It will then parse the string to your resolver.

For the following example, the program will search the first query-pair which is DOI of the item. If there is a DOI for that item, your retrieval results will be, for example:
`http://researchspace.auckland.ac.nz/handle/2292/5763`

Example. For setting DOI in `sfx.xml`

```
<query-pairs>
    <field>
        <querystring>rft_id=info:doi/</querystring>
        <dc-schema>dc</dc-schema>
        <dc-element>identifier</dc-element>
        <dc-qualifier>doi</dc-qualifier>
    </field>
</query-pairs>
```

If there is no DOI for that item, it will search next query-pair based on the `[dspace]/config/sfx.xml` and then so on.

Example of using ISSN, volume, issue for item without DOI
`[http://researchspace.auckland.ac.nz/handle/2292/4947]`

For parameter passing to the `<querystring>`

```
<querystring>rft_id=info:doi/</querystring>
```

Please refer to these:
`[http://ocoins.info/cobgbook.html]`
`[http://ocoins.info/cobg.html]`

Program assume won't get empty string for the item, as there will at least author, title for the item to pass to the resolver.

For contributor author, program maintains original DSpace SFX function of extracting author's first and last name.

```
<field>
    <querystring>rft.aulast=</querystring>
    <dc-schema>dc</dc-schema>
    <dc-element>contributor</dc-element>
    <dc-qualifier>author</dc-qualifier>
</field>
<field>
    <querystring>rft.aufirst=</querystring>
    <dc-schema>dc</dc-schema>
    <dc-element>contributor</dc-element>
    <dc-qualifier>author</dc-qualifier>
</field>
```

7.1.3.37 Controlled Vocabulary Settings

DSpace now supports controlled vocabularies to confine the set of keywords that users can use while describing items.

The need for a limited set of keywords is important since it eliminates the ambiguity of a free description system, consequently simplifying the task of finding specific items of information.

The controlled vocabulary add-on allows the user to choose from a defined set of keywords organized in an tree (taxonomy) and then use these keywords to describe items while they are being submitted.

We have also developed a small search engine that displays the classification tree (or taxonomy) allowing the user to select the branches that best describe the information that he/she seeks.

The taxonomies are described in XML following this (very simple) structure:

```
<node id="acmccs98" label="ACMCCS98">
    <isComposedBy>
        <node id="A." label="General Literature">
            <isComposedBy>
                <node id="A.0" label="GENERAL"/>
                <node id="A.1" label="INTRODUCTORY AND SURVEY"/>
            </isComposedBy>
        </node>
    </isComposedBy>
</node>
```

You are free to use any application you want to create your controlled vocabularies. A simple text editor should be enough for small projects. Bigger projects will require more complex tools. You may use Protegé to create your taxonomies, save them as OWL and then use a XML Stylesheet (XSLT) to transform your documents to the appropriate format. Future enhancements to this add-on should make it compatible with standard schemas such as OWL or RDF.

New vocabularies should be placed in `[dspace]/config/controlled-vocabularies/` and must be according to the structure described. A validation XML Schema (named `controlledvocabulary.xsd`) is also available in that directory.

Vocabularies need to be associated with the correspondent DC metadata fields. Edit the file `[dspace]/config/input-forms.xml` and place a *"vocabulary"* tag under the *"field"* element that you want to control. Set value of the *"vocabulary"* element to the name of the file that contains the vocabulary, leaving out the extension (the add-on will only load files with extension "*.xml"). For example:

```
<field>
    <dc-schema>dc</dc-schema>
    <dc-element>subject</dc-element>
    <dc-qualifier></dc-qualifier>
    <!-- An input-type of twobox MUST be marked as repeatable -->
    <repeatable>true</repeatable>
    <label>Subject Keywords</label>
    <input-type>twobox</input-type>
    <hint> Enter appropriate subject keywords or phrases below. </hint>
    <required></required>
    <vocabulary [closed="false"]>nsi</vocabulary>
</field>
```

The vocabulary element has an optional boolean attribute **closed** that can be used to force input only with the JavaScript of controlled-vocabulary add-on. The default behavior (i.e. without this attribute) is as set **closed="false"**. This allow the user also to enter the value in free way.

The following vocabularies are currently available by default:

- **nsi** - *nsi.xml* - The Norwegian Science Index
- **srsc** - *srsc.xml* - Swedish Research Subject Categories

7.1.4 Optional or Advanced Configuration Settings

The following section explains how to configure either optional features or advanced features that are not necessary to make DSpace "out-of-the-box"

7.1.4.1 The Metadata Format and Bitstream Format Registries

The *[dspace]/config/registries* directory contains three XML files. These are used to load the *initial* contents of the Dublin Core Metadata registry and Bitstream Format registry and SWORD metadata registry. After the initial loading (performed by *ant fresh_install* above), the registries reside in the database; the XML files are not updated.

In order to change the registries, you may adjust the XML files before the first installation of DSpace. On an already running instance it is recommended to change bitstream registries via DSpace admin UI, but the metadata registries can be loaded again at any time from the XML files without difficulty. The changes made via admin UI are not reflected in the XML files.

Metadata Format Registries

The default metadata schema is Dublin Core, so DSpace is distributed with a default Dublin Core Metadata Registry. Currently, the system requires that every item have a Dublin Core record.

There is a set of Dublin Core Elements, which is used by the system and should not be removed or moved to another schema. See Appendix: Default Dublin Core Metadata registry.

Note: altering a Metadata Registry has no effect on corresponding parts, e.g. item submission interface, item display, item import and vice versa. Every metadata element used in submission interface or item import must be registered before using it.

Note also that deleting a metadata element will delete all its corresponding values.

If you wish to add more metadata elements, you can do this in one of two ways. Via the DSpace admin UI you may define new metadata elements in the different available schemas. But you may also modify the XML file (or provide an additional one), and re-import the data as follows:

```
[dspace]/bin/dspace registry-loader -metadata [xml file]
```

The XML file should be structured as follows:

```
<dspace-dc-types>
    <dc-type>
        <schema>dc</schema>
        <element>contributor</element>
        <qualifier>advisor</qualifier>
        <scope_note>Use primarily for thesis advisor.</scope_note>
    </dc-type>
</dspace-dc-types>
```

The set of metadata registry files which is read by the MetadataImporter tool is configured by the `metadata.registry.load` property in `dspace.cfg` or `local.cfg`. If you wish to use the importer to load a new metadata namespace from a new file, you will need to add the path to your new registry file as an additional value of this property before running the tool.

Bitstream Format Registry

The bitstream formats recognized by the system and levels of support are similarly stored in the bitstream format registry. This can also be edited at install-time via *[dspace]/config/registries/bitstream-formats.xml* or by the administration Web UI. The contents of the bitstream format registry are entirely up to you, though the system requires that the following two formats are present:

- *Unknown*
- *License*
  Deleting a format will cause any existing bitstreams of this format to be reverted to the unknown bitstream format.

7.1.4.2 Configuring Usage Instrumentation Plugins

A usage instrumentation plugin is configured as a Spring bean in the applicationContext.xml for each of the various user interface web applications. It will require the injection of an instance of EventService, which it will use to register itself on the UsageEvent bus. See the configuration file for examples.

More than one such plugin may be configured – each will receive all usage events.

If you wish to write your own, it must extend the abstract class org.dspace.usage.AbstractUsageEventListener.

The Passive Plugin

The Passive plugin is provided as the class org.dspace.usage.PassiveUsageEventListener. It absorbs events without effect, and serves as a simple example of how to write a UsageEvent listener.

The Tab File Logger Plugin

The Tab File Logger plugin is provided as the class org.dspace.usage.TabFileUsageEventListener. It writes event records to a file in tab-separated column format. If left unconfigured, it will write to `[DSpace]/log/usage-events.tsv`. To specify the file path, provide an absolute path, or a path relative to log.dir, as the value for usageEvent.tabFileLogger.file in dspace.cfg.

7.1.4.3 Behavior of the workflow system

DSpace contains a workflow system to review submissions as described in detail as part of the architecture of the business logic layer(see page 655) and in Configurable Workflow(see page 250). The file **[dspace]/config/modules/workflow.cfg** contains additional properties to configure details of the workflow system.

The property `workflow.reviewer.file-edit` controls whether files may be added/edited/removed during review (set to true) or whether files can be downloaded during review only.

\[dspace\]/config/modules/workflow.cfg

```
#Allow the reviewers to add/edit/remove files from the submission
#When changing this property you might want to alert submitters in the license that reviewers can alter
their files
workflow.reviewer.file-edit=false
```

The workflow system will send notifications on new Items waiting to be reviewed to all EPersons that may resolve those. Tasks can be taken to avoid that two EPersons work on the same task at the same time without knowing from each other. When a EPerson returns a task to the pool without resolving it (by accepting or rejecting the submission), another E-Mail is sent. In case you only want to be notified of completely new tasks entering a step of the workflow system, you may switch off notifications on tasks returned to the pool by setting workflow.notify.returend.tasks to false in config/modules/workflow.cfg as shown below:

\[dspace\]/config/modules/workflow.cfg

```
# Notify reviewers about tasks returned to the pool
workflow.notify.returned.tasks = false
```

By default notifications are sent for tasks returned to the pool.

7.1.4.4 Recognizing Web Spiders (Bots, Crawlers, etc.)

DSpace can often recognize that a given access request comes from a web spider that is indexing your repository. These accesses can be flagged for separate treatment (perhaps exclusion) in usage statistics. This requires patterns to match against incoming requests. These patterns exist in files that you will find in `config/spiders`.

In the `spiders` directory itself, you will find a number of files provided by iplists.com. These files contain network address patterns which have been discovered to identify a number of known indexing services and other spiders. You can add your own files here if you wish to exclude more addresses that you know of. You will need to include your files' names in the list configured in `config/modules/solr-statistics.cfg`. The `iplists.com-*.txt` files can be updated using a tool provided by DSpace. See SOLR Statistics(see page 338) for details.

In the `spiders` directory you will also find two subdirectories. `agents` contains files filled with regular expressions, one per line. An incoming request's `User-Agent` header is tested with each expression found in any of these files until an expression matches. If there is a match, the request is marked as being from a spider, otherwise

not. `domains` similarly contains files filled with regular expressions which are used to test the domain name from which the request comes. You may add your own files of regular expressions to either directory if you wish to test requests with patterns of your own devising.

7.1.5 Command-line Access to Configuration Properties

You can resolve a configuration property name to its value using the command `dspace dsprop -p some.property.name`. The output is undecorated and may be suitable for use in scripts.

The `dsprop command has these options:`

name	argument	meaning
--property -p	name	the name of the desired configuration property. This option is required.
--module -m	name	the name of the module in which the property is found. If omitted, the value of --property is the entire name. If used, the name will be composed as module.property. For example, "`-m dspace -p url`" will look up the value of `dspace.url`.
--raw -r		if used, this prevents the substitution of other property values into the value of the requested property. It is also useful to see all of the propery values when a specific property has an **array of values** (i.e. the configuration supports specifying multiple values). Otherwise, by default , dsprop may only return the first value in the array.
--help -h -?		Display help similar to this table.

7.2 DSpace Item State Definitions

Workspace item

An item that is under submission and active edit by an authorized user. The workspace item is visible only to the submitter and the system administrators. (Currently there is no simple way to find/browse such items other than with the direct item ID or to use the supervisor functionality). Using the supervisor functionality, a system admin can allow other authorized user to see/edit the item in the workspace state.

Expected use cases:

- Self deposit
- Collaboration over an in-progress submission for a small group of researchers. (This use case is implemented only with major limitations, using the supervision feature – concurrency, lack of delegation: supervision must be defined by the system administrators, etc.)

Workflow Item

An item that is under review for quality control and policy compliance. The workflow item is visible to the original submitter (currently only basic metadata are visible out-of-box in the mydspace summary list), users assigned to the specific workflow step where the item resides, and system administrators. (Currently there is no simple way to find/browse such items other than with the direct item ID or to use the abort workflow functionality).

Expected use cases:

- Quality control
- Improvements to the bibliographic record (metadata available in workflow can be different than those asked of the submitter)
- Check of policy / copyright

Withdrawn item

It is the removal of an Item from the archive. However, a withdrawn item is still available to Administrative users (and may optionally be restored to the archive at a later date). A withdrawn item disappears from DSpace (except from Administrative screens) and the item appears to be deleted.

Expected use cases:

- Staging area for item to be removed when copyright issues arise with publisher. If the copyright issue is confirmed, the item will be permanently deleted or kept in the withdrawn state for future reference.
- Logical deletion delegated to community/collection admin, where permanent deletion is reserved to system administrators
- Logical deletion, where permanent deletion is not an option for an organization
- Removal of an old version of an item, forcing redirect to a new up-to-date version of the item (this use case is not currently implemented out-of-box in DSpace, see)

Private item

> ⚠ Despite its name, a "private" item is not necessarily access restricted. It's simply **hidden** from all search/browse/OAI results, and is therefore only accessible via direct link (or bookmark). If you wish to access restrict the item so that it is no longer available to a certain group of users (or only available to Admins), you should edit the Item's Authorization Policies (via the Edit Item screens).

This state should only refer to the discoverable nature of the item. A private item will not be included in any system that aims to help users to find items. So it will not appear in:

- Browse
- Recent submission
- Search result
- OAI-PMH (at least for the ListRecords and ListIdentifiers verb; though the OAI-PMH specification is not clear about inconsistent implementation of the ListRecords and GetRecord verb)
- REST list and search methods

It should be accessible under the actual Authorization Policies of DSpace using direct URL or query method such as:

- Splash page access (i.e. /handle/<xxxxx>/<yyyyy>)
- OAI-PMH GetRecord verb
- REST direct access /rest/item/<item-id> or equivalent

Expected use cases:

- Provide a light rights awareness feature where discovery is not enabled for search and/or browse
- Hide “special items” such as repository presentations, guides or support materials
- Hide an old version of an Item in cases where real versioning is not appropriate or liked
- Hide specific types of item such as “Item used to record Journal record: Journal Title, ISSN, Publisher etc.” used as authority file for metadata (dc.relation.ispartof) of “normal item”

Archived/Published item

An item that is in a stable state, available in the repository under the defined Authorization Policies. Changes to these items are possible only for a restricted group of users (administrators) and should produce versioning according to the Institution's policy.

Embargoed Item(see page 115)

Are a special case of Archived/Published Item. The item has some time based access policy attached to it and/or the underlying bitstreams. Specifically, read permission for someone (EPerson Group) starting from a defined date. Typically embargo is applied to the bitstreams so that "fulltext" has initially very limited access (normally administrators or other "repository staff" groups) and only after a defined date will the fulltext become visible to all users (Anonymous group). This scenario is used to implement typical "embargo requirements" from publishers -- see Delayed Open Access[485].

If the metadata of the item should be visible only to a specific group of users, it is possible to define an embargo policy also for the ITEM itself. A READ policy for a specific group will mean that only the users in that group will be able to access the item splash page. Note that the DSpace REST API & UI is fully rights aware (see Discovery(see page 386) documentation for more information, especially the section on "Access Rights Awareness"), meaning that an embargoed item is hidden automatically until the embargo expires.

7.3 Directories and Files

7.3.1 Overview

A complete DSpace installation consists of three separate directory trees:

485 http://en.wikipedia.org/wiki/Delayed_open_access

- **The source directory:**: This is where (surprise!) the source code lives. Note that the config files here are used only during the initial install process. After the install, config files should be changed in the install directory. It is referred to in this document as *[dspace-source]*.
- **The install directory:**: This directory is populated during the install process and also by DSpace as it runs. It contains config files, command-line tools (and the libraries necessary to run them), and usually -- although not necessarily -- the contents of the DSpace archive (depending on how DSpace is configured). After the initial build and install, changes to config files should be made in this directory. It is referred to in this document as *[dspace]*.
- **The web deployment directory:**: This directory is generated by the web server the first time it finds a dspace.war file in its webapps directory. It contains the unpacked contents of dspace.war, i.e. the JSPs and java classes and libraries necessary to run DSpace. Files in this directory should never be edited directly; if you wish to modify your DSpace installation, you should edit files in the source directory and then rebuild. The contents of this directory aren't listed here since its creation is completely automatic. It is usually referred to in this document as *[tomcat]/webapps/dspace*.

7.3.2 Source Directory Layout

- *[dspace-source]*
  - *LICENSE* - DSpace source code license.
  - *README* - Obligatory basic information file.
  - *dspace/* - Directory which contains all build and configuration information for DSpace
    - *bin/* - Some shell and Perl scripts for running DSpace command-line tasks. Primary among them is the 'dspace' commandline utility(see page 460)
    - *config/* - Configuration files:
      - local.cfg.EXAMPLE - an example "local.cfg(see page 560)" file, which can be used to store all your local configuration overrides. See Configuration Reference(see page 552).
      - *controlled-vocabularies/* - Fixed, limited vocabularies used in metadata entry
      - *crosswalks/* - Metadata crosswalks - property files or XSL stylesheets
      - *emails/* - Text and layout templates for emails sent out by the system.
      - entities/ - Configuration files for Configurable Entities(see page 134)
      - modules/ - Configurations for modules / individual features within DSpace
      - registries/ - **Initial** contents of the bitstream format registry and Dublin Core element/qualifier registry. These are only used on initial system setup, after which they are maintained in the database.
      - spring/ - Spring XML configurations used by DSpace for various features.
      - *dspace.cfg* - The Main DSpace configuration(see page 552) file
      - *dc2mods.cfg* - Mappings from Dublin Core metadata to MODS[486] for the METS export.
      - *default.license* - The default license that users must grant when submitting items.
      - *dstat.cfg* , *dstat.map* - Configuration for statistical reports.
      - *submission-forms.xml* , *item-submission.xml* - Submission UI configuration files(see page 260)
    - *modules/* - The Web UI modules "overlay" directory. DSpace uses Maven to automatically look here for any customizations you wish to make to DSpace Web interfaces. See also Advanced Customisation(see page 499)
    - *solr/* - Solr configuration files for all Solr indexes used by DSpace.
    - *src/* - Maven configurations for DSpace System. This directory contains the Maven and Ant build files for DSpace.
    - *target/* - (Only exists after building DSpace) This is the location Maven uses to build your DSpace installation package.

486 http://www.loc.gov/standards/mods/

    - *dspace-installer*- The location of the DSpace Installation Package (which can then be installed by running *ant update*)
- The Source Release contains the following additional directories :-
    - *dspace-api* - Java API source module (to build the dspace-api.jar)
    - *dspace-oai* - OAI-PMH(see page 179) source module (to build to dspace-oai.jar)
    - *dspace-rdf* - RDF(see page 160) source module (to build to dspace-rdf.jar)
    - *dspace-rest* - REST API v6 (deprecated)(see page 506) source module
    - dspace-server-webapp - Primary backend webapp which hosts the REST API(see page 502), along with any other enabled modules (OAI, RDF, SWORD, etc).
    - *dspace-services* - Common Services module
    - *dspace-sword* - SWORD(see page 216) (Simple Web-service Offering Repository Deposit) deposit service source module
    - *dspace-swordv2* - SWORDv2(see page 202) source module
    - *pom.xml* - DSpace Parent Project definition

7.3.3 Installed Directory Layout

Below is the basic layout of a DSpace installation using the default configuration. These paths can be configured if necessary.

- *[dspace]*
    - *assetstore/* - assetstore files. This is where all the files uploaded into DSpace are stored by default. See Storage Layer(see page 686).
    - *bin/* - shell scripts for DSpace command-line tasks. Primary among them is the 'dspace' commandline utility(see page 460)
    - *config/* - configuration, with sub-directories as above
    - *etc/* - Administrative and database management files
    - *exports/* - temporary storage for any export packages
    - *handle-server/* - Handles server files and configuration
    - *imports/* - temporary storage for any import packages
    - *lib/* - JARs, including dspace-api.jar, containing the DSpace classes
    - *log/* - Log files
    - *reports/* - Reports generated by statistical report generator
    - *solr/* - Solr search/browse indexes
    - *triplestore/* - RDF triple store index files (when enabled)
    - *upload/* - temporary directory used during file uploads etc.
    - *webapps/* - location where DSpace installs all Web Applications

7.3.4 Contents of Server Web Application

DSpace's Ant build file creates a webapps/*server/* directory with the following structure:

- server/
    - index.html - Root page of the third party HAL Browser (used to browse/search REST API(see page 502))
    - login.html - (Custom) Login page for HAL Browser (supporting DSpace authentication plugins)
    - js/ - Javascript overrides for HAL Browser (main HAL Browser code is brought in via Spring REST dependencies)

7.3.5 Log Files

The first source of potential confusion is the log files. Since DSpace uses a number of third-party tools, problems can occur in a variety of places. Below is a table listing the main log files used in a typical DSpace setup. The locations given are defaults, and might be different for your system depending on where you installed DSpace and the third-party tools. The ordering of the list is roughly the recommended order for searching them for the details about a particular problem or error.

Log File	**What's In It**
[dspace]/log/dspace.log.yyyy-mm-dd	Main DSpace log file. This is where the DSpace code writes a simple log of events and errors that occur within the DSpace code. You can control the verbosity of this by editing the *[dspace-source]/config/templates/log4j.properties* file and then running "*ant init_configs*".
[dspace]/log/handle-plugin.log	The Handle server runs as a separate process from the DSpace Web UI (which runs under Tomcat's JVM). Due to a limitation of log4j's 'rolling file appenders', the DSpace code running in the Handle server's JVM must use a separate log file. The DSpace code that is run as part of a Handle resolution request writes log information to this file. You can control the verbosity of this by editing *[dspace-source]/config/templates/log4j-handle-plugin.properties*.
[dspace]/log/handle-server.log	This is the log file for CNRI's Handle server code. If a problem occurs within the Handle server code, before DSpace's plug-in is invoked, this is where it may be logged.
[tomcat]/logs/catalina.out	This is where Tomcat's standard output is written. Many errors that occur within the Tomcat code are logged here. For example, if Tomcat can't find the DSpace code (*dspace.jar*), it would be logged in *catalina.out*.
[tomcat]/logs/hostname_log.yyyy-mm-dd.txt	If you're running Tomcat stand-alone (without Apache), it logs some information and errors for specific Web applications to this log file. *hostname* will be your host name (e.g. *dspace.myu.edu*) and *yyyy-mm-dd* will be the date.
[tomcat]/logs/apache_log.yyyy-mm-dd.txt	If you're using Apache, Tomcat logs information about Web applications running through Apache (*mod_webapp*) in this log file (*yyyy-mm-dd* being the date.)

[apache]/error_log	Apache logs to this file. If there is a problem with getting *mod_webapp* working, this is a good place to look for clues. Apache also writes to several other log files, though *error_log* tends to contain the most useful information for tracking down problems.
PostgreSQL log	PostgreSQL also writes a log file. This one doesn't seem to have a default location, you probably had to specify it yourself at some point during installation. In general, this log file rarely contains pertinent information--PostgreSQL is pretty stable, you're more likely to encounter problems with connecting via JDBC, and these problems will be logged in *dspace.log*.

7.3.5.1 log4j2.xml File.

the file *[dspace]/config/log4j2.xml* controls how and where log files are created. There are three sets of configurations in that file, called A1, and A2. These are used to control the logs for DSpace (as a whole), and the checksum checker respectively. As implied by the name, this configuration use Log4j v2. For more information on syntax, see https://logging.apache.org/log4j/2.x/manual/configuration.html

7.4 Metadata and Bitstream Format Registries

7.4.1 Default Dublin Core Metadata Registry (DC)

The default DSpace Dublin Core Metadata Registry was originally derived from the 15 Dublin Core elements. This registry initializes the default schema, where **dc** is used to identify the namespace. As this registry is meant to track the Dublin Core standard, it's recommended that the local DSpace administrator not add/remove metadata fields from this namespace; the "local" namespace should be used instead (see below).

element	qualifier	scope note
contributor		A person, organization, or service responsible for the content of the resource. Catch-all for unspecified contributors.
contributor	advisor	Use primarily for thesis advisor.
contributor	author[1]	Author(s) of the work (used by default)

element	qualifier	scope note
contributor	editor	
contributor	illustrator	
contributor	other	
coverage	spatial	Spatial characteristics of content.
coverage	temporal	Temporal characteristics of content.
creator		May be used as an alternative to "contributor.author"
date		Use qualified form if possible.
date	accessioned[1]	Date DSpace takes possession of item.
date	available[1]	Date or date range item became available to the public.
date	copyright	Date of copyright.
date	created	Date of creation or manufacture of intellectual content if different from date.issued.
date	issued[1]	Date of publication or distribution.
date	submitted	Recommend for theses/dissertations.
identifier		Catch-all for unambiguous identifiers not defined by qualified form; use identifier.other for a known identifier common to a local collection instead of unqualified form.

element	qualifier	scope note
identifier	citation[2]	Human-readable, standard bibliographic citation of non-DSpace format of this item
identifier	govdoc[2]	A government document number
identifier	isbn[2]	International Standard Book Number
identifier	issn[2]	International Standard Serial Number
identifier	sici	Serial Item and Contribution Identifier
identifier	ismn[2]	International Standard Music Number
identifier	other[2]	A known identifier type common to a local collection.
identifier	uri[1]	Uniform Resource Identifier
description[1]		Catch-all for any description not defined by qualifiers.
description	abstract[1]	Abstract or summary.
description	provenance[1]	The history of custody of the item since its creation, including any changes successive custodians made to it.
description	sponsorship[2]	Information about sponsoring agencies, individuals, or contractual arrangements for the item.
description	statementofresponsibility	To preserve statement of responsibility from MARC records.
description	tableofcontents	A table of contents for a given item.

element	qualifier	scope note
description	uri	Uniform Resource Identifier pointing to description of this item.
format[2]		Catch-all for any format information not defined by qualifiers.
format	extent[2]	Size or duration.
format	medium[2]	Physical medium.
format	mimetype[2]	Registered MIME type identifiers.
language		Catch-all for non-ISO forms of the language of the item, accommodating harvested values.
language	iso[2]	Current ISO standard for language of intellectual content, including country codes (e.g. "en_US").
publisher[2]		Entity responsible for publication, distribution, or imprint.
relation		Catch-all for references to other related items.
relation	isformatof	References additional physical form.
relation	ispartof	References physically or logically containing item.
relation[1]	ispartofseries	Series name and number within that series, if available.
relation	haspart	References physically or logically contained item.
relation	isversionof	References earlier version.

element	qualifier	scope note
relation	hasversion	References later version.
relation	isbasedon	References source.
relation	isreferencedby	Pointed to by referenced resource.
relation	requires	Referenced resource is required to support function, delivery, or coherence of item.
relation	replaces	References preceeding item.
relation	isreplacedby	References succeeding item.
relation	uri	References Uniform Resource Identifier for related item
rights		Terms governing use and reproduction.
rights	uri	References terms governing use and reproduction.
source		Do not use; only for harvested metadata.
source	uri	Do not use; only for harvested metadata.
subject[2]		Uncontrolled index term.
subject	classification	Catch-all for value from local classification system. Global classification systems will receive specific qualifier
subject	ddc	Dewey Decimal Classification Number
subject	lcc	Library of Congress Classification Number

element	qualifier	scope note
subject	lcsh	Library of Congress Subject Headings
subject	mesh	MEdical Subject Headings
subject	other	Local controlled vocabulary; global vocabularies will receive specific qualifier.
title[1]		Title statement/title proper.
title	alternative[2]	Varying (or substitute) form of title proper appearing in item, e.g. abbreviation or translation
type[1]		Nature or genre of content.

[1] Used by several functional areas of DSpace. **DO NOT REMOVE WITHOUT INVESTIGATING THE CONSEQUENCES**

[2] This field is included in the default DSpace Submission User Interface(see page 260). Removing this field from your registry will break the default DSpace submission form.

7.4.2 Dublin Core Terms Registry (DCTERMS)

The Dublin Core Terms (DCTERMS) registry was introduced in DSpace 4. This registry initializes an optional metadata schema, where **dcterms** is used to identify the namespace. In DSpace 4, none of these fields are used by any of the system functionality out of the box. The registry and schema were added as a first step to facilitate the future migration of the DSpace specific DC schema, to this schema that complies to current Dublin Core standards.

The main advantage of the DCTERMS schema is that no field name details gets lost during harvesting, as opposed to harvesting of so called "simple" dublin core, where the qualifiers from the above schema are omitted during harvesting.

As this registry is meant to track the Dublin Core Terms standard, it's recommended that the local DSpace administrator not add/remove metadata fields from this namespace; the "local" namespace should be used instead (see below).

term	scope note
abstract	A summary of the resource.
accessRights	Information about who can access the resource or an indication of its security status. May include information regarding access or restrictions based on privacy, security, or other policies.

term	scope note
accrualMethod	The method by which items are added to a collection.
accrualPeriodicity	The frequency with which items are added to a collection.
accrualPolicy	The policy governing the addition of items to a collection.
alternative	An alternative name for the resource.
audience	A class of entity for whom the resource is intended or useful.
available	Date (often a range) that the resource became or will become available.
bibliographicCitation	Recommended practice is to include sufficient bibliographic detail to identify the resource as unambiguously as possible.
conformsTo	An established standard to which the described resource conforms.
contributor	An entity responsible for making contributions to the resource. Examples of a Contributor include a person, an organization, or a service.
coverage	The spatial or temporal topic of the resource, the spatial applicability of the resource, or the jurisdiction under which the resource is relevant.
created	Date of creation of the resource.
creator	An entity primarily responsible for making the resource.
date	A point or period of time associated with an event in the lifecycle of the resource.
dateAccepted	Date of acceptance of the resource.
dateCopyrighted	Date of copyright.
dateSubmitted	Date of submission of the resource.
description	An account of the resource.

term	scope note
educationLevel	A class of entity, defined in terms of progression through an educational or training context, for which the described resource is intended.
extent	The size or duration of the resource.
format	The file format, physical medium, or dimensions of the resource.
hasFormat	A related resource that is substantially the same as the pre-existing described resource, but in another format.
hasPart	A related resource that is included either physically or logically in the described resource.
hasVersion	A related resource that is a version, edition, or adaptation of the described resource.
identifier	An unambiguous reference to the resource within a given context.
instructionalMethod	A process, used to engender knowledge, attitudes and skills, that the described resource is designed to support.
isFormatOf	A related resource that is substantially the same as the described resource, but in another format.
isPartOf	A related resource in which the described resource is physically or logically included.
isReferencedBy	A related resource that references, cites, or otherwise points to the described resource.
isReplacedBy	A related resource that supplants, displaces, or supersedes the described resource.
isRequiredBy	A related resource that requires the described resource to support its function, delivery, or coherence.
issued	Date of formal issuance (e.g., publication) of the resource.
isVersionOf	A related resource of which the described resource is a version, edition, or adaptation.
language	A language of the resource.
license	A legal document giving official permission to do something with the resource.
mediator	An entity that mediates access to the resource and for whom the resource is intended or useful.

term	scope note
medium	The material or physical carrier of the resource.
modified	Date on which the resource was changed.
provenance	A statement of any changes in ownership and custody of the resource since its creation that are significant for its authenticity, integrity, and interpretation.
publisher	An entity responsible for making the resource available.
references	A related resource that is referenced, cited, or otherwise pointed to by the described resource.
relation	A related resource.
replaces	A related resource that is supplanted, displaced, or superseded by the described resource.
requires	A related resource that is required by the described resource to support its function, delivery, or coherence.
rights	Information about rights held in and over the resource.
rightsHolder	A person or organization owning or managing rights over the resource.
source	A related resource from which the described resource is derived.
spatial	Spatial characteristics of the resource.
subject	The topic of the resource.
tableOfContents	A list of subunits of the resource.
temporal	Temporal characteristics of the resource.
title	A name given to the resource.
type	The nature or genre of the resource.
valid	Date (often a range) of validity of a resource.

7.4.3 Local Metadata Registry (local)

Editing the DC and DCTERMS schemas is recommended against because it may complicate the upgrade path in case a newer version of DSpace needs to make changes or migrations in these standard metadata fields. Therefore, an empty metadata schema called "local" is provided (since DSpace 6), which can be used by the DSpace

administrator as a namespace for custom local metadata fields. Such custom fields would be anything that does not fit into DC or DCTERMS. Future DSpace migrations will not touch fields in the "local" namespace.

element	qualifier	scope note
<empty by default>		<fields to be populated by DSpace administrator if needed>

7.4.4 Default Bitstream Format Registry

Mimetype	Short Description	Description	Support Level	Internal	Extensions
application/octet-stream[1]	Unknown	Unknown data format	Unknown	false	
text/plain[1]	License	Item-specific license agreed upon to submission	Known	true	
application/marc	MARC	Machine-Readable Cataloging records	Known	false	
application/ mathematica	Mathematica	Mathematica Notebook	Known	false	ma
application/ msword	Microsoft Word	Microsoft Word	Known	false	doc
application/pdf	Adobe PDF	Adobe Portable Document Format	Known	false	pdf
application/ postscript	Postscript	Postscript Files	Known	false	ai, eps, ps
application/sgml	SGML	SGML application (RFC 1874)	Known	false	sgm, sgml
application/ vnd.ms-excel	Microsoft Excel	Microsoft Excel	Known	false	xls

application/vnd.ms-powerpoint	Microsoft Powerpoint	Microsoft Powerpoint	Known	false	ppt
application/vnd.ms-project	Microsoft Project	Microsoft Project	Known	false	mpd, mpp, mpx
application/vnd.visio	Microsoft Visio	Microsoft Visio	Known	false	vsd
application/wordperfect5.1	WordPerfect	WordPerfect 5.1 document	Known	false	wpd
application/x-dvi	TeX dvi	TeX dvi format	Known	false	dvi
application/x-filemaker	FMP3	Filemaker Pro	Known	false	fm
application/x-latex	LateX	LaTeX document	Known	false	latex
application/x-photoshop	Photoshop	Photoshop	Known	false	pdd, psd
application/x-tex	TeX	Tex/LateX document	Known	false	tex
audio/basic	audio/basic	Basic Audio	Known	false	au, snd
audio/x-aiff	AIFF	Audio Interchange File Format	Known	false	aif, aifc, aiff
audio/x-mpeg	MPEG Audio	MPEG Audio	Known	false	abs, mpa, mpega
audio/x-pn-realaudio	RealAudio	RealAudio file	Known	false	ra, ram

audio/x-wav	WAV	Broadcase Wave Format	Known	false	wav
image/gif	GIF	Graphics Interchange Format	Known	false	gif
image/jpeg	JPEG	Joint Photographic Experts Group/JPEG File Interchange Format (JFIF)	Known	false	jpeg, jpg
image/png	image/png	Portable Network Graphics	Known	false	png
image/tiff	TIFF	Tag Image File Format	Known	false	tif, tiff
image/x-ms-bmp	BMP	Microsoft Windows bitmap	Known	false	bmp
image/x-photo-cd	Photo CD	Kodak Photo CD image	Known	false	pcd
text/css	CSS	Cascading Style Sheets	Known	false	css
text/html	HTML	Hypertext Markup Language	Known	false	htm, html
text/plain	Text	Plain Text	Known	false	asc, txt
text/richtext	RTF	Rich Text Format	Known	false	rtf
text/xml	XML	Extensible Markup Language	Known	false	xml
video/mpeg	MPEG	Moving Picture Experts Group	Known	false	mpe, mpeg, mpg

video/quicktime	Video Quicktime	Video Quicktime	Known	false	mov, qt

[1] Used by several functional areas of DSpace. **DO NOT REMOVE WITHOUT INVESTIGATING THE CONSEQUENCES**

7.5 Architecture

- Overview(see page 652)

7.5.1 Overview

The DSpace system is organized into three layers, each of which consists of a number of components.

- **Application Layer** - All external/public facing interfaces/tools. These include the Web User Interface, REST API(see page 502), OAI-PMH(see page 179), RDF(see page 160), and SWORD (v1(see page 216) and v2(see page 202)) interfaces. Also includes the Command Line(see page 460) interface, and various tools that can be used to import/export data to/from DSpace.
- **Business Logic Layer** - Primarily the Java API layer ([dspace-source]/dspace-api and dspace-services), which provides the core business logic for all the various application interfaces.
- **Storage Layer** - A subset of the dspace-api (*org.dspace.storage.* classes)* whose role is to manage all content storage (metadata, relationships, bitstreams) for all business layer objects. This layer is provides access to a relational database (Postgres or Oracle usually) via Hibernate ORM[487] & using FlywayDB[488] for migrations/updates. It also defines a custom BitStoreService for storing files (bitstreams) via storage plugins (currently supporting filesystem storage or Amazon S3 storage).

DSpace System Architecture

The storage layer is responsible for physical storage of metadata and content. The business logic layer deals with managing the content of the archive, users of the archive (e-people), authorization, and workflow. The application layer contains components that communicate with the world outside of the individual DSpace installation, for example the Web user interface and the Open Archives Initiative[489] protocol for metadata harvesting service.

Each layer only invokes the layer below it; the application layer may not use the storage layer directly, for example. Each component in the storage and business logic layers has a defined public API. The union of the APIs of those components are referred to as the Storage API (in the case of the storage layer) and the DSpace Java API (in the case of the business logic layer), and the DSpace REST API (in the case of the application layer). In the Application Layer, it's worth noting that the Web User Interface only accesses DSpace via the REST API.

It is important to note that each layer is *trusted*. Although the logic for *authorising actions* is in the business logic layer, the system relies on individual applications in the application layer to correctly and securely *authenticate* e-people. If a 'hostile' or insecure application were allowed to invoke the Java API directly, it could very easily perform actions as any e-person in the system.

The reason for this design choice is that authentication methods will vary widely between different applications, so it makes sense to leave the logic and responsibility for that in these applications.

The source code is organized to cohere very strictly to this three-layer architecture.

487 https://hibernate.org/orm/
488 https://flywaydb.org/
489 http://www.openarchives.org/

The storage and business logic layer APIs are extensively documented with Javadoc-style comments. Generate the HTML version of these by entering the [dspace-source]/dspace directory and running:

```
mvn javadoc:javadoc
```

The resulting documentation will be at *[dspace-source]dspace-api/target/site/apidocs/index.html*. The package-level documentation of each package usually contains an overview of the package and some example usage. This information is not repeated in this architecture document; this and the Javadoc APIs are intended to be used in parallel.

The REST API provides not only JavaDocs, but also a public contract. See REST API(see page 502).

Each layer is described in a separate section:

- Storage Layer(see page 686)
  - RDBMS
  - Bitstream Store
- Business Logic Layer(see page 655)
  - Core Classes
  - Content Management API
  - Workflow System
  - Administration Toolkit
  - E-person/Group Manager
  - Authorisation
  - Handle Manager/Handle Plugin
  - Search
  - Browse API
  - History Recorder
  - Checksum Checker
- Application Layer(see page 653)
  - Web User Interface
  - OAI-PMH Data Provider
  - Item Importer and Exporter
  - Transferring Items Between DSpace Instances
  - Registration
  - METS Tools
  - Media Filters
  - Sub-Community Management

7.5.2 Application Layer

The following explains the components of the Application Layer.

- Web User Interface(see page 654)
  - Web UI Files(see page 654)
- REST API(see page 654)
- OAI-PMH Data Provider(see page 654)
- RDF / Linked Data Provider(see page 654)
- SWORD v1 Service / Server(see page 654)
- SWORD v2 Service / Server(see page 654)
- DSpace Command Line Launcher(see page 654)
  - Command Launcher Structure(see page 654)

7.5.2.1 Web User Interface

The DSpace Web UI is the largest and most-used component in the application layer. As of DSpace 7, it has been rebuilt on Angular.io[490] communicating via the REST API(see page 502) to the rest of DSpace.

Web UI Files

The web User Interface code is managed in a separate GitHub Project:

https://github.com/DSpace/dspace-angular/

Quick setup and configuration instructions can be found in the README of that project.

7.5.2.2 REST API

This component defines the main public API of the Application Layer. See REST API(see page 502) section of the documentation.

7.5.2.3 OAI-PMH Data Provider

See OAI(see page 179) section of the documentation

7.5.2.4 RDF / Linked Data Provider

See Linked (Open) Data(see page 160) section of the documentation

7.5.2.5 SWORD v1 Service / Server

See SWORDv1 Server(see page 216) section of the documentation

7.5.2.6 SWORD v2 Service / Server

See SWORDv2 Server(see page 202) section of the documentation

7.5.2.7 DSpace Command Line Launcher

The DSpace Command Launcher brings together the various command and scripts into a standard-practice for running CLI runtime programs. See Command Line Operations(see page 460)

Command Launcher Structure

There are two components to the command launcher: the dspace script and the launcher.xml. The DSpace command calls a java class which in turn refers to *launcher.xml* that is stored in the *[dspace]/config* directory

launcher.xml is made of several components:

- <*command*> begins the stanza for a command
- <*name*>*_name of command_*</*name*> the name of the command that you would use.

490 https://angular.io/

- *<description>_the description of the command_</description>*
- *<step> </step>* User arguments are parsed and tested.
- *<class>_<the java class that is being used to run the CLI program>_</class>*

See Command Line Operations(see page 460) *for additional details.*

7.5.3 Business Logic Layer

7.5.3.1 Core Classes

The *org.dspace.core* package provides some basic classes that are used throughout the DSpace code.

The Configuration Service

The configuration service is responsible for reading the main *dspace.cfg* properties file, managing the 'template' configuration files for other applications such as Apache, and for obtaining the text for e-mail messages.

The system is configured by editing the relevant files in `[dspace]/config`, as described in the configuration section.

When editing configuration files for applications that DSpace uses, such as Apache Tomcat, you may want to edit the copy in `[dspace-source]` and then run `ant update` or `ant overwrite_configs` rather than editing the 'live' version directly! This will ensure you have a backup copy of your modified configuration files, so that they are not accidentally overwritten in the future.

The *ConfigurationService* class can also be invoked as a command line tool:

- `[dspace]/bin/dspace dsprop property.name` This writes the value of *property.name* from *dspace.cfg* to the standard output, so that shell scripts can access the DSpace configuration. If the property has no value, nothing is written.

For many more details on configuration in DSpace, see Configuration Reference(see page 552)

Constants

This class contains constants that are used to represent types of object and actions in the database. For example, authorization policies can relate to objects of different types, so the *resourcepolicy* table has columns *resource_id*, which is the internal ID of the object, and *resource_type_id*, which indicates whether the object is an item, collection, bitstream etc. The value of *resource_type_id* is taken from the *Constants* class, for example *Constants.ITEM*.

Here are a some of the most commonly used constants you might come across:

DSpace types

- Bitstream: 0
- Bundle: 1
- Item: 2
- Collection: 3
- Community: 4
- Site: 5
- Group: 6
- Eperson: 7

DSpace actions

- Read: 0

- Write: 1
- Delete: 2
- Add: 3
- Remove: 4

Refer to the org.dspace.core.Constants[491] for all of the Constants.

Context

The *Context* class is central to the DSpace operation. Any code that wishes to use the any API in the business logic layer must first create itself a *Context* object. This is akin to opening a connection to a database (which is in fact one of the things that happens.)

A context object is involved in most method calls and object constructors, so that the method or object has access to information about the current operation. When the context object is constructed, the following information is automatically initialized:

- A connection to the database. This is a transaction-safe connection. i.e. the 'auto-commit' flag is set to false.
- A cache of content management API objects. Each time a content object is created (for example *Item* or *Bitstream*) it is stored in the *Context* object. If the object is then requested again, the cached copy is used. Apart from reducing database use, this addresses the problem of having two copies of the same object in memory in different states.
  The following information is also held in a context object, though it is the responsibility of the application creating the context object to fill it out correctly:
- The current authenticated user, if any
- Any 'special groups' the user is a member of. For example, a user might automatically be part of a particular group based on the IP address they are accessing DSpace from, even though they don't have an e-person record. Such a group is called a 'special group'.
- Any extra information from the application layer that should be added to log messages that are written within this context. For example, the Web UI adds a session ID, so that when the logs are analyzed the actions of a particular user in a particular session can be tracked.
- A flag indicating whether authorization should be circumvented. This should only be used in rare, specific circumstances. For example, when first installing the system, there are no authorized administrators who would be able to create an administrator account!As noted above, the public API is *trusted*, so it is up to applications in the application layer to use this flag responsibly.
  Typical use of the context object will involve constructing one, and setting the current user if one is authenticated. Several operations may be performed using the context object. If all goes well, *complete* is called to commit the changes and free up any resources used by the context. If anything has gone wrong, *abort* is called to roll back any changes and free up the resources.

You should always *abort* a context if *any* error happens during its lifespan; otherwise the data in the system may be left in an inconsistent state. You can also *commit* a context, which means that any changes are written to the database, and the context is kept active for further use.

Email

Sending e-mails is pretty easy. Just use the configuration manager's *getEmail* method, set the arguments and recipients, and send.

The e-mail texts are stored in `[dspace]/config/emails`. They are processed by the standard *java.text.MessageFormat*. At the top of each e-mail are listed the appropriate arguments that should be filled out by the sender. Example usage is shown in the *org.dspace.core.Email* Javadoc API documentation.

---

491 https://github.com/DSpace/DSpace/blob/master/dspace-api/src/main/java/org/dspace/core/Constants.java

LogManager

The log manager consists of a method that creates a standard log header, and returns it as a string suitable for logging. Note that this class does not actually write anything to the logs; the log header returned should be logged directly by the sender using an appropriate Log4J call, so that information about where the logging is taking place is also stored.

The level of logging can be configured on a per-package or per-class basis by editing `[dspace]/config/log4j.properties`. You will need to stop and restart Tomcat for the changes to take effect.

A typical log entry looks like this:

2002-11-11 08:11:32,903 INFO org.dspace.app.webui.servlet.DSpaceServlet @ anonymous:session_id=BD84E7C194C2CF4BD0EC3A6CAD0142BB:view_item:handle=1721.1/1686

This is breaks down like this:

Date and time, milliseconds	*2002-11-11 08:11:32,903*
Level (*FATAL*, *WARN*, *INFO* or *DEBUG*)	*INFO*
Java class	*org.dspace.app.webui.servlet.DSpaceServlet*
	@
User email or *anonymous*	*anonymous*
	:
Extra log info from context	*session_id=BD84E7C194C2CF4BD0EC3A6CAD0142BB*
	:
Action	*view_item*
	:
Extra info	*handle=1721.1/1686*

The above format allows the logs to be easily parsed and analyzed. The `[dspace]/bin/log-reporter` script is a simple tool for analyzing logs. Try:

```
[dspace]/bin/log-reporter --help
```

It's a good idea to 'nice' this log reporter to avoid an impact on server performance.

Utils

Utils contains miscellaneous utility method that are required in a variety of places throughout the code, and thus have no particular 'home' in a subsystem.

7.5.3.2 Content Management API

The content management API package *org.dspace.content* contains Java classes for reading and manipulating content stored in the DSpace system. This is the API that components in the application layer will probably use most.

Classes corresponding to the main elements in the DSpace data model (*Community*, *Collection*, *Item*, *Bundle* and *Bitstream*) are sub-classes of the abstract class *DSpaceObject*. The *Item* object handles the Dublin Core metadata record.

Each class generally has one or more static *find* methods, which are used to instantiate content objects. Constructors do not have public access and are just used internally. The reasons for this are:

- "Constructing" an object may be misconstrued as the action of creating an object in the DSpace system, for example one might expect something like:

  ```
  Context dsContent = new Context();
  Item myItem = new Item(context, id)
  ```

  to construct a brand new item in the system, rather than simply instantiating an in-memory instance of an object in the system.
- *find* methods may often be called with invalid IDs, and return *null* in such a case. A constructor would have to throw an exception in this case. A *null* return value from a static method can in general be dealt with more simply in code.
- If an instantiation representing the same underlying archival entity already exists, the *find* method can simply return that same instantiation to avoid multiple copies and any inconsistencies which might result.

Collection, *Bundle* and *Bitstream* do not have *create* methods; rather, one has to create an object using the relevant method on the container. For example, to create a collection, one must invoke *createCollection* on the community that the collection is to appear in:

```
Context context = new Context();
Community existingCommunity = Community.find(context, 123);
Collection myNewCollection = existingCommunity.createCollection();
```

The primary reason for this is for determining authorization. In order to know whether an e-person may create an object, the system must know which container the object is to be added to. It makes no sense to create a collection outside of a community, and the authorization system does not have a policy for that.

Items are first created in the form of an implementation of InProgressSubmission. An *InProgressSubmission* represents an item under construction; once it is complete, it is installed into the main archive and added to the relevant collection by the *InstallItem* class. The *org.dspace.content* package provides an implementation of *InProgressSubmission* called *WorkspaceItem*; this is a simple implementation that contains some fields used by the

Web submission UI. The *org.dspace.workflow* also contains an implementation called *WorkflowItem* which represents a submission undergoing a workflow process.

In the previous chapter there is an overview of the item ingest process which should clarify the previous paragraph. Also see the section on the workflow system.

Community and *BitstreamFormat* do have static *create* methods; one must be a site administrator to have authorization to invoke these.

Other Classes

Classes whose name begins *DC* are for manipulating Dublin Core metadata, as explained below.

The *FormatIdentifier* class attempts to guess the bitstream format of a particular bitstream. Presently, it does this simply by looking at any file extension in the bitstream name and matching it up with the file extensions associated with bitstream formats. Hopefully this can be greatly improved in the future!

The *ItemIterator* class allows items to be retrieved from storage one at a time, and is returned by methods that may return a large number of items, more than would be desirable to have in memory at once.

The *ItemComparator* class is an implementation of the standard *java.util.Comparator* that can be used to compare and order items based on a particular Dublin Core metadata field.

Modifications

When creating, modifying or for whatever reason removing data with the content management API, it is important to know when changes happen in-memory, and when they occur in the physical DSpace storage.

Primarily, one should note that no change made using a particular *org.dspace.core.Context* object will actually be made in the underlying storage unless *complete* or *commit* is invoked on that *Context*. If anything should go wrong during an operation, the context should always be aborted by invoking *abort*, to ensure that no inconsistent state is written to the storage.

Additionally, some changes made to objects only happen in-memory. In these cases, invoking the *update* method lines up the in-memory changes to occur in storage when the *Context* is committed or completed. In general, methods that change any metadata field only make the change in-memory; methods that involve relationships with other objects in the system line up the changes to be committed with the context. See individual methods in the API Javadoc.

Some examples to illustrate this are shown below:

`` Context context = new Context(); Bitstream b = Bitstream.find(context, 1234); b.setName("newfile.txt"); b.update(); context.complete(); ```	**Will** change storage
`` Context context = new Context(); Bitstream b = Bitstream.find(context, 1234); b.setName("newfile.txt"); b.update(); context.abort(); ```	**Will not** change storage (context aborted)
`` Context context = new Context(); Bitstream b = Bitstream.find(context, 1234); b.setName("newfile.txt"); context.complete(); ```	The new name **will not** be stored since *update* was not invoked
`` Context context = new Context(); Bitstream bs = Bitstream.find(context, 1234); Bundle bnd = Bundle.find(context, 5678); bnd.add(bs); context.complete(); ```	The bitstream **will** be included in the bundle, since *update* doesn't need to be called

What's In Memory?

Instantiating some content objects also causes other content objects to be loaded into memory.

Instantiating a *Bitstream* object causes the appropriate *BitstreamFormat* object to be instantiated. Of course the *Bitstream* object does not load the underlying bits from the bitstream store into memory!

Instantiating a *Bundle* object causes the appropriate *Bitstream* objects (and hence *BitstreamFormat*s) to be instantiated.

Instantiating an *Item* object causes the appropriate *Bundle* objects (etc.) and hence *BitstreamFormat*s to be instantiated. All the Dublin Core metadata associated with that item are also loaded into memory.

The reasoning behind this is that for the vast majority of cases, anyone instantiating an item object is going to need information about the bundles and bitstreams within it, and this methodology allows that to be done in the most efficient way and is simple for the caller. For example, in the Web UI, the servlet (controller) needs to pass information about an item to the viewer (JSP), which needs to have all the information in-memory to display the item without further accesses to the database which may cause errors mid-display.

You do not need to worry about multiple in-memory instantiations of the same object, or any inconsistencies that may result; the *Context* object keeps a cache of the instantiated objects. The *find* methods of classes in *org.dspace.content* will use a cached object if one exists.

It may be that in enough cases this automatic instantiation of contained objects reduces performance in situations where it is important; if this proves to be true the API may be changed in the future to include a *loadContents* method or somesuch, or perhaps a Boolean parameter indicating what to do will be added to the *find* methods.

When a *Context* object is completed, aborted or garbage-collected, any objects instantiated using that context are invalidated and should not be used (in much the same way an AWT button is invalid if the window containing it is destroyed).

Dublin Core Metadata

The Metadatum class is a simple container that represents a single Dublin Core-like element, optional qualifier, value and language. Note that since DSpace 1.4 the *MetadataValue* and associated classes are preferred (see Support for Other Metadata Schemas). The other classes starting with *DC* are utility classes for handling types of data in Dublin Core, such as people's names and dates. As supplied, the DSpace registry of elements and qualifiers corresponds to the Library Application Profile[492] for Dublin Core. It should be noted that these utility classes assume that the values will be in a certain syntax, which will be true for all data generated within the DSpace system, but since Dublin Core does not always define strict syntax, this may not be true for Dublin Core originating outside DSpace.

Below is the specific syntax that DSpace expects various fields to adhere to:

Element	**Qualifier**	**Syntax**	**Helper Class**
date	Any or unqualified	ISO 8601 in the UTC time zone, with either year, month, day, or second precision. Examples:_2000 2002-10 2002-08-14 1999-01-01T14:35:23Z _	*DCDate*
contributor	Any or unqualified	In general last name, then a comma, then first names, then any additional information like "Jr.". If the contributor is an organization, then simply the name. Examples:_Doe, John Smith, John Jr. van Dyke, Dick Massachusetts Institute of Technology _	*DCPersonName*
language	*iso*	A two letter code taken ISO 639, followed optionally by a two letter country code taken from ISO 3166. Examples:_en fr en_US _	*DCLanguage*

492 http://www.dublincore.org/documents/2002/09/24/library-application-profile/

relation	*ispartofseries*	The series name, following by a semicolon followed by the number in that series. Alternatively, just free text._MIT-TR; 1234 My Report Series; ABC-1234 NS1234 _	*DCSeriesNumber*

Support for Other Metadata Schemas

To support additional metadata schemas a new set of metadata classes have been added. These are backwards compatible with the DC classes and should be used rather than the DC specific classes wherever possible. Note that hierarchical metadata schemas are not currently supported, only flat schemas (such as DC) are able to be defined.

The *MetadataField* class describes a metadata field by schema, element and optional qualifier. The value of a *MetadataField* is described by a *MetadataValue* which is roughly equivalent to the older Metadatum class. Finally the *MetadataSchema* class is used to describe supported schemas. The DC schema is supported by default. Refer to the javadoc for method details.

Packager Plugins

The Packager plugins let you *ingest* a package to create a new DSpace Object, and *disseminate* a content Object as a package. A package is simply a data stream; its contents are defined by the packager plugin's implementation.

To ingest an object, which is currently only implemented for Items, the sequence of operations is:

1. Get an instance of the chosen *PackageIngester* plugin.
2. Locate a Collection in which to create the new Item.
3. Call its *ingest* method, and get back a *WorkspaceItem*.
   The packager also takes a *PackageParameters* object, which is a property list of parameters specific to that packager which might be passed in from the user interface.

Here is an example package ingestion code fragment:

```
Collection collection =  find target collection
     InputStream source = ...;
     PackageParameters params = ...;
     String license = null;

    PackageIngester sip = (PackageIngester) PluginManager
            .getNamedPlugin(PackageIngester.class, packageType);

    WorkspaceItem wi = sip.ingest(context, collection, source, params, license);
```

Here is an example of a package dissemination:

```
OutputStream destination = ...;
     PackageParameters params = ...;
     DSpaceObject dso = ...;

     PackageIngester dip = (PackageDisseminator) PluginManager
             .getNamedPlugin(PackageDisseminator.class, packageType);

     dip.disseminate(context, dso, params, destination);
```

7.5.3.3 Plugin Service

> ⚠ In DSpace 6, the old "PluginManager" was replaced by `org.dspace.core.service.PluginService` which performs the same activities/actions.

The PluginService is a very simple component container. It creates and organizes components (plugins), and helps select a plugin in the cases where there are many possible choices. It also gives some limited control over the life cycle of a plugin.

Concepts

The following terms are important in understanding the rest of this section:

- **Plugin Interface** A Java interface, the defining characteristic of a plugin. The consumer of a plugin asks for its plugin by interface.
- **Plugin** a.k.a. Component, this is an instance of a class that implements a certain interface. It is interchangeable with other implementations, so that any of them may be "plugged in", hence the name. A Plugin is an instance of any class that implements the plugin interface.
- **Implementation class** The actual class of a plugin. It may implement several plugin interfaces, but must implement at least one.
- **Name** Plugin implementations can be distinguished from each other by name, a short String meant to symbolically represent the implementation class. They are called "named plugins". Plugins only need to be named when the caller has to make an active choice between them.
- **SelfNamedPlugin class** Plugins that extend the *SelfNamedPlugin* class can take advantage of additional features of the Plugin Manager. Any class can be managed as a plugin, so it is not necessary, just possible.
- **Reusable** Reusable plugins are only instantiated once, and the Plugin Manager returns the same (cached) instance whenever that same plugin is requested again. This behavior can be turned off if desired.

Using the Plugin Service

Types of Plugin

The Plugin Service supports three different patterns of usage:

1. **Singleton Plugins** There is only one implementation class for the plugin. It is indicated in the configuration. This type of plugin chooses an implementation of a service, for the entire system, at configuration time. Your application just fetches the plugin for that interface and gets the configured-in choice. See the `getSinglePlugin()` method.
2. **Sequence Plugins** You need a sequence or series of plugins, to implement a mechanism like Stackable Authentication or a pipeline, where each plugin is called in order to contribute its implementation of a process to the whole. The Plugin Manager supports this by letting you configure a sequence of plugins for a given interface. See the `getPluginSequence()` method.
3. **Named Plugins** Use a named plugin when the application has to choose one plugin implementation out of many available ones. Each implementation is bound to one or more names (symbolic identifiers) in the configuration. The name is just a string to be associated with the combination of implementation class and interface. It may contain any characters except for comma (,) and equals (=). It may contain embedded spaces. Comma is a special character used to separate names in the configuration entry. Names must be unique within an interface: No plugin classes implementing the same interface may have the same name. Think of plugin names as a controlled vocabulary – for a given plugin interface, there is a set of names for which plugins can be found. The designer of a Named Plugin interface is responsible for deciding what the

name means and how to derive it; for example, names of metadata crosswalk plugins may describe the target metadata format. See the `getNamedPlugin()` method and the `getAllPluginNames()` methods.

Self-Named Plugins

Named plugins can get their names either from the configuration or, for a variant called self-named plugins, from within the plugin itself.

Self-named plugins are necessary because one plugin implementation can be configured itself to take on many "personalities", each of which deserves its own plugin name. It is already managing its own configuration for each of these personalities, so it makes sense to allow it to export them to the Plugin Manager rather than expecting the plugin configuration to be kept in sync with it own configuration.

An example helps clarify the point: There is a named plugin that does crosswalks, call it *CrosswalkPlugin*. It has several implementations that crosswalk some kind of metadata. Now we add a new plugin which uses XSL stylesheet transformation (XSLT) to crosswalk many types of metadata – so the single plugin can act like many different plugins, depending on which stylesheet it employs.

This XSLT-crosswalk plugin has its own configuration that maps a Plugin Name to a stylesheet – it has to, since of course the Plugin Manager doesn't know anything about stylesheets. It becomes a self-named plugin, so that it reads its configuration data, gets the list of names to which it can respond, and passes those on to the Plugin Manager.

When the Plugin Service creates an instance of the XSLT-crosswalk, it records the Plugin Name that was responsible for that instance. The plugin can look at that Name later in order to configure itself correctly for the Name that created it. This mechanism is all part of the SelfNamedPlugin class which is part of any self-named plugin.

Obtaining a Plugin Instance

The most common thing you will do with the Plugin Service is obtain an instance of a plugin. To request a plugin, you must always specify the plugin interface you want. You will also supply a name when asking for a named plugin.

A sequence plugin is returned as an array of _Object_s since it is actually an ordered list of plugins.

See the getSinglePlugin(), getPluginSequence(), getNamedPlugin() methods.

Lifecycle Management

When *PluginService* fulfills a request for a plugin, a new instance is always created.

Getting Meta-Information

The *PluginService* can list all the names of the Named Plugins which implement an interface. You may need this, for example, to implement a menu in a user interface that presents a choice among all possible plugins. See the getAllPluginNames() method.

Note that it only returns the plugin name, so if you need a more sophisticated or meaningful "label" (i.e. a key into the I18N message catalog) then you should add a method to the plugin itself to return that.

Implementation

Note: The *PluginService* refers to interfaces and classes internally only by their names whenever possible, to avoid loading classes until absolutely necessary (i.e. to create an instance). As you'll see below, self-named classes still have to be loaded to query them for names, but for the most part it can avoid loading classes. This saves a lot of time at start-up and keeps the JVM memory footprint down, too. As the Plugin Manager gets used for more classes, this will become a greater concern.

The only downside of "on-demand" loading is that errors in the configuration don't get discovered right away. The solution is to call the *checkConfiguration()* method after making any changes to the configuration.

LegacyPluginServiceImpl Class

The *LegacyPluginServiceImpl* class is the default PluginService implementation. While it is possible to implement your own version of PluginService, no other implementations are provided with DSpace

Here are the public methods, followed by explanations:

- `Object getSinglePlugin(Class interfaceClass)` - Returns an instance of the singleton (single) plugin implementing the given interface. There must be exactly one single plugin configured for this interface, otherwise the *PluginConfigurationError* is thrown. Note that this is the only "get plugin" method which throws an exception. It is typically used at initialization time to set up a permanent part of the system so any failure is fatal. See the *plugin.single* configuration key for configuration details.
- `Object[] getPluginSequence(Class interfaceClass)` - Returns instances of all plugins that implement the interface *interfaceClass*, in an *Array*. Returns an empty array if no there are no matching plugins. The order of the plugins in the array is the same as their class names in the configuration's value field. See the *plugin.sequence* configuration key for configuration details.
- `Object getNamedPlugin(Class interfaceClass, String name)` - Returns an instance of a plugin that implements the interface *interfaceClass* and is bound to a name matching name. If there is no matching plugin, it returns null. The names are matched by *String.equals()*. See the *plugin.named* and *plugin.selfnamed* configuration keys for configuration details.
- `String[] getAllPluginNames(Class interfaceClass)` - Returns all of the names under which a named plugin implementing the interface *interfaceClass* can be requested (with *getNamedPlugin()*). The array is empty if there are no matches. Use this to populate a menu of plugins for interactive selection, or to document what the possible choices are. The names are NOT returned in any predictable order, so you may wish to sort them first. Note: Since a plugin may be bound to more than one name, the list of names this returns does not represent the list of plugins. To get the list of unique implementation classes corresponding to the names, you might have to eliminate duplicates (i.e. create a Set of classes).

SelfNamedPlugin Class

A named plugin implementation must extend this class if it wants to supply its own Plugin Name(s). See Self-Named Plugins for why this is sometimes necessary.

```
abstract class SelfNamedPlugin
{
    // Your class must override this:
    // Return all names by which this plugin should be known.
    public static String[] getPluginNames();

    // Returns the name under which this instance was created.
    // This is implemented by SelfNamedPlugin and should NOT be
    overridden.
    public String getPluginInstanceName();
}
```

Errors and Exceptions

```
public class PluginConfigurationError extends Error
{
    public PluginConfigurationError(String message);
}
```

An error of this type means the caller asked for a single plugin, but either there was no single plugin configured matching that interface, or there was more than one. Either case causes a fatal configuration error.

```
public class PluginInstantiationException extends RuntimeException
{
    public PluginInstantiationException(String msg, Throwable cause)
}
```

This exception indicates a fatal error when instantiating a plugin class. It should only be thrown when something unexpected happens in the course of instantiating a plugin, e.g. an access error, class not found, etc. Simply not finding a class in the configuration is not an exception.

This is a *RuntimeException* so it doesn't have to be declared, and can be passed all the way up to a generalized fatal exception handler.

Configuring Plugins

All of the Plugin Service's configuration comes from the DSpace Configuration Service (see Configuration Reference(see page 552)). You can configure these characteristics of each plugin:

1. **Interface**: Classname of the Java interface which defines the plugin, including package name. e.g. *org.dspace.app.mediafilter.FormatFilter*
2. **Implementation Class**: Classname of the implementation class, including package. e.g. *org.dspace.app.mediafilter.PDFFilter*
3. **Names**: (Named plugins only) There are two ways to bind names to plugins: listing them in the value of a plugin.named.interface key, or configuring a class in *plugin.selfnamed.interface* which extends the *SelfNamedPlugin* class.
4. **Reusable option**: (Optional) This is declared in a *plugin.reusable* configuration line. Plugins are reusable by default, so you only need to configure the non-reusable ones.

Configuring Singleton (Single) Plugins

This entry configures a Single Plugin for use with getSinglePlugin():

```
plugin.single.interface = classname
```

For example, this configures the class *org.dspace.checker.SimpleDispatcher* as the plugin for interface *org.dspace.checker.BitstreamDispatcher*:

```
plugin.single.org.dspace.checker.BitstreamDispatcher=org.dspace.checker.SimpleDispatcher
```

Configuring Sequence of Plugins

This kind of configuration entry defines a Sequence Plugin, which is bound to a sequence of implementation classes. The key identifies the interface, and the value is a comma-separated list of classnames: plugin.sequence.interface = classname, ...
The plugins are returned by *getPluginSequence()* in the same order as their classes are listed in the configuration value.

For example, this entry configures Stackable Authentication with three implementation classes:

```
plugin.sequence.org.dspace.eperson.AuthenticationMethod = \
            org.dspace.eperson.X509Authentication, \
            org.dspace.eperson.PasswordAuthentication, \
            edu.mit.dspace.MITSpecialGroup
```

Configuring Named Plugins

There are two ways of configuring named plugins:

1. **Plugins Named in the Configuration** A named plugin which gets its name(s) from the configuration is listed in this kind of entry:_plugin.named.interface = classname = name [, name..] [classname = name..]_The syntax of the configuration value is: classname, followed by an equal-sign and then at least one plugin name. Bind more names to the same implementation class by adding them here, separated by commas. Names may include any character other than comma (,) and equal-sign (=).For example, this entry creates one plugin with the names GIF, JPEG, and image/png, and another with the name TeX:

   ```
   plugin.named.org.dspace.app.mediafilter.MediaFilter = \
           org.dspace.app.mediafilter.JPEGFilter = GIF, JPEG, image/png \
           org.dspace.app.mediafilter.TeXFilter = TeX
   ```

   This example shows a plugin name with an embedded whitespace character. Since comma (,) is the separator character between plugin names, spaces are legal (between words of a name; leading and trailing spaces are ignored).This plugin is bound to the names "Adobe PDF", "PDF", and "Portable Document Format".

   ```
   plugin.named.org.dspace.app.mediafilter.MediaFilter = \
         org.dspace.app.mediafilter.TeXFilter = TeX \
         org.dspace.app.mediafilter.PDFFilter =  Adobe PDF, PDF, Portable Document Format
   ```

   NOTE: Since there can only be one key with plugin.named. followed by the interface name in the configuration, all of the plugin implementations must be configured in that entry.
2. **Self-Named Plugins** Since a self-named plugin supplies its own names through a static method call, the configuration only has to include its interface and classname:*plugin.selfnamed.interface = classname [, classname..] The following example first demonstrates how the plugin class, XsltDisseminationCrosswalk* is configured to implement its own names "MODS" and "DublinCore". These come from the keys starting with *crosswalk.dissemination.stylesheet.*. The value is a stylesheet file. The class is then configured as a self-named plugin:

```
crosswalk.dissemination.stylesheet.DublinCore = xwalk/TESTDIM-2-DC_copy.xsl
crosswalk.dissemination.stylesheet.MODS = xwalk/mods.xsl

plugin.selfnamed.crosswalk.org.dspace.content.metadata.DisseminationCrosswalk = \
        org.dspace.content.metadata.MODSDisseminationCrosswalk, \
        org.dspace.content.metadata.XsltDisseminationCrosswalk
```

NOTE: Since there can only be one key with *plugin.selfnamed.* followed by the interface name in the configuration, all of the plugin implementations must be configured in that entry. The *MODSDisseminationCrosswalk* class is only shown to illustrate this point.

Use Cases

Here are some usage examples to illustrate how the Plugin Service works.

Managing the MediaFilter plugins transparently

The MediaFilterService implementation relies heavily on the Plugin Service. The MediaFilter classes become plugins named in the configuration. Refer to the Configuration Reference(see page 552) for further details.

A Singleton Plugin

This shows how to configure and access a single anonymous plugin, such as the BitstreamDispatcher plugin:

Configuration:

plugin.single.org.dspace.checker.BitstreamDispatcher=org.dspace.checker.SimpleDispatcher

The following code fragment shows how dispatcher, the service object, is initialized and used:

```
BitstreamDispatcher dispatcher =
(BitstreamDispatcher)PluginManager.getSinglePlugin(BitstreamDispatcher.class);

int id = dispatcher.next();

while (id != BitstreamDispatcher.SENTINEL)
{
     /*
        do some processing here
     */

     id = dispatcher.next();
}
```

Plugin that Names Itself

This crosswalk plugin acts like many different plugins since it is configured with different XSL translation stylesheets. Since it already gets each of its stylesheets out of the DSpace configuration, it makes sense to have the plugin give PluginService the names to which it answers instead of forcing someone to configure those names in two places (and try to keep them synchronized).

Here is the configuration file listing both the plugin's own configuration and the *PluginService* config line:

```
crosswalk.dissemination.stylesheet.DublinCore = xwalk/TESTDIM-2-DC_copy.xsl
crosswalk.dissemination.stylesheet.MODS = xwalk/mods.xsl

plugin.selfnamed.org.dspace.content.metadata.DisseminationCrosswalk = \
  org.dspace.content.metadata.XsltDisseminationCrosswalk
```

This look into the implementation shows how it finds configuration entries to populate the array of plugin names returned by the *getPluginNames()* method. Also note, in the *getStylesheet()* method, how it uses the plugin name that created the current instance (returned by *getPluginInstanceName()*) to find the correct stylesheet.

```
public class XsltDisseminationCrosswalk extends SelfNamedPlugin
{
    ....
    private final String prefix =
    "crosswalk.dissemination.stylesheet.";
    ....
    public static String[] getPluginNames()
    {
        List aliasList = new ArrayList();
        Enumeration pe = ConfigurationManager.propertyNames();

        while (pe.hasMoreElements())
        {
            String key = (String)pe.nextElement();
            if (key.startsWith(prefix))
                aliasList.add(key.substring(prefix.length()));
        }
        return (String[])aliasList.toArray(new
    String[aliasList.size()]);
    }

    // get the crosswalk stylesheet for an instance of the plugin:
    private String getStylesheet()
    {
        return ConfigurationManager.getProperty(prefix +
    getPluginInstanceName());
    }
}
```

Stackable Authentication

The Stackable Authentication mechanism needs to know all of the plugins configured for the interface, in the order of configuration, since order is significant. It gets a Sequence Plugin from the Plugin Manager. Refer to the Configuration Section on Stackable Authentication for further details.

7.5.3.4 Workflow System

The primary classes are:

org.dspace.content.WorkspaceItem	contains an Item before it enters a workflow

org.dspace.workflow.WorkflowItem	contains an Item while in a workflow
org.dspace.workflow.WorkflowService	responds to events, manages the WorkflowItem states. There are two implementations, the traditional, default workflow (described below) and Configurable Workflow(see page 250).
org.dspace.content.Collection	contains List of defined workflow steps
org.dspace.eperson.Group	people who can perform workflow tasks are defined in EPerson Groups
org.dspace.core.Email	used to email messages to Group members and submitters

The default workflow system models the states of an Item in a state machine with 5 states (SUBMIT, STEP_1, STEP_2, STEP_3, ARCHIVE.) These are the three optional steps where the item can be viewed and corrected by different groups of people. Actually, it's more like 8 states, with STEP_1_POOL, STEP_2_POOL, and STEP_3_POOL. These pooled states are when items are waiting to enter the primary states. Optionally, you can also choose to enable the enhanced, Configurable Workflow(see page 250), if you wish to have more control over your workflow steps/states. (*Note: the remainder of this description relates to the traditional, default workflow. For more information on the Configurable Workflow option, visit* Configurable Workflow(see page 250).)

The WorkflowService is invoked by events. While an Item is being submitted, it is held by a WorkspaceItem. Calling the start() method in the WorkflowService converts a WorkspaceItem to a WorkflowItem, and begins processing the WorkflowItem's state. Since all three steps of the workflow are optional, if no steps are defined, then the Item is simply archived.

Workflows are set per Collection, and steps are defined by creating corresponding entries in the List named workflowGroup. If you wish the workflow to have a step 1, use the administration tools for Collections to create a workflow Group with members who you want to be able to view and approve the Item, and the workflowGroup[0] becomes set with the ID of that Group.

If a step is defined in a Collection's workflow, then the WorkflowItem's state is set to that step_POOL. This pooled state is the WorkflowItem waiting for an EPerson in that group to claim the step's task for that WorkflowItem. The WorkflowManager emails the members of that Group notifying them that there is a task to be performed (the text is defined in config/emails,) and when an EPerson goes to their 'My DSpace' page to claim the task, the WorkflowManager is invoked with a claim event, and the WorkflowItem's state advances from STEP_x_POOL to STEP_x (where x is the corresponding step.) The EPerson can also generate an 'unclaim' event, returning the WorkflowItem to the STEP_x_POOL.

Other events the WorkflowService handles are advance(), which advances the WorkflowItem to the next state. If there are no further states, then the WorkflowItem is removed, and the Item is then archived. An EPerson performing one of the tasks can reject the Item, which stops the workflow, rebuilds the WorkspaceItem for it and sends a rejection note to the submitter. More drastically, an abort() event is generated by the admin tools to cancel a workflow outright.

7.5.3.5 Administration Toolkit

The *org.dspace.administer* package contains some classes for administering a DSpace system that are not generally needed by most applications.

The *CreateAdministrator* class is a simple command-line tool, executed via `[dspace]/bin/dspace create-administrator`, that creates an administrator e-person with information entered from standard input. This is generally used only once when a DSpace system is initially installed, to create an initial administrator who can then use the Web administration UI to further set up the system. This script does not check for authorization, since it is typically run before there are any e-people to authorize! Since it must be run as a command-line tool on the server machine, generally this shouldn't cause a problem. A possibility is to have the script only operate when there are no e-people in the system already, though in general, someone with access to command-line scripts on your server is probably in a position to do what they want anyway!

The *DCType* class is similar to the *org.dspace.content.BitstreamFormat* class. It represents an entry in the Dublin Core type registry, that is, a particular element and qualifier, or unqualified element. It is in the *administer* package because it is only generally required when manipulating the registry itself. Elements and qualifiers are specified as literals in *org.dspace.content.Item* methods and the *org.dspace.content.Metadatum* class. Only administrators may modify the Dublin Core type registry.

The *org.dspace.administer.RegistryLoader* class contains methods for initializing the Dublin Core type registry and bitstream format registry with entries in an XML file. Typically this is executed via the command line during the build process (see *build.xml* in the source.) To see examples of the XML formats, see the files in *config/registries* in the source directory. There is no XML schema, they aren't validated strictly when loaded in.

7.5.3.6 E-person/Group Manager

DSpace keeps track of registered users with the *org.dspace.eperson.EPerson* class. The class has methods to create and manipulate an *EPerson* such as get and set methods for first and last names, email, and password. (Actually, there is no *getPassword()* method, an MD5 hash of the password is stored, and can only be verified with the *checkPassword()* method.) There are find methods to find an EPerson by email (which is assumed to be unique,) or to find all EPeople in the system.

The *EPerson* object should probably be reworked to allow for easy expansion; the current EPerson object tracks pretty much only what MIT was interested in tracking - first and last names, email, phone. The access methods are hardcoded and should probably be replaced with methods to access arbitrary name/value pairs for institutions that wish to customize what EPerson information is stored.

Groups are simply lists of *EPerson* objects. Other than membership, *Group* objects have only one other attribute: a name. Group names must be unique, so (for groups associated with workflows) we have adopted naming conventions where the role of the group is its name, such as *COLLECTION_100_ADD*. Groups add and remove EPerson objects with *addMember()* and *removeMember()* methods. One important thing to know about groups is that they store their membership in memory until the *update()* method is called - so when modifying a group's membership don't forget to invoke *update()* or your changes will be lost! Since group membership is used heavily by the authorization system a fast *isMember()* method is also provided.

Two specific groups are created when DSpace is installed: Administrator (which can bypass authorization) and Anonymous (which is assigned to all sessions that are not logged in). The code expects these groups to exist. They cannot be renamed or deleted.

Another kind of Group is also implemented in DSpace, special Groups. The *Context* object for each session carries around a List of Group IDs that the user is also a member of, currently the MITUser Group ID is added to the list of a user's special groups if certain IP address or certificate criteria are met.

7.5.3.7 Authorization

The primary classes are:

org.dspace.authorize.AuthorizeService	does all authorization, checking policies against Groups
org.dspace.authorize.ResourcePolicy	defines all allowable actions for an object
org.dspace.eperson.Group	all policies are defined in terms of EPerson Groups

The authorization system is based on the classic 'police state' model of security; no action is allowed unless it is expressed in a policy. The policies are attached to resources (hence the name *ResourcePolicy*,) and detail who can perform that action. The resource can be any of the DSpace object types, listed in *org.dspace.core.Constants* (*BITSTREAM*, *ITEM*, *COLLECTION*, etc.) The 'who' is made up of EPerson groups. The actions are also in *Constants.java* (*READ*, *WRITE*, *ADD*, etc.) The only non-obvious actions are *ADD* and *REMOVE*, which are authorizations for container objects. To be able to create an Item, you must have *ADD* permission in a Collection, which contains Items. (Communities, Collections, Items, and Bundles are all container objects.)

Currently most of the read policy checking is done with items, communities and collections are assumed to be openly readable, but items and their bitstreams are checked. Separate policy checks for items and their bitstreams enables policies that allow publicly readable items, but parts of their content may be restricted to certain groups.

Three new attributes have been introduced in the *ResourcePolicy* class as part of the DSpace Embargo[493] Contribution:

- *rpname*: resource policy name
- *rptype*: resource policy type
- *rpdescription*: resource policy description

While *rpname* and *rpdescription _are fields manageable by the users the _rptype* is a fields managed by the system. It represents a type that a resource policy can assume beteween the following:

- TYPE_SUBMISSION: all the policies added automatically during the submission process
- TYPE_WORKFLOW: all the policies added automatically during the workflow stage
- TYPE_CUSTOM: all the custom policies added by users
- TYPE_INHERITED: all the policies inherited by the *DSO* father.

An custom policy, created for the purpose of creating an embargo could look like:

```
policy_id: 4847
resource_type_id: 2
resource_id: 89
action_id: 0
eperson_id:
epersongroup_id: 0
start_date: 2013-01-01
end_date:
rpname: Embargo Policy
rpdescription:  Embargoed through 2012
rptype: TYPE_CUSTOM
```

The *AuthorizeService* class'
authorizeAction(Context, object, action) is the primary source of all authorization in the system. It gets a list of all of the ResourcePolicies in the system that match the object and action. It then iterates through the policies, extracting

493 https://wiki.lyrasis.org/display/DSDOC4x/Embargo

the EPerson Group from each policy, and checks to see if the EPersonID from the Context is a member of any of those groups. If all of the policies are queried and no permission is found, then an *AuthorizeException* is thrown. An *authorizeAction()* method is also supplied that returns a boolean for applications that require higher performance.

ResourcePolicies are very simple, and there are quite a lot of them. Each can only list a single group, a single action, and a single object. So each object will likely have several policies, and if multiple groups share permissions for actions on an object, each group will get its own policy. (It's a good thing they're small.)

Special Groups

All users are assumed to be part of the public group (ID=0.) DSpace admins (ID=1) are automatically part of all groups, much like super-users in the Unix OS. The Context object also carries around a List of special groups, which are also first checked for membership. These special groups are used at MIT to indicate membership in the MIT community, something that is very difficult to enumerate in the database! When a user logs in with an MIT certificate or with an MIT IP address, the login code adds this MIT user group to the user's Context.

Miscellaneous Authorization Notes

Where do items get their read policies? From the their collection's read policy. There once was a separate item read default policy in each collection, and perhaps there will be again since it appears that administrators are notoriously bad at defining collection's read policies. There is also code in place to enable policies that are timed, have a start and end date. However, the admin tools to enable these sorts of policies have not been written.

7.5.3.8 Handle Manager/Handle Plugin

The *org.dspace.handle* package contains two classes; *HandleService* is used to create and look up Handles, and *HandlePlugin* is used to expose and resolve DSpace Handles for the outside world via the CNRI Handle Server code.

Handles are stored internally in the *handle* database table in the form:

1721.123/4567

Typically when they are used outside of the system they are displayed in either URI or "URL proxy" forms:

```
hdl:1721.123/4567
http://hdl.handle.net/1721.123/4567
```

It is the responsibility of the caller to extract the basic form from whichever displayed form is used.

The *handle* table maps these Handles to resource type/resource ID pairs, where resource type is a value from *org.dspace.core.Constants* and resource ID is the internal identifier (database primary key) of the object. This allows Handles to be assigned to any type of object in the system, though as explained in the functional overview, only communities, collections and items are presently assigned Handles.

HandleService contains static methods for:

- Creating a Handle
- Finding the Handle for a *DSpaceObject*, though this is usually only invoked by the object itself, since *DSpaceObject* has a *getHandle* method
- Retrieving the *DSpaceObject* identified by a particular Handle
- Obtaining displayable forms of the Handle (URI or "proxy URL").
  HandlePlugin is a simple implementation of the Handle Server's *net.handle.hdllib.HandleStorage* interface. It only implements the basic Handle retrieval methods, which get information from the *handle* database table. The CNRI Handle Server is configured to use this plug-in via its *config.dct* file.

Note that since the Handle server runs as a separate JVM to the DSpace Web applications, it uses a separate 'Log4J' configuration, since Log4J does not support multiple JVMs using the same daily rolling logs. This alternative configuration is located at `[dspace]/config/log4j-handle-plugin.properties`. The `[dspace]/bin/start-handle-server` script passes in the appropriate command line parameters so that the Handle server uses this configuration.

> In additional to Handles, DSpace also provides basic support for DOIs (Digital Object Identifiers). For more information visit DOI Digital Object Identifier(see page 296).

7.5.3.9 Search

DSpace's search code is a simple, configurable API which currently wraps Apache Solr. See Discovery(see page 386) for more information on how to customize the default search settings, etc.

Harvesting API

The *org.dspace.search* package also provides a 'harvesting' API. This allows callers to extract information about items modified within a particular timeframe, and within a particular scope (all of DSpace, or a community or collection.) Currently this is used by the Open Archives Initiative metadata harvesting protocol application, and the e-mail subscription code.

The *Harvest.harvest* is invoked with the required scope and start and end dates. Either date can be omitted. The dates should be in the ISO8601, UTC time zone format used elsewhere in the DSpace system.

HarvestedItemInfo objects are returned. These objects are simple containers with basic information about the items falling within the given scope and date range. Depending on parameters passed to the *harvest* method, the *containers* and *item* fields may have been filled out with the IDs of communities and collections containing an item, and the corresponding *Item* object respectively. Electing not to have these fields filled out means the harvest operation executes considerable faster.

In case it is required, *Harvest* also offers a method for creating a single *HarvestedItemInfo* object, which might make things easier for the caller.

7.5.3.10 Browse API

The browse API uses the same underlying technology as the Search API (Apache Solr, see also Discovery(see page 386)). It maintains indexes of dates, authors, titles and subjects, and allows callers to extract parts of these:

- **Title**: Values of the Dublin Core element **title** (unqualified) are indexed. These are sorted in a case-insensitive fashion, with any leading article removed. For example: "The DSpace System" would appear under 'D' rather than 'T'.
- **Author**: Values of the **contributor** (any qualifier or unqualified) element are indexed. Since *contributor* values typically are in the form 'last name, first name', a simple case-insensitive alphanumeric sort is used which orders authors in last name order. Note that this is an index of *authors*, and not *items by author*. If four items have the same author, that author will appear in the index only once. Hence, the index of authors may be greater or smaller than the index of titles; items often have more than one author, though the same author may have authored several items. The author indexing in the browse API does have limitations:
    - Ideally, a name that appears as an author for more than one item would appear in the author index only once. For example, 'Doe, John' may be the author of tens of items. However, in practice, author's names often appear in slightly differently forms, for example:

```
Doe, John
Doe, John Stewart
Doe, John S.
```

Currently, the above three names would all appear as separate entries in the author index even though they may refer to the same author. In order for an author of several papers to be correctly appear once in the index, each item must specify *exactly* the same form of their name, which doesn't always happen in practice.

- Another issue is that two authors may have the same name, even within a single institution. If this is the case they may appear as one author in the index. These issues are typically resolved in libraries with *authority control records*, in which are kept a 'preferred' form of the author's name, with extra information (such as date of birth/death) in order to distinguish between authors of the same name. Maintaining such records is a huge task with many issues, particularly when metadata is received from faculty directly rather than trained library catalogers.

- **Date of Issue**: Items are indexed by date of issue. This may be different from the date that an item appeared in DSpace; many items may have been originally published elsewhere beforehand. The Dublin Core field used is **date.issued**. The ordering of this index may be reversed so 'earliest first' and 'most recent first' orderings are possible. Note that the index is of *items by date*, as opposed to an index of *dates*. If 30 items have the same issue date (say 2002), then those 30 items all appear in the index adjacent to each other, as opposed to a single 2002 entry. Since dates in DSpace Dublin Core are in ISO8601, all in the UTC time zone, a simple alphanumeric sort is sufficient to sort by date, including dealing with varying granularities of date reasonably. For example:

```
2001-12-10
2002
2002-04
2002-04-05
2002-04-09T15:34:12Z
2002-04-09T19:21:12Z
2002-04-10
```

- **Date Accessioned**: In order to determine which items most recently appeared, rather than using the date of issue, an item's accession date is used. This is the Dublin Core field **date.accessioned**. In other aspects this index is identical to the date of issue index.
- **Items by a Particular Author**: The browse API can perform is to extract items by a particular author. They do not have to be primary author of an item for that item to be extracted. You can specify a scope, too; that is, you can ask for items by author X in collection Y, for example.This particular flavor of browse is slightly simpler than the others. You cannot presently specify a particular subset of results to be returned. The API call will simply return all of the items by a particular author within a certain scope. Note that the author of the item must *exactly* match the author passed in to the API; see the explanation about the caveats of the author index browsing to see why this is the case.
- **Subject**: Values of the Dublin Core element **subject** (both unqualified and with any qualifier) are indexed. These are sorted in a case-insensitive fashion.

Using the API

The API is generally invoked by creating a *BrowseScope* object, and setting the parameters for which particular part of an index you want to extract. This is then passed to the relevant *Browse* method call, which returns a *BrowseInfo* object which contains the results of the operation. The parameters set in the *BrowseScope* object are:

- How many entries from the index you want
- Whether you only want entries from a particular community or collection, or from the whole of DSpace

- Which part of the index to start from (called the *focus* of the browse). If you don't specify this, the start of the index is used
- How many entries to include before the *focus* entry

To illustrate, here is an example:

- We want **7** entries in total
- We want entries from collection *x*
- We want the focus to be 'Really'
- We want **2** entries included before the focus.

The results of invoking *Browse.getItemsByTitle* with the above parameters might look like this:

```
Rabble-Rousing Rabbis From Sardinia
        Reality TV: Love It or Hate It?
FOCUS>  The Really Exciting Research Video
        Recreational Housework Addicts: Please Visit My House
        Regional Television Variation Studies
        Revenue Streams
        Ridiculous Example Titles:  I'm Out of Ideas
```

Note that in the case of title and date browses, *Item* objects are returned as opposed to actual titles. In these cases, you can specify the 'focus' to be a specific item, or a partial or full literal value. In the case of a literal value, if no entry in the index matches exactly, the closest match is used as the focus. It's quite reasonable to specify a focus of a single letter, for example.

Being able to specify a specific item to start at is particularly important with dates, since many items may have the save issue date. Say 30 items in a collection have the issue date 2002. To be able to page through the index 20 items at a time, you need to be able to specify exactly which item's 2002 is the focus of the browse, otherwise each time you invoked the browse code, the results would start at the first item with the issue date 2002.

Author browses return *String* objects with the actual author names. You can only specify the focus as a full or partial literal *String*.

Another important point to note is that presently, the browse indexes contain metadata for all items in the main archive, regardless of authorization policies. This means that all items in the archive will appear to all users when browsing. Of course, should the user attempt to access a non-public item, the usual authorization mechanism will apply. Whether this approach is ideal is under review; implementing the browse API such that the results retrieved reflect a user's level of authorization may be possible, but rather tricky.

7.5.3.11 Checksum checker

Checksum checker is used to verify every item within DSpace. While DSpace calculates and records the checksum of every file submitted to it, the checker can determine whether the file has been changed. The idea being that the earlier you can identify a file has changed, the more likely you would be able to record it (assuming it was not a wanted change).

`org.dspace.checker.CheckerCommand` class, is the class for the checksum checker tool, which calculates checksums for each bitstream whose ID is in the *most_recent_checksum* table, and compares it against the last calculated checksum for that bitstream.

7.5.3.12 OpenSearch Support

DSpace is able to support OpenSearch. For those not acquainted with the standard, a very brief introduction, with emphasis on what possibilities it holds for current use and future development.

OpenSearch is a small set of conventions and documents for describing and using 'search engines', meaning any service that returns a set of results for a query. It is nearly ubiquitous, but also nearly invisible, in modern web sites with search capability. If you look at the page source of Wikipedia, Facebook, CNN, etc you will find buried a link element declaring OpenSearch support. It is very much a lowest-common-denominator abstraction (think Google box), but does provide a means to extend its expressive power. This first implementation for DSpace supports *none* of these extensions, many of which are of potential value, so it should be regarded as a foundation, not a finished solution. So the short answer is that DSpace appears as a 'search-engine' to OpenSearch-aware software.

Another way to look at OpenSearch is as a RESTful web service for search, very much like SRW/U, but considerably simpler. This comparative loss of power is offset by the fact that it is widely supported by web tools and players: browsers understand it, as do large metasearch tools.

How Can It Be Used

- Browser IntegrationMany recent browsers (IE7+, FF2+) can detect, or 'autodiscover', links to the document describing the search engine. Thus you can easily add your or other DSpace instances to the drop-down list of search engines in your browser. This list typically appears in the upper right corner of the browser, with a search box. In Firefox, for example, when you visit a site supporting OpenSearch, the color of the drop-down list widget changes color, and if you open it to show the list of search engines, you are offered an opportunity to add the site to the list. IE works nearly the same way but instead labels the web sites 'search providers'. When you select a DSpace instance as the search engine and enter a search, you are simply sent to the regular search results page of the instance.
- Flexible, interesting RSS FeedsBecause one of the formats that OpenSearch specifies for its results is RSS (or Atom), you can turn any search query into an RSS feed. So if there are keywords highly discriminative of content in a collection or repository, these can be turned into a URL that a feed reader can subscribe to. Taken to the extreme, one could take any search a user makes, and dynamically compose an RSS feed URL for it in the page of returned results. To see an example, if you have a DSpace with OpenSearch enabled, try:

  ```
  http://dspace.mysite.edu/open-search/?query=<your query>
  ```

  The default format returned is Atom 1.0, so you should see an Atom document containing your search results.
- You can extend the syntax with a few other parameters, as follows:

Parameter	Values
format	atom, rss, html
scope	handle of a collection or community to restrict the search to
rpp	number indicating the number of results per page (i.e. per request)
start	number of page to start with (if paginating results)

Parameter	Values
sort_by	number indicating sorting criteria (same as DSpace advanced search values

Multiple parameters may be specified on the query string, using the "&" character as the delimiter, e.g.:

```
http://dspace.mysite.edu/open-search/?query=<your query>&format=rss&scope=123456789/1
```

- Cheap metasearchSearch aggregators like A9 (Amazon) recognize OpenSearch-compliant providers, and so can be added to metasearch sets using their UIs. Then you site can be used to aggregate search results with others.

Configuration is through the `dspace.cfg` file. See OpenSearch Support[494] for more details.

7.5.3.13 Embargo Support

What is an Embargo?

An embargo is a temporary access restriction placed on content, commencing at time of accession. It's scope or duration may vary, but the fact that it eventually expires is what distinguishes it from other content restrictions. For example, it is not unusual for content destined for DSpace to come with permanent restrictions on use or access based on license-driven or other IP-based requirements that limit access to institutionally affiliated users. Restrictions such as these are imposed and managed using standard administrative tools in DSpace, typically by attaching specific policies to Items or Collections, Bitstreams, etc. The embargo functionally introduced in 1.6, however, includes tools to automate the imposition and removal of restrictions in managed timeframes.

Embargo Model and Life-Cycle

Functionally, the embargo system allows you to attach 'terms' to an item before it is placed into the repository, which express how the embargo should be applied. What do 'we mean by terms' here? They are really any expression that the system is capable of turning into (1) the time the embargo expires, and (2) a concrete set of access restrictions. Some examples:
"2020-09-12" - an absolute date (i.e. the date embargo will be lifted)"6 months" - a time relative to when the item is accessioned"forever" - an indefinite, or open-ended embargo"local only until 2015" - both a time and an exception (public has no access until 2015, local users OK immediately)"Nature Publishing Group standard" - look-up to a policy somewhere (typically 6 months)
These terms are 'interpreted' by the embargo system to yield a specific date on which the embargo can be removed or 'lifted', and a specific set of access policies. Obviously, some terms are easier to interpret than others (the absolute date really requires none at all), and the 'default' embargo logic understands only the most basic terms (the first and third examples above). But as we will see below, the embargo system provides you with the ability to add in your own 'interpreters' to cope with any terms expressions you wish to have. This date that is the result of the interpretation is stored with the item and the embargo system detects when that date has passed, and removes the embargo ("lifts it"), so the item bitstreams become available. Here is a more detailed life-cycle for an embargoed item:

1. **Terms Assignment.** The first step in placing an embargo on an item is to attach (assign) 'terms' to it. If these terms are missing, no embargo will be imposed. As we will see below, terms are carried in a configurable

494 https://wiki.lyrasis.org/display/DSDOC4x/Configuration+Reference#ConfigurationReference-OpenSearchSupport

DSpace metadata field, so assigning terms just means assigning a value to a metadata field. This can be done in a web submission user interface form, in a SWORD deposit package, a batch import, etc. - anywhere metadata is passed to DSpace. The terms are not immediately acted upon, and may be revised, corrected, removed, etc, up until the next stage of the life-cycle. Thus a submitter could enter one value, and a collection editor replace it, and only the last value will be used. Since metadata fields are multivalued, theoretically there can be multiple terms values, but in the default implementation only one is recognized.

2. **Terms interpretation/imposition.** In DSpace terminology, when an item has exited the last of any workflow steps (or if none have been defined for it), it is said to be 'installed' into the repository. At this precise time, the 'interpretation' of the terms occurs, and a computed 'lift date' is assigned, which like the terms is recorded in a configurable metadata field. It is important to understand that this interpretation happens only once, (just like the installation), and cannot be revisited later. Thus, although an administrator can assign a new value to the metadata field holding the terms after the item has been installed, this will have no effect on the embargo, whose 'force' now resides entirely in the 'lift date' value. For this reason, you cannot embargo content already in your repository (at least using standard tools). The other action taken at installation time is the actual imposition of the embargo. The default behavior here is simply to remove the read policies on all the bundles and bitstreams except for the "LICENSE" or "METADATA" bundles. See the section on *Extending Embargo Functionality* for how to alter this behavior. Also note that since these policy changes occur before installation, there is no time during which embargoed content is 'exposed' (accessible by non-administrators). The terms interpretation and imposition together are called 'setting' the embargo, and the component that performs them both is called the embargo 'setter'.
3. **Embargo Period.** After an embargoed item has been installed, the policy restrictions remain in effect until removed. This is not an automatic process, however: a 'lifter' must be run periodically to look for items whose 'lift date' is past. Note that this means the effective removal of an embargo is **not** the lift date, but the earliest date after the lift date that the lifter is run. Typically, a nightly cron-scheduled invocation of the lifter is more than adequate, given the granularity of embargo terms. Also note that during the embargo period, all metadata of the item remains visible. This default behavior can be changed. One final point to note is that the 'lift date', although it was computed and assigned during the previous stage, is in the end a regular metadata field. That means, if there are extraordinary circumstances that require an administrator (or collection editor, anyone with edit permissions on metadata) to change the lift date, they can do so. Thus, they can 'revise' the lift date without reference to the original terms. This date will be checked the next time the 'lifter' is run. One could immediately lift the embargo by setting the lift date to the current day, or change it to 'forever' to indefinitely postpone lifting.
4. **Embargo Lift.** When the lifter discovers an item whose lift date is in the past, it removes (lifts) the embargo. The default behavior of the lifter is to add the resource policies *that would have been added* had the embargo not been imposed. That is, it replicates the standard DSpace behavior, in which an item inherits it's policies from its owning collection. As with all other parts of the embargo system, you may replace or extend the default behavior of the lifter (see section V. below). You may wish, e.g. to send an email to an administrator or other interested parties, when an embargoed item becomes available.
5. **Post Embargo.** After the embargo has been lifted, the item ceases to respond to any of the embargo life-cycle events. The values of the metadata fields reflect essentially historical or provenance values. With the exception of the additional metadata fields, they are indistinguishable from items that were never subject to embargo.

> **More Embargo Details**
>
> More details on Embargo configuration, including specific examples can be found in the Embargo[495] section of the documentation.

495 https://wiki.lyrasis.org/display/DSDOC4x/Embargo

7.5.4 DSpace Services Framework

The DSpace Services Framework is a backporting of the DSpace 2.0 Development Group's work in creating a reasonable and abstractable "Core Services" layer for DSpace components to operate within. The Services Framework represents a "best practice" for new DSpace architecture and implementation of extensions to the DSpace application. DSpace Services are best described as a "Simple Registry" where plugins can be "looked up" or located. The DS2 (DSpace 2.0[496]) core services are the main services that make up a DS2 system. These includes services for things like user and permissions management and storage and caching. These services can be used by any developer writing DS2 plugins (e.g. statistics), providers (e.g. authentication), or user interfaces.

7.5.4.1 Architectural Overview

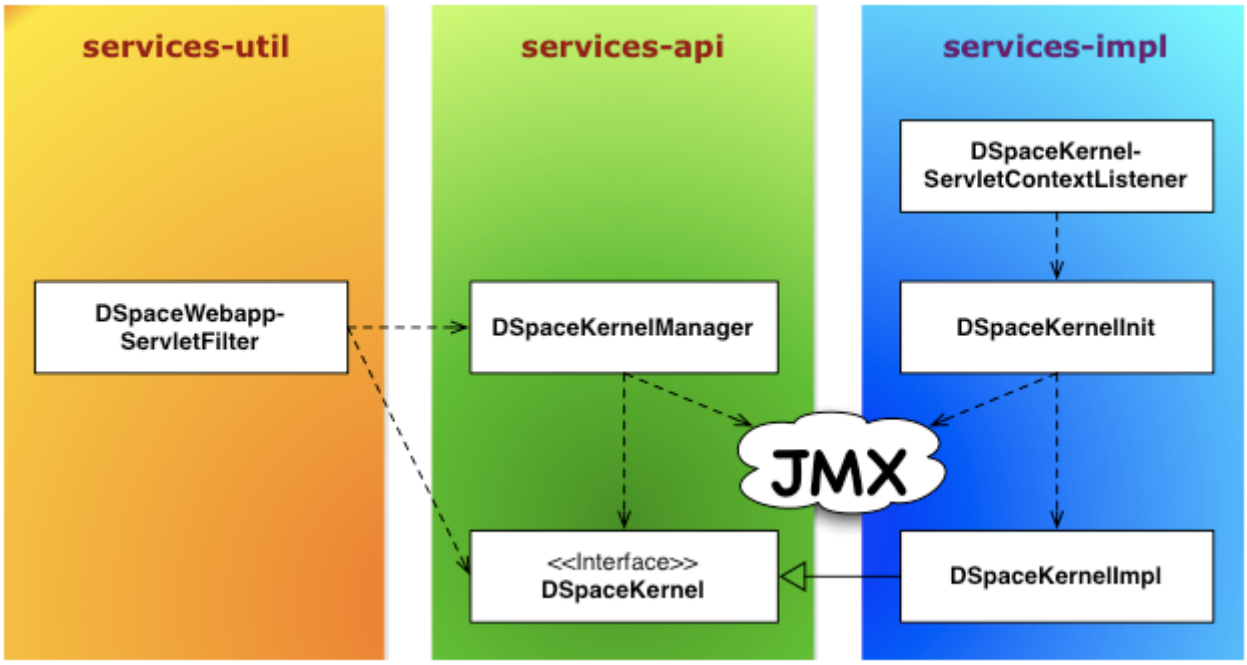
DSpace Kernel

The DSpace Kernel manages the start up and access services in the DSpace Services framework. It is meant to allow for a simple way to control the core parts of DSpace and allow for flexible ways to startup the kernel. For example, the kernel can be run inside a single webapp along with a frontend UI or it can be started as part of the servlet container so that multiple webapps can use a single kernel (this increases speed and efficiency). The kernel is also designed to happily allow multiple kernels to run in a single servlet container using identifier keys.

Kernel registration

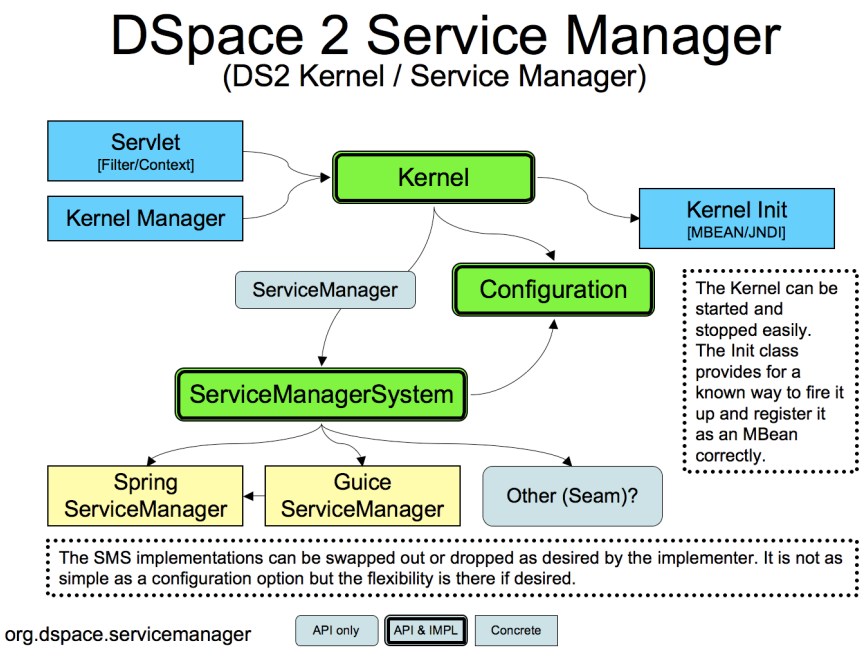
The kernel will automatically register itself as an MBean when it starts up so that it can be managed via JMX[497]. It allows startup and shutdown and provides direct access to the ServiceManager and the ConfigurationService. All the other core services can be retrieved from the ServiceManager by their APIs.

496 http://wiki.dspace.org/index.php/DSpace_2.0
497 http://www.oracle.com/technetwork/java/javase/tech/docs-jsp-135989.html



Service Manager

The ServiceManager abstracts the concepts of service lookups and lifecycle control. It also manages the configuration of services by allowing properties to be pushed into the services as they start up (mostly from the ConfigurationService). The ServiceManagerSystem abstraction allows the DSpace ServiceManager to use different systems to manage its services. The current implementations include Spring and Guice. This allows DSpace 2 to have very little service management code but still be flexible and not tied to specific technology. Developers who are comfortable with those technologies can consume the services from a parent Spring ApplicationContext or a parent Guice Module. The abstraction also means that we can replace Spring/Guice or add other dependency injection systems later without requiring developers to change their code. The interface provides simple methods for looking up services by interface type for developers who do not want to have to use or learn a dependency injection system or are using one which is not currently supported.



The DS2 kernel is compact so it can be completely started up in a unit test (technically integration test) environment. (This is how we test the kernel and core services currently). This allows developers to execute code against a fully functional kernel while developing and then deploy their code with high confidence.

7.5.4.2 Basic Usage

To use the Framework you must begin by instantiating and starting a DSpaceKernel. The kernel will give you references to the ServiceManager and the ConfigurationService. The ServiceManager can be used to get references to other services and to register services which are not part of the core set.

Access to the kernel is provided via the Kernel Manager through the DSpace object, which will locate the kernel object and allow it to be used.

Standalone Applications

For standalone applications, access to the kernel is provided via the Kernel Manager and the DSpace object which will locate the kernel object and allow it to be used.

```
/* Instantiate the Utility Class */
DSpace dspace = new DSpace();


/* Access get the Service Manager by convenience method */
ServiceManager manager = dspace.getServiceManager();


/* Or access by convenience method for core services */
EventService service = dspace.getEventService();
```

The DSpace launcher (

```
bin/dspace
```

) initializes a kernel before dispatching to the selected command.

Application Frameworks (Spring, Guice, etc.)

Similar to Standalone Applications(see page 683), but you can use your framework to instantiate an `org.dspace.utils.DSpace` object.

```
    <bean id="dspace" class="org.dspace.utils.DSpace"/>
```

Web Applications

In web applications, the kernel can be started and accessed through the use of Servlet Filter/ContextListeners which are provided as part of the DSpace 2 utilities. Developers don't need to understand what is going on behind the scenes and can simply write their applications and package them as webapps and take advantage of the services which are offered by DSpace 2.

7.5.4.3 Providers and Plugins

For developers (how we are trying to make your lives easier): The DS2 ServiceManager supports a plugin/provider system which is runtime hot-swappable. The implementor can register any service/provider bean or class with the DS2 kernel ServiceManager. The ServiceManager will manage the lifecycle of beans (if desired) and will instantiate and manage the lifecycle of any classes it is given. This can be done at any time and does not have to be done during Kernel startup. This allows providers to be swapped out at runtime without disrupting the service if desired. The goal of this system is to allow DS2 to be extended without requiring any changes to the core codebase or a rebuild of the code code.

Activators

Developers can provide an activator to allow the system to startup their service or provider. It is a simple interface with 2 methods which are called by the ServiceManager to startup the provider(s) and later to shut them down. These simply allow a developer to run some arbitrary code in order to create and register services if desired. It is the method provided to add plugins directly to the system via configuration as the activators are just listed in the configuration file and the system starts them up in the order it finds them.

Provider Stacks

Utilities are provided to assist with stacking and ordering providers. Ordering is handled via a priority number such that 1 is the highest priority and something like 10 would be lower. 0 indicates that priority is not important for this service and can be used to ensure the provider is placed at or near the end without having to set some arbitrarily high number.

7.5.4.4 Core Services

The core services are all behind APIs so that they can be reimplemented without affecting developers who are using the services. Most of the services have plugin/provider points so that customizations can be added into the system without touching the core services code. For example, let's say a deployer has a specialized authentication system and wants to manage the authentication calls which come into the system. The implementor can simply implement

an AuthenticationProvider and then register it with the DS2 kernel's ServiceManager. This can be done at any time and does not have to be done during Kernel startup. This allows providers to be swapped out at runtime without disrupting the DS2 service if desired. It can also speed up development by allowing quick hot redeploys of code during development.

Caching Service

Provides for a centralized way to handle caching in the system and thus a single point for configuration and control over all caches in the system. Provider and plugin developers are strongly encouraged to use this rather than implementing their own caching. The caching service has the concept of scopes so even storing data in maps or lists is discouraged unless there are good reasons to do so.

Configuration Service

The ConfigurationService controls the external and internal configuration of DSpace 2. It reads Properties files when the kernel starts up and merges them with any dynamic configuration data which is available from the services. This service allows settings to be updated as the system is running, and also defines listeners which allow services to know when their configuration settings have changed and take action if desired. It is the central point to access and manage all the configuration settings in DSpace 2.

Manages the configuration of the DSpace 2 system. Can be used to manage configuration for providers and plugins also.

EventService

Handles events and provides access to listeners for consumption of events.

RequestService

In DS2 a request is an atomic transaction in the system. It is likely to be an HTTP request in many cases but it does not have to be. This service provides the core services with a way to manage atomic transactions so that when a request comes in which requires multiple things to happen they can either all succeed or all fail without each service attempting to manage this independently. In a nutshell this simply allows identification of the current request and the ability to discover if it succeeded or failed when it ends. Nothing in the system will enforce usage of the service, but we encourage developers who are interacting with the system to make use of this service so they know if the request they are participating in with has succeeded or failed and can take appropriate actions.

SessionService

In DS2 a session is like an HttpSession (and generally is actually one) so this service is here to allow developers to find information about the current session and to access information in it. The session identifies the current user (if authenticated) so it also serves as a way to track user sessions. Since we use HttpSession directly it is easy to mirror sessions across multiple servers in order to allow for no-interruption failover for users when servers go offline.

7.5.4.5 Examples

Configuring Event Listeners

Event Listeners can be created by overriding the the EventListener interface:

In Spring:

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

    <bean id="dspace" class="org.dspace.utils.DSpace"/>

    <bean id="dspace.eventService"
          factory-bean="dspace"
          factory-method="getEventService"/>

    <bean class="org.my.EventListener">
         <property name="eventService" >
            <ref bean="dspace.eventService"/>
        </property>
    </bean>
</beans>
```

(org.my.EventListener will need to register itself with the EventService, for which it is passed a reference to that service via the eventService property.)

or in Java:

```
DSpace dspace = new DSpace();

EventService eventService = dspace.getEventService();

EventListener listener = new org.my.EventListener();
eventService.registerEventListener(listener);
```

(This registers the listener externally – the listener code assumes it is registered.)

7.5.4.6 Tutorials

Several tutorials on Spring / DSpace Services are available:

- DSpace Spring Services Tutorial[498]
- The TAO of DSpace Services[499]

7.5.5 Storage Layer

In this section, we explain the storage layer: the database structure, maintenance, and the bitstream store and configurations. The bitstream store, also known as assetstore or bitstore, holds the uploaded, ingested, or generated files (documents, images, audio, video, datasets, ...), where as the database holds all of the metadata, organization, and permissions of content.

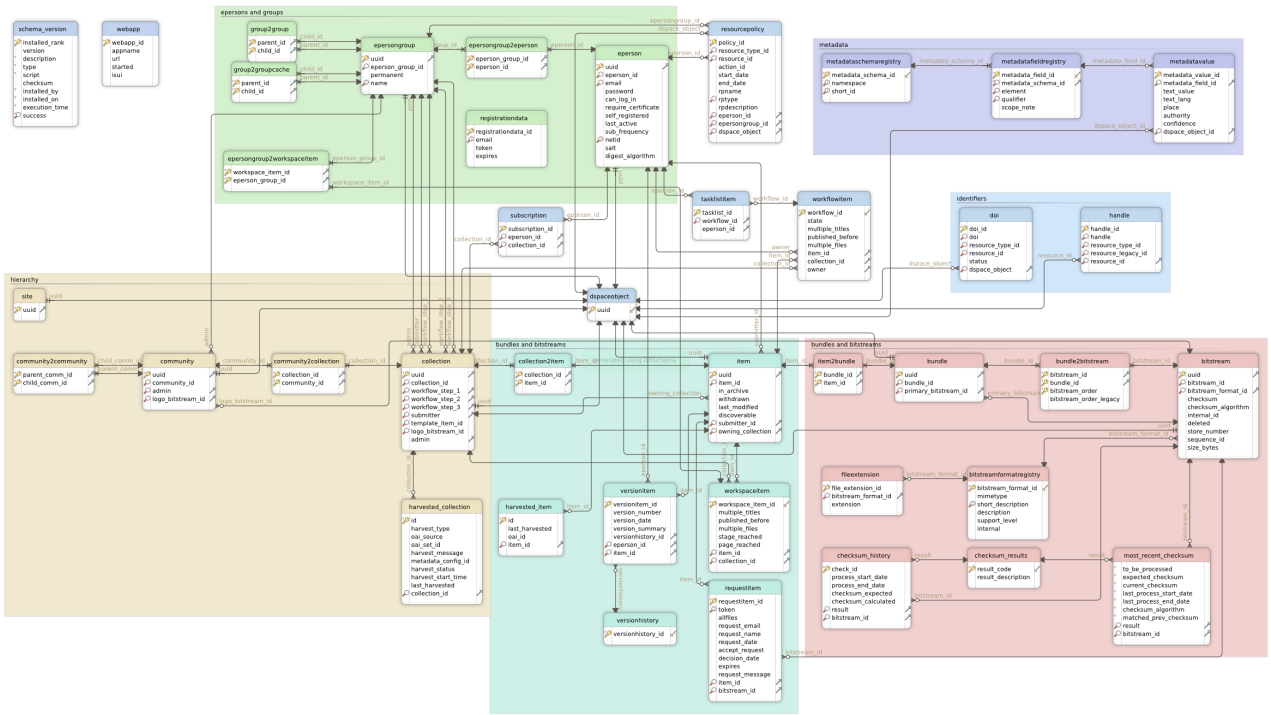
- RDBMS / Database Structure(see page 687)

498 https://wiki.lyrasis.org/display/DSPACE/DSpace+Spring+Services+Tutorial
499 https://wiki.lyrasis.org/display/DSPACE/The+TAO+of+DSpace+Services

7.5.5.1 RDBMS / Database Structure

DSpace uses a relational database to store all information about the organization of content, metadata about the content, information about e-people and authorization, and the state of currently-running workflows.

DSpace 6 database schema (Postgres). Right-click the image and choose "Save as" to save in full resolution. Instructions on updating this schema diagram are in How to update database schema diagram[500].



- DSpace uses FlywayDB[501] to perform automated database initialization and upgrades. Flyway's role is to initialize the database tables (and default content) prior to Hibernate initialization.
  - The `org.dspace.storage.rdbms.DatabaseUtils` class manages all Flyway API calls, and executes the SQL migrations under the `org.dspace.storage.rdbms.sqlmigration` package and the Java migrations under the `org.dspace.storage.rdbms.migration` package.

500 https://wiki.lyrasis.org/display/DSDOC/How+to+update+database+schema+diagram
501 https://flywaydb.org/

    - Once all database migrations have run, a series of Flyway Callbacks[502] are triggered to initialize the (empty) database with required default content. For example, callbacks exist for adding default DSpace Groups (`GroupServiceInitializer`), default Metadata & Format Registries (`DatabaseRegistryUpdater`), and the default Site object (`SiteServiceInitializer`). All Callbacks are under the `org.dspace.storage.rdbms` package.
    - While Flyway is automatically initialized and executed during startup, various Database Utilities[503] are also available on the command line. These utilities allow you to manually trigger database upgrades or check the status of your database.
- DSpace uses Hibernate ORM[504] as the object relational mapping layer between the DSpace database and the DSpace code.
    - The main Hibernate configuration can be found at `[dspace]/config/hibernate.cfg.xml`
    - Hibernate initialization is triggered via Spring (beans) defined `[dspace]/config/spring/api/core-hibernate.xml`. This Spring configuration pulls in some settings from DSpace Configuration[505], namely all Database (db.*) settings defined there.
    - All DSpace Object Classes provide a DAO (Data Access Object) implementation class that extends a GenericDAO interface defined in `org.dspace.core.GenericDAO` class. The default (abstract) implementation is in `org.dspace.core.AbstractHibernateDAO` class.
    - The DSpace Context object (`org.dspace.core.Context`) provides access to the configured `org.dspace.core.DBConnection` (Database Connection), which is HibernateDBConnection by default. The `org.dspace.core.HibernateDBConnection` class provides access the the Hibernate Session interface (org.hibernate.Session) and its Transactions.
        - Each Hibernate Session opens a single database connection when it is created, and holds onto it until the Session is closed. A Session may consist of one or more Transactions. Sessions are NOT thread-safe (so individual objects cannot be shared between threads).
        - Hibernate will intelligently cache objects in the current Hibernate Session (on object access), allowing for optimized performance.
        - DSpace provides methods on the Context object to specifically remove (`Context.uncacheEntity()`) or reload (`Context.reloadEntity()`) objects within Hibernate's Session cache.
        - DSpace also provides special Context object "modes" to optimize Hibernate performance for read-only access (`Mode.READ_ONLY`) or batch processing (`Mode.BATCH_EDIT`). These modes can be specified when constructing a new Context object.

Most of the functionality that DSpace uses can be offered by any standard SQL database that supports transactions. However, at this time, DSpace only provides Flyway migration scripts for PostgreSQL[506] and Oracle[507] (and has only been tested with those database backends). Additional database backends should be possible, but would minimally require creating custom Flyway migration scripts for that database backend.

502 https://flywaydb.org/documentation/callbacks.html
503 https://wiki.lyrasis.org/display/DSDOC6x/Database+Utilities
504 http://hibernate.org/orm/
505 https://wiki.lyrasis.org/display/DSDOC6x/Configuration+Reference
506 http://www.postgresql.org/
507 http://www.oracle.com/database/

Maintenance and Backup

When using PostgreSQL, it's a good idea to perform regular 'vacuuming' of the database to optimize performance. By default, PostgreSQL performs automatic vacuuming[508] on your behalf. However, if you have this feature disabled, then we recommend scheduling the `vacuumdb`[509] command to run on a regular basis.

```
# clean up the database nightly
40 2 * * * /usr/local/pgsql/bin/vacuumdb --analyze dspace > /dev/null 2>&1
```

Backups: The DSpace database can be backed up and restored using usual PostgreSQL Backup and Restore[510] methods, for example with `pg_dump`[511] and `psql`[512]. However when restoring a database, you will need to perform these additional steps:

- After restoring a backup, you will need to reset the primary key generation sequences so that they do not produce already-used primary keys. Do this by executing the SQL in `[dspace]/etc/postgres/update-sequences.sql`, for example with:

  ```
  psql -U dspace -f [dspace]/etc/update-sequences.sql
  ```

Configuring the RDBMS Component

The database manager is configured with the following properties in `dspace.cfg`:

`db.url`	The JDBC URL to use for accessing the database. This should not point to a connection pool, since DSpace already implements a connection pool.
`db.driver`	JDBC driver class name. Since presently, DSpace uses PostgreSQL-specific features, this should be *org.postgresql.Driver*.
`db.username`	Username to use when accessing the database.
`db.password`	Corresponding password ot use when accessing the database.

Custom RDBMS tables, columns or views

When at all possible, we recommend creating custom database tables or views within a *separate schema* from the DSpace database tables. Since the DSpace database is initialized and upgraded automatically using Flyway DB[513], the upgrade process may stumble or throw errors if you've directly modified the DSpace database schema, views or

508 http://www.postgresql.org/docs/current/static/runtime-config-autovacuum.html
509 http://www.postgresql.org/docs/current/static/app-vacuumdb.html
510 http://www.postgresql.org/docs/current/static/backup.html
511 http://www.postgresql.org/docs/current/static/app-pgdump.html
512 http://www.postgresql.org/docs/current/static/app-psql.html
513 http://flywaydb.org/

tables. Flyway itself assumes it has full control over the DSpace database schema, and it is not "smart" enough to know what to do when it encounters a locally customized database.

That being said, if you absolutely need to customize your database tables, columns or views, it is possible to create *custom Flyway migration scripts* , which should make your customizations easier to manage in future upgrades. (Keep in mind though, that you may still need to maintain/update your custom Flyway migration scripts if they ever conflict directly with future DSpace database changes. The only way to "future proof" your local database changes is to try and make them as independent as possible, and avoid directly modifying the DSpace database schema as much as possible.)

If you wish to add custom Flyway migrations, they may be added to the following locations:

- Custom Flyway SQL migrations may be added anywhere under the `org.dspace.storage.rdbms.sqlmigration` package (e.g. `[src]/dspace-api/src/main/resources/org/dspace/storage/rdbms/sqlmigration` or subdirectories)
- Custom Flyway Java migrations may be added anywhere under the `org.dspace.storage.rdbms.migration` package (e.g. `[src]/dspace-api/src/main/java/org/dspace/storage/rdbms/migration/` or subdirectories)
- Additionally, for backwards support, custom SQL migrations may also be placed in the `[dspace]/etc/[db-type]/` folder (e.g. `[dspace]/etc/postgres/` for a PostgreSQL specific migration script)

Adding Flyway migrations to any of the above location will cause Flyway to auto-discover the migration. It will be run in the order in which it is named. Our DSpace Flyway script naming convention follows Flyway best practices and is as follows:

- SQL script names: `V[version]_[date]__[description].sql`
  - E.g. `V5.0_2014.09.26__DS-1582_Metadata_For_All_Objects.sql` is a SQL migration script created for DSpace 5.x (`V5.0`) on Sept 26, 2014 (`2014_09_24`). Its purpose was to fulfill the needs of ticket DS-1582, which was to migrate the database in order to support adding metadata on all objects.
  - More examples can be found under the `org.dspace.storage.rdbms.sqlmigration` package
- Java migration script naming convention: `V[version]_[date]__[description].java`
  - E.g. `V5_0_2014_09_25__DS_1582_Metadata_For_All_Objects_drop_constraint.java` is a Java migration created for DSpace 5.x (`V5_0`) on Sept 25, 2014 (`2014_09_25`). Its purpose was to fulfill the needs of ticket DS-1582, specifically to drop a few constraints.
  - More examples can be found under the `org.dspace.storage.rdbms.migration` package
- Flyway will execute migrations in order, based on their Version and Date. So, V1.x (or V1_x) scripts are executed first, followed by V3.0 (or V3_0), etc. If two migrations have the same version number, the date is used to determine ordering (earlier dates are run first).

7.5.5.2 Bitstream Store

DSpace offers two means for storing content.

1. Storage in a mounted file system on the server (DSBitStore)
2. Storage using AWS S3 (Simple Storage Service), (S3BitStore).

Both are achieved using a simple, lightweight BitStore API, providing actions of Get, Put, About, Remove. Higher level operations include Store, Register, Checksum, Retrieve, Cleanup, Clone, Migrate. Digital assets are stored on the bitstores by being transferred to the bitstore when it is uploaded or ingested. The exception to this is for "registered" objects, that the assets are put onto the filesystem ahead of time out-of-band, and during ingest, it just maps the database to know where the object already resides. The storage interface is such that additional storage implementations (i.e. other cloud storage providers) can be added with minimal difficulty.

DSBitStore stores content on a path on the filesystem. This could be locally attached normal filesystem, a mounted drive, or a mounted networked filesystem, it will all be treated as a local filesystem. All DSpace needs to be configured with for a filesystem, is the filesystem path, i.e. /dspace/assetstore, /opt/data/assetstore. The

DSBitStore uses a "Directory Scatter" method of storing an asset within 3 levels of subfolders, to minimize any single folder having too many objects for normal filesystem performance.

S3BitStore uses Amazon Web Services S3 (Simple Storage Service) to offer limitless cloud storage into a bucket, and each distinct asset will have a unique key. S3 is a commercial service (costs money), but is available at low price point, and is fully managed, content is automatically replicated, 99.999999999% object durability, integrity checked. Since S3 operates within the AWS network, using other AWS services, such virtual server on EC2 will provide lower network latency than local "on premises" servers. Additionally there could be some in-bound / out-bound bandwidth costs associated with DSpace application server outside of the AWS network communicating with S3, compared to AWS-internal EC2 servers. S3 has a checksum computing operation, in which the S3 service can return the checksum from the storage service, without having to shuttle the bits from S3, to your application server, and then computing the checksum. S3BitStore requires an S3 bucketName, accessKey, secretKey, and optionally specifying the AWS region, or a subfolder within the bucket.

There can be multiple bitstream stores. Each of these bitstream stores can be traditional storage or S3 storage. This means that the potential storage of a DSpace system is not bound by the maximum size of a single disk or file system and also that filesystem and S3storage can be combined in one DSpace installation. Both filesystem and S3 storage are specified by configuration. Also see Configuring the Bitstream Store below.

Stores are numbered, starting with zero, then counting upwards. Each bitstream entry in the database has a store number, used to retrieve the bitstream when required. An example of having multiple asset stores configured is that assetstore0 is /dspace/assetstore, when the filesystem gets nearly full, you could then configure a second filesystem path assetstore1 at /data/assetstore1, later, if you wanted to use S3 for storage, assetstore2 could be s3://dspace-assetstore-xyz(see page 686). In this example various bitstreams (database objects) refer to different assetstore for where the files reside. It is typically simplest to just have a single assetstore configured, and all assets reside in that one. If policy dictated, infrequently used masters could be moved to slower/cheaper disk, where as access copies are on the fastest storage. This could be accomplished through migrating assets to different stores.

Bitstreams also have an 38-digit internal ID, different from the primary key ID of the bitstream table row. This is not visible or used outside of the bitstream storage manager. It is used to determine the exact location (relative to the relevant store directory) that the bitstream is stored in traditional storage. The first three pairs of digits are the directory path that the bitstream is stored under. The bitstream is stored in a file with the internal ID as the filename.

For example, a bitstream with the internal ID *12345678901234567890123456789012345678* is stored in the directory:

```
[dspace]/assetstore/12/34/56/12345678901234567890123456789012345678
```

The reasons for storing files this way are:

- Using a randomly-generated 38-digit number means that the 'number space' is less cluttered than simply using the primary keys, which are allocated sequentially and are thus close together. This means that the bitstreams in the store are distributed around the directory structure, improving access efficiency.
- The internal ID is used as the filename partly to avoid requiring an extra lookup of the filename of the bitstream, and partly because bitstreams may be received from a variety of operating systems. The original name of a bitstream may be an illegal UNIX filename.
- When storing a bitstream, the *BitstreamStorageService* DOES set the following fields in the corresponding database table row:
    - *bitstream_id*
    - *size*
    - *checksum*
    - *checksum_algorithm*
    - *internal_id*
    - *deleted*

    - *store_number*
- The remaining fields are the responsibility of the *Bitstream* content management API class.

The bitstream storage manager is fully transaction-safe. In order to implement transaction-safety, the following algorithm is used to store bitstreams:

1. A database connection is created, separately from the currently active connection in the current DSpace context.
2. An unique internal identifier (separate from the database primary key) is generated.
3. The bitstream DB table row is created using this new connection, with the *deleted* column set to *true*.
4. The new connection is _commit_ted, so the 'deleted' bitstream row is written to the database
5. The bitstream itself is stored in a file in the configured 'asset store directory', with a directory path and filename derived from the internal ID
6. The *deleted* flag in the bitstream row is set to *false*. This will occur (or not) as part of the current DSpace *Context*.

This means that should anything go wrong before, during or after the bitstream storage, only one of the following can be true:

- No bitstream table row was created, and no file was stored
- A bitstream table row with *deleted=true* was created, no file was stored
- A bitstream table row with *deleted=true* was created, and a file was stored
  None of these affect the integrity of the data in the database or bitstream store.

Similarly, when a bitstream is deleted for some reason, its *deleted* flag is set to true as part of the overall transaction, and the corresponding file in storage is *not* deleted.

Cleanup

The above techniques mean that the bitstream storage manager is transaction-safe. Over time, the bitstream database table and file store may contain a number of 'deleted' bitstreams. The *cleanup* method of *BitstreamStorageService* goes through these deleted rows, and actually deletes them along with any corresponding files left in the storage. It only removes 'deleted' bitstreams that are more than one hour old, just in case cleanup is happening in the middle of a storage operation.

This cleanup can be invoked from the command line via the *cleanup* command, which can in turn be easily executed from a shell on the server machine using `[dspace]/bin/dspace cleanup`. You might like to have this run regularly by *cron*, though since DSpace is read-lots, write-not-so-much it doesn't need to be run very often.

```
# Clean up any deleted files from local storage on first of the month at 2:40am
40 2 1 * * [dspace]/bin/dspace cleanup > /dev/null 2>&1
```

Backup

The bitstreams (files) in traditional storage may be backed up very easily by simply 'tarring' or 'zipping' the `[dspace]/assetstore/` directory (or whichever directory is configured in *dspace.cfg*). Restoring is as simple as extracting the backed-up compressed file in the appropriate location.

It is important to note that since the bitstream storage manager holds the bitstreams in storage, and information about them in the database, that a database backup and a backup of the files in the bitstream store must be made at the same time; the bitstream data in the database must correspond to the stored files.

Of course, it isn't really ideal to 'freeze' the system while backing up to ensure that the database and files match up. Since DSpace uses the bitstream data in the database as the authoritative record, it's best to back up the database before the files. This is because it's better to have a bitstream in storage but not the database (effectively non-

existent to DSpace) than a bitstream record in the database but not storage, since people would be able to find the bitstream but not actually get the contents.

With DSpace 1.7 and above, there is also the option to backup both files and metadata via the AIP Backup and Restore(see page 411) feature.

Configuring the Bitstream Store

BitStores (aka assetstores) are configured with `[dspace]/config/spring/api/bitstore.xml`

Configuring Traditional Storage

By default, DSpace uses a traditional filesystem bitstore of `[dspace]/assetstore/`

To configure traditional filesystem bitstore, as a specific directory, configure the bitstore like this:

```
<bean name="org.dspace.storage.bitstore.BitstreamStorageService" class="org.dspace.storage.bitstore.BitstreamStorageServiceImpl">
    <property name="incoming" value="0"/>
    <property name="stores">
        <map>
            <entry key="0" value-ref="localStore"/>
        </map>
    </property>
</bean>

<bean name="localStore" class="org.dspace.storage.bitstore.DSBitStoreService" scope="singleton">
        <property name="baseDir" value="${dspace.dir}/assetstore"/>
</bean>
```

This would configure store number 0 named localStore, which is a `DSBitStore` (filesystem), at the filesystem path of `${dspace.dir}/assetstore` (i.e. `[dspace]/assetstore/`)

It is also possible to use multiple local filesystems. In the below example, key #0 is localStore at `${dspace.dir}/assetstore`, and key #1 is localStore2 at `/data/assetstore2`. Note that incoming is set to store "1", which in this case refers to localStore2. That means that any new files (bitstreams) uploaded to DSpace will be stored in localStore2, but some existing bitstreams may still exist in localStore.

```
<bean name="org.dspace.storage.bitstore.BitstreamStorageService" class="org.dspace.storage.bitstore.Bitstre
amStorageServiceImpl">
    <property name="incoming" value="1"/>
    <property name="stores">
        <map>
            <entry key="0" value-ref="localStore"/>
            <entry key="1" value-ref="localStore2"/>
        </map>
    </property>
</bean>
<bean name="localStore" class="org.dspace.storage.bitstore.DSBitStoreService" scope="singleton">
    <property name="baseDir" value="${dspace.dir}/assetstore"/>
</bean>
<bean name="localStore2" class="org.dspace.storage.bitstore.DSBitStoreService" scope="singleton">
    <property name="baseDir" value="/data/assetstore2"/>
</bean>
```

Configuring Amazon S3 Storage

To use Amazon S3[514] as a bitstore, add a bitstore entry `s3Store`, using `S3BitStoreService`, and configure it with `awsAccessKey`, `awsSecretKey`, and `bucketName`. NOTE: Before you can specify these settings, you obviously will have to create an account in the Amazon AWS[515] console, and create an IAM[516] user with credentials and privileges to an existing S3 bucket[517].

514 https://aws.amazon.com/s3/
515 http://aws.amazon.com/
516 https://aws.amazon.com/iam/
517 http://docs.aws.amazon.com/AmazonS3/latest/dev/UsingBucket.html

```
<bean name="org.dspace.storage.bitstore.BitstreamStorageService" class="org.dspace.storage.bitstore.Bitstre
amStorageServiceImpl">
    <property name="incoming" value="1"/>
    <property name="stores">
        <map>
            <entry key="0" value-ref="localStore"/>
            <entry key="1" value-ref="s3Store"/>
        </map>
    </property>
</bean>
<bean name="localStore" class="org.dspace.storage.bitstore.DSBitStoreService" scope="singleton">
    <property name="baseDir" value="${dspace.dir}/assetstore"/>
</bean>
<bean name="s3Store" class="org.dspace.storage.bitstore.S3BitStoreService" scope="singleton">
    <!-- AWS Security credentials, with policies for specified bucket -->
    <property name="awsAccessKey" value=""/>
    <property name="awsSecretKey" value=""/>
    <!-- S3 bucket name to store assets in. example: longsight-dspace-auk -->
    <property name="bucketName" value=""/>
    <!-- AWS S3 Region to use: {us-east-1, us-west-1, eu-west-1, eu-central-1, ap-southeast-1, ... } -->
    <!-- Optional, sdk default is us-east-1 -->
    <property name="awsRegionName" value=""/>
    <!-- Subfolder to organize assets within the bucket, in case this bucket is shared  -->
    <!-- Optional, default is root level of bucket -->
    <property name="subfolder" value=""/>
</bean>
```

The incoming property specifies which assetstore receives incoming assets (i.e. when new files are uploaded, they will be stored in the "incoming" assetstore). This defaults to store 0.

S3BitStore has parameters for awsAccessKey, awsSecretKey, bucketName, awsRegionName (optional), and subfolder (optional).

- `awsAccessKey` and `awsSecretKey` are created from the Amazon AWS[518] console. You'll want to create an IAM[519] user, and generate a Security Credential, which provides you the accessKey and secret. Since you need permission to use S3, you could give this IAM user a quick & dirty policy of AmazonS3FullAccess (for all S3 buckets that you own), or for finer grain controls, you can assign an IAM user to have certain permissions to certain resources, such as read/write to a specific subfolder within a specific s3 bucket.

518 http://aws.amazon.com/
519 https://aws.amazon.com/iam/

- `bucketName` is a globally unique name that distinguishes your S3 bucket. It has to be unique among all other S3 users in the world.
- `awsRegionName` is a region in AWS where S3 will be stored. Default is US Eastern. Consider distance to primary users, and pricing when choosing the region.
- `subfolder` is a folder within the S3 bucket, where you could organize the assets to be in. If you wanted to re-use a bucket for multiple purposes (bucketname/assets vs bucketname/backups) or DSpace instances (bucketname/XYZDSpace or bucketname/ABCDSpace or bucketname/ABCDSpaceProduction).

Migrate BitStores

There is a command line migration tool to move all the assets within a bitstore, to another bitstore. `bin/dspace bitstore-migrate`

```
[dspace]/bin/dspace bitstore-migrate
usage: BitstoreMigrate
 -a,--source <arg>        Source assetstore store_number (to lose content). This is a number such as 0 or 1
 -b,--destination <arg>   Destination assetstore store_number (to gain content). This is a number such as 0
 or 1.
 -d,--delete              Delete file from losing assetstore. (Default: Keep bitstream in old assetstore)
 -h,--help                Help
 -p,--print               Print out current assetstore information
 -s,--size <arg>          Batch commit size. (Default: 1, commit after each file transfer)

[dspace]/bin/dspace bitstore-migrate -p
store[0] == DSBitStore, which has 2 bitstreams.
store[1] == S3BitStore, which has 2 bitstreams.
Incoming assetstore is store[1]

[dspace]/bin/dspace bitstore-migrate -a 0 -b 1


[dspace]/bin/dspace bitstore-migrate -p
store[0] == DSBitStore, which has 0 bitstreams.
store[1] == S3BitStore, which has 4 bitstreams.
Incoming assetstore is store[1]
```

7.6 History

7.6.1 Changes in 7.x

7.6.1.1 Changes in DSpace 7.0

Changes in 7.0

- UI Changes: https://github.com/DSpace/dspace-angular/milestone/15?closed=1
- REST API Changes: https://github.com/DSpace/DSpace/milestone/21?closed=1

Changes in 7.0 Beta 5

- UI Changes: https://github.com/DSpace/dspace-angular/milestone/11?closed=1
- REST API Changes: https://github.com/DSpace/DSpace/milestone/35?closed=1

Changes in 7.0 Beta 4

- UI Changes: https://github.com/DSpace/dspace-angular/milestone/10?closed=1
- REST API Changes: https://github.com/DSpace/DSpace/milestone/33?closed=1

Changes in 7.0 Beta 3

- UI Changes: https://github.com/DSpace/dspace-angular/milestone/9?closed=1
- REST API Changes: https://github.com/DSpace/DSpace/milestone/34?closed=1

Changes in 7.0 Beta 2

- UI Changes: https://github.com/DSpace/dspace-angular/milestone/8?closed=1
- REST API Changes: https://github.com/DSpace/DSpace/milestone/38?closed=1

Changes in 7.0 Beta 1

- UI Changes: https://github.com/DSpace/dspace-angular/milestone/7?closed=1
- REST API Changes: https://github.com/DSpace/DSpace/milestone/32?closed=1

Changes in 7.0 Preview Release (Alpha)

- UI Changes: https://github.com/DSpace/dspace-angular/milestone/16?closed=1
- REST API Changes: https://github.com/DSpace/DSpace/milestone/39?closed=1

7.6.2 Changes in 6.x

7.6.2.1 Changes in DSpace 6.3

Improvements in 6.3

Key	Summary	T	Created	Updated	Due	Assignee	Reporter	P	Status	Resolution
DS-3447[520]	Transition ORCID integration to ORCID API 2.0[521]		Jan 11, 2017	Aug 29, 2018		Kevin Van de Velde (Atmire)	Bram Luyten (Atmire)		CLOSED	Fixed
DS-3560[522]	MathJax - Mirage MathJax depends on CDN that is being shut down[523]		Apr 10, 2017	Jun 22, 2018		Kim Shepherd	Tom Shorock		CLOSED	Fixed
DS-3667[524]	Document the persistence support classes[525]		Aug 08, 2017	Jun 22, 2018		Mark H. Wood	Mark H. Wood		CLOSED	Fixed
DS-3704[526]	Expand DSpace REST Reports to include bitstream fields in item listing[527]		Sep 28, 2017	Jun 22, 2018		Terrence W Brady	Terrence W Brady		CLOSED	Fixed
DS-3713[528]	REST Query/Collection Report - Bug Filtering for Bitstream Permissions[529]		Oct 05, 2017	Jun 22, 2018		Terrence W Brady	Terrence W Brady		CLOSED	Fixed
DS-3714[530]	REST Collection Report - Need a paginated findByCollection call that can return withdrawn items[531]		Oct 05, 2017	Jun 22, 2018		Terrence W Brady	Terrence W Brady		CLOSED	Fixed

520 https://jira.lyrasis.org/browse/DS-3447?src=confmacro
521 https://jira.lyrasis.org/browse/DS-3447?src=confmacro
522 https://jira.lyrasis.org/browse/DS-3560?src=confmacro
523 https://jira.lyrasis.org/browse/DS-3560?src=confmacro
524 https://jira.lyrasis.org/browse/DS-3667?src=confmacro
525 https://jira.lyrasis.org/browse/DS-3667?src=confmacro
526 https://jira.lyrasis.org/browse/DS-3704?src=confmacro
527 https://jira.lyrasis.org/browse/DS-3704?src=confmacro
528 https://jira.lyrasis.org/browse/DS-3713?src=confmacro
529 https://jira.lyrasis.org/browse/DS-3713?src=confmacro
530 https://jira.lyrasis.org/browse/DS-3714?src=confmacro
531 https://jira.lyrasis.org/browse/DS-3714?src=confmacro

DS-3792[532]	A couple of tags missing/changed in translation to German from 6.x[533]		Dec 20, 2017	Dec 20, 2017	Claudia Jürgen	Claudia Jürgen		CLOSED	Fixed
DS-3795[534]	bump dependency versions[535]		Jan 02, 2018	Aug 07, 2018	Mark H. Wood	Hardy Pottinger		CLOSED	Fixed
DS-3803[536]	Remove db.jndi setting from dspace.cfg[537]		Jan 16, 2018	Feb 05, 2018	Unassigned	Alan Orth		CLOSED	Fixed
DS-3811[538]	DSpace 6x REST Report Tools - Enable SSO Login using shibb auth[539]		Jan 22, 2018	Jun 22, 2018	Terrence W Brady	Terrence W Brady		CLOSED	Fixed
DS-3832[540]	GeoIP-API(com.maxmind.geoip:geoip-api) needs to be replaced by GeoIP2 (com.maxmind.geoip2:geoip2)[541]		Feb 07, 2018	Jun 22, 2018	Mark H. Wood	Sven Soliman		CLOSED	Fixed
DS-3835[542]	XMLUI Search Results Lose Collection/Community Context when Sorting with the Gear Icon[543]		Feb 08, 2018	Jun 22, 2018	Terrence W Brady	Terrence W Brady		CLOSED	Fixed
DS-3883[544]	DSpace 6x Performance Issues for Items with 100+ bitstreams[545]		Apr 04, 2018	Jun 24, 2019	Tim Donohue	Ying Jin		CLOSED	Fixed

13 issues[546]

Bug Fixes in 6.3

532 https://jira.lyrasis.org/browse/DS-3792?src=confmacro
533 https://jira.lyrasis.org/browse/DS-3792?src=confmacro
534 https://jira.lyrasis.org/browse/DS-3795?src=confmacro
535 https://jira.lyrasis.org/browse/DS-3795?src=confmacro
536 https://jira.lyrasis.org/browse/DS-3803?src=confmacro
537 https://jira.lyrasis.org/browse/DS-3803?src=confmacro
538 https://jira.lyrasis.org/browse/DS-3811?src=confmacro
539 https://jira.lyrasis.org/browse/DS-3811?src=confmacro
540 https://jira.lyrasis.org/browse/DS-3832?src=confmacro
541 https://jira.lyrasis.org/browse/DS-3832?src=confmacro
542 https://jira.lyrasis.org/browse/DS-3835?src=confmacro
543 https://jira.lyrasis.org/browse/DS-3835?src=confmacro
544 https://jira.lyrasis.org/browse/DS-3883?src=confmacro
545 https://jira.lyrasis.org/browse/DS-3883?src=confmacro
546 https://jira.lyrasis.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project+%3D+DS+AND+issuetype+in+%28Task%2C+Improvement%2C+%22Code+Task%22%2C+Documentation%2C+Sub-task%29+AND+resolution+%3D+Fixed+AND+fixVersion+%3D+%226.3%22+ORDER+BY+key+ASC+++&src=confmacro

Key	Summary	T	Created	Updated	Due	Assignee	Reporter	P	Status	Resolution
DS-1614[547]	MetadataExport.export() hides all exceptions from the user[548]		Aug 06, 2013	May 10, 2019		Unassigned	Ivan Masár		CLOSED	Fixed
DS-1818[549]	Ensure DSpace works with Creative Commons 4.0 licenses[550]		Dec 02, 2013	Aug 24, 2018		Unassigned	Tim Donohue		CLOSED	Fixed
DS-2675[551]	Jump to value in descending browse jumps to incorrect location, for author and subject browse[552]		Jul 23, 2015	Oct 03, 2018		Kim Shepherd	Hardy Pottinger		CLOSED	Fixed
DS-2862[553]	embargo lifer not setting bundle/bitstream policies[554]		Oct 30, 2015	May 03, 2018		Kim Shepherd	Monika Mevenkamp		CLOSED	Fixed
DS-3087[555]	Mirage2 uses single dolar sign delimiters for MathJax[556]		Feb 29, 2016	Dec 14, 2017		Unassigned	Hardy Pottinger		CLOSED	Fixed
DS-3310[557]	HTTP 500 error in the SWORD v2 interface[558]		Sep 08, 2016	Aug 06, 2018		Kim Shepherd	Anonymous (No Reply)		CLOSED	Fixed
DS-3332[559]	Handle resolver is hardcoded in org.dspace.handle.UpdateHandlePrefix[560]		Sep 21, 2016	Aug 06, 2018		Kim Shepherd	Pascal-Nicolas Becker		CLOSED	Fixed
DS-3377[561]	Solr query may be longer than allowed by the Web Container[562]		Nov 07, 2016	Jun 26, 2019		Kim Shepherd	Nicolas Schwab (SEDICI)		CLOSED	Fixed

547 https://jira.lyrasis.org/browse/DS-1614?src=confmacro
548 https://jira.lyrasis.org/browse/DS-1614?src=confmacro
549 https://jira.lyrasis.org/browse/DS-1818?src=confmacro
550 https://jira.lyrasis.org/browse/DS-1818?src=confmacro
551 https://jira.lyrasis.org/browse/DS-2675?src=confmacro
552 https://jira.lyrasis.org/browse/DS-2675?src=confmacro
553 https://jira.lyrasis.org/browse/DS-2862?src=confmacro
554 https://jira.lyrasis.org/browse/DS-2862?src=confmacro
555 https://jira.lyrasis.org/browse/DS-3087?src=confmacro
556 https://jira.lyrasis.org/browse/DS-3087?src=confmacro
557 https://jira.lyrasis.org/browse/DS-3310?src=confmacro
558 https://jira.lyrasis.org/browse/DS-3310?src=confmacro
559 https://jira.lyrasis.org/browse/DS-3332?src=confmacro
560 https://jira.lyrasis.org/browse/DS-3332?src=confmacro
561 https://jira.lyrasis.org/browse/DS-3377?src=confmacro
562 https://jira.lyrasis.org/browse/DS-3377?src=confmacro

DS-3404[563]	Authority Control: Wrong value for lookup popup (JavaScript) and missing index of repeatable fields in JSPUI[564]		Nov 28, 2016	May 16, 2018	Unassigned	Eike Kleiner		CLOSED	Fixed
DS-3434[565]	DSpace fails to start when a database connection pool is supplied through JNDI[566]		Dec 30, 2016	Feb 07, 2018	Mark H. Wood	Mark H. Wood		CLOSED	Fixed
DS-3461[567]	cleanup command fails to properly delete object files from storage[568]		Jan 19, 2017	Jun 22, 2018	Unassigned	Pedro Amorim		CLOSED	Fixed
DS-3498[569]	Full Text Index Behavior on Public Items with Embargoed Bitstreams[570]		Feb 16, 2017	Aug 06, 2018	Tim Donohue	Terrence W Brady		CLOSED	Fixed
DS-3507[571]	Search query escaping is not accurate[572]		Feb 23, 2017	May 09, 2018	Unassigned	Marsa Haoua		CLOSED	Fixed
DS-3511[573]	REST API, DSpace6.0 update bitstream data returns http 500[574]		Feb 27, 2017	Jun 22, 2018	Kim Shepherd	Elie Abu Haydar		CLOSED	Fixed
DS-3522[575]	Submission Resource Policy not correctly removed during XMLWorkflow[576]		Mar 08, 2017	Apr 02, 2020	Kim Shepherd	April Herron (Atmire)		CLOSED	Fixed
DS-3556[577]	OAI interface not working with a document using special charaters[578]		Apr 05, 2017	Apr 26, 2018	Kim Shepherd	Nelson Torres		CLOSED	Fixed

563 https://jira.lyrasis.org/browse/DS-3404?src=confmacro
564 https://jira.lyrasis.org/browse/DS-3404?src=confmacro
565 https://jira.lyrasis.org/browse/DS-3434?src=confmacro
566 https://jira.lyrasis.org/browse/DS-3434?src=confmacro
567 https://jira.lyrasis.org/browse/DS-3461?src=confmacro
568 https://jira.lyrasis.org/browse/DS-3461?src=confmacro
569 https://jira.lyrasis.org/browse/DS-3498?src=confmacro
570 https://jira.lyrasis.org/browse/DS-3498?src=confmacro
571 https://jira.lyrasis.org/browse/DS-3507?src=confmacro
572 https://jira.lyrasis.org/browse/DS-3507?src=confmacro
573 https://jira.lyrasis.org/browse/DS-3511?src=confmacro
574 https://jira.lyrasis.org/browse/DS-3511?src=confmacro
575 https://jira.lyrasis.org/browse/DS-3522?src=confmacro
576 https://jira.lyrasis.org/browse/DS-3522?src=confmacro
577 https://jira.lyrasis.org/browse/DS-3556?src=confmacro
578 https://jira.lyrasis.org/browse/DS-3556?src=confmacro

DS-3616[579]	Controlled vocabulary search doesn't work when delimiter is used[580]		Jun 09, 2017	Aug 06, 2018	Kim Shepherd	Eduardo Speroni		CLOSED	Fixed
DS-3627[581]	Cleanup utility leaves files in assetstore[582]		Jun 15, 2017	Aug 06, 2018	Pascal-Nicolas Becker	Chris Wilper		CLOSED	Fixed
DS-3629[583]	Listing of all Groups misses pagination[584]		Jun 19, 2017	Aug 06, 2018	Kim Shepherd	Martin Walk		CLOSED	Fixed
DS-3662[585]	DSpace 'logging in' without password or with non-existent e-mail using Shib and Password authentication[586]		Aug 03, 2017	Jun 27, 2018	Kim Shepherd	Jakub Řihák		CLOSED	Fixed

Showing 20 out of 58 issues[587]

7.6.2.2 Changes in DSpace 6.2

Bug Fixes in 6.2

Key	Summary	T	Created	Updated	Due	Assignee	Reporter	P	Status	Resolution
DS-1818[588]	Ensure DSpace works with Creative Commons 4.0 licenses[589]		Dec 02, 2013	Aug 24, 2018		Unassigned	Tim Donohue		CLOSED	Fixed
DS-3286[590]	Slow batch operations due to Hibernate caching[591]		Aug 11, 2016	Aug 17, 2017		Unassigned	Chris Wilper		CLOSED	Fixed

579 https://jira.lyrasis.org/browse/DS-3616?src=confmacro
580 https://jira.lyrasis.org/browse/DS-3616?src=confmacro
581 https://jira.lyrasis.org/browse/DS-3627?src=confmacro
582 https://jira.lyrasis.org/browse/DS-3627?src=confmacro
583 https://jira.lyrasis.org/browse/DS-3629?src=confmacro
584 https://jira.lyrasis.org/browse/DS-3629?src=confmacro
585 https://jira.lyrasis.org/browse/DS-3662?src=confmacro
586 https://jira.lyrasis.org/browse/DS-3662?src=confmacro
587 https://jira.lyrasis.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project+%3D+DS+AND+issuetype+%3D+Bug+AND+resolution+%3D+Fixed+AND+fixVersion+%3D+%226.3%22+ORDER+BY+key+ASC+++++&src=confmacro
588 https://jira.lyrasis.org/browse/DS-1818?src=confmacro
589 https://jira.lyrasis.org/browse/DS-1818?src=confmacro
590 https://jira.lyrasis.org/browse/DS-3286?src=confmacro
591 https://jira.lyrasis.org/browse/DS-3286?src=confmacro

DS-3517[592]	ImageMagick PDF thumbnail should always create sRGB JPEG files[593]	Mar 02, 2017	Sep 16, 2017	Unassigned	Alan Orth		CLOSED	Fixed
DS-3602[594]	Bitstream statistics are not shown after a migration to 6.x[595]	May 22, 2017	Jun 29, 2020	Terrence W Brady	Adan Roman		CLOSED	Fixed
DS-3648[596]	XMLUI Batch Import Failure[597]	Jul 14, 2017	Oct 01, 2018	Tom Desair	Steve Michaels		CLOSED	Fixed
DS-3656[598]	DOIOrganiser CLI does not change the database anymore[599]	Jul 20, 2017	Sep 06, 2017	Tom Desair	Sven Soliman		CLOSED	Fixed
DS-3659[600]	Database migrate fails to create the initial groups[601]	Aug 01, 2017	Sep 06, 2017	Tim Donohue	Alexander Sulfrian		CLOSED	Fixed
DS-3660[602]	Items fail to be reindexed on metadata change when 'authority' consumer is enabled[603]	Aug 02, 2017	Oct 01, 2018	Alexander Sulfrian	Sven Soliman		CLOSED	Fixed
DS-3661[604]	ImageMagick PDF Processing Degraded with Changes in 5.7 release[605]	Aug 02, 2017	Sep 06, 2017	Unassigned	Terrence W Brady		CLOSED	Fixed
DS-3674[606]	PluginServiceTest.java does not have a provided testing config and thus tests a live config[607]	Aug 15, 2017	Aug 15, 2017	Hardy Pottinger	Hardy Pottinger		CLOSED	Fixed
DS-3680[608]	Database changes of consumers aren't persisted anymore[609]	Aug 23, 2017	Sep 06, 2017	Pascal-Nicolas Becker	Pascal-Nicolas Becker		CLOSED	Fixed

592 https://jira.lyrasis.org/browse/DS-3517?src=confmacro
593 https://jira.lyrasis.org/browse/DS-3517?src=confmacro
594 https://jira.lyrasis.org/browse/DS-3602?src=confmacro
595 https://jira.lyrasis.org/browse/DS-3602?src=confmacro
596 https://jira.lyrasis.org/browse/DS-3648?src=confmacro
597 https://jira.lyrasis.org/browse/DS-3648?src=confmacro
598 https://jira.lyrasis.org/browse/DS-3656?src=confmacro
599 https://jira.lyrasis.org/browse/DS-3656?src=confmacro
600 https://jira.lyrasis.org/browse/DS-3659?src=confmacro
601 https://jira.lyrasis.org/browse/DS-3659?src=confmacro
602 https://jira.lyrasis.org/browse/DS-3660?src=confmacro
603 https://jira.lyrasis.org/browse/DS-3660?src=confmacro
604 https://jira.lyrasis.org/browse/DS-3661?src=confmacro
605 https://jira.lyrasis.org/browse/DS-3661?src=confmacro
606 https://jira.lyrasis.org/browse/DS-3674?src=confmacro
607 https://jira.lyrasis.org/browse/DS-3674?src=confmacro
608 https://jira.lyrasis.org/browse/DS-3680?src=confmacro
609 https://jira.lyrasis.org/browse/DS-3680?src=confmacro

DS-3687[610]	Hard coded note not compatible with multi-lingual sites for legacy stats[611]		Sep 01, 2017	Sep 06, 2017	Unassigned	Sébastien Nadeau	CLOSED	Fixed

12 issues[612]

7.6.2.3 Changes in DSpace 6.1

Improvements in 6.1

Key	Summary	T	Created	Updated	Due	Assignee	Reporter	P	Status	Resolution
DS-1140[613]	Update MSWord Media Filter to use Apache POI (like PPT Filter) and also support .docx[614]		Mar 12, 2012	Nov 09, 2018		Mark H. Wood	Tim Donohue		CLOSED	Fixed
DS-2646[615]	JSPUI should sort drop down list in collection selection form alphabetically[616]		Jul 08, 2015	Jul 13, 2017		Unassigned	Monika Mevenkamp		CLOSED	Fixed
DS-2696[617]	JSPUI sometimes fails to report error stack on exception[618]		Aug 07, 2015	Mar 29, 2017		Unassigned	Monika Mevenkamp		CLOSED	Fixed
DS-2840[619]	Changes INFO level sidebar facet transformer log entries to DEBUG[620]		Oct 25, 2015	Feb 18, 2017		Bram Luyten (Atmire)	Bram Luyten (Atmire)		CLOSED	Fixed
DS-3127[621]	Create a "whitelist" of formats allowable in citation_pdf_url for		Apr 13, 2016	Jul 11, 2017		Unassigned	Tim Donohue		CLOSED	Fixed

610 https://jira.lyrasis.org/browse/DS-3687?src=confmacro
611 https://jira.lyrasis.org/browse/DS-3687?src=confmacro
612 https://jira.lyrasis.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project+%3D+DS+AND+issuetype+%3D+Bug+AND+resolution+%3D+Fixed+AND+fixVersion+%3D+%226.2%22+ORDER+BY+key+ASC+++++&src=confmacro
613 https://jira.lyrasis.org/browse/DS-1140?src=confmacro
614 https://jira.lyrasis.org/browse/DS-1140?src=confmacro
615 https://jira.lyrasis.org/browse/DS-2646?src=confmacro
616 https://jira.lyrasis.org/browse/DS-2646?src=confmacro
617 https://jira.lyrasis.org/browse/DS-2696?src=confmacro
618 https://jira.lyrasis.org/browse/DS-2696?src=confmacro
619 https://jira.lyrasis.org/browse/DS-2840?src=confmacro
620 https://jira.lyrasis.org/browse/DS-2840?src=confmacro
621 https://jira.lyrasis.org/browse/DS-3127?src=confmacro

	Google Scholar (request from Google)[622]							
DS-3535[623]	Reduced error logging by interrupted download[624]	Mar 21, 2017	Apr 19, 2017	Unassigned	Per Broman		CLOSED	Fixed
DS-3564[625]	Change default value for db.maxidle[626]	Apr 11, 2017	Apr 20, 2017	Mark H. Wood	Pascal-Nicolas Becker		CLOSED	Fixed
DS-3573[627]	Filtername in XMLUI Discovery filter labels[628]	Apr 14, 2017	Apr 19, 2017	Unassigned	Yana De Pauw (Atmire)		CLOSED	Fixed
DS-3575[629]	Misguiding find method in ResourcePolicyService[630]	Apr 18, 2017	Apr 21, 2017	Unassigned	Pascal-Nicolas Becker		CLOSED	Fixed
DS-3601[631]	NullPointerException when accessing the feedback page without Referer header set[632]	May 22, 2017	Nov 02, 2018	Unassigned	Alexander Sulfrian		CLOSED	Fixed

10 issues[633]

Bug Fixes in 6.1

Key	Summary	T	Created	Updated	Due	Assignee	Reporter	P	Status	Resolution
DS-1818[634]	Ensure DSpace works with Creative Commons 4.0 licenses[635]		Dec 02, 2013	Aug 24, 2018		Unassigned	Tim Donohue		CLOSED	Fixed

622 https://jira.lyrasis.org/browse/DS-3127?src=confmacro
623 https://jira.lyrasis.org/browse/DS-3535?src=confmacro
624 https://jira.lyrasis.org/browse/DS-3535?src=confmacro
625 https://jira.lyrasis.org/browse/DS-3564?src=confmacro
626 https://jira.lyrasis.org/browse/DS-3564?src=confmacro
627 https://jira.lyrasis.org/browse/DS-3573?src=confmacro
628 https://jira.lyrasis.org/browse/DS-3573?src=confmacro
629 https://jira.lyrasis.org/browse/DS-3575?src=confmacro
630 https://jira.lyrasis.org/browse/DS-3575?src=confmacro
631 https://jira.lyrasis.org/browse/DS-3601?src=confmacro
632 https://jira.lyrasis.org/browse/DS-3601?src=confmacro
8 https://jira.lyrasis.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project+%3D+DS+AND+issuetype+in+%28Task%2C+Improvement%2C+%22Code+Task%22%2C+Documentation%2C+Sub-task%29+AND+resolution+%3D+Fixed+AND+fixVersion+%3D+%226.1%22+ORDER+BY+key+ASC++&src=confmacro
634 https://jira.lyrasis.org/browse/DS-1818?src=confmacro
635 https://jira.lyrasis.org/browse/DS-1818?src=confmacro

DS-2227[636]	failed AIP import leaves files in assetstore[637]	Oct 29, 2014	May 11, 2017	Hardy Pottinger	Ivan Masár		CLOSED	Fixed
DS-2291[638]	Autocomplete not working in submission in Mirage 2[639]	Nov 13, 2014	Jul 09, 2020	Unassigned	Elvi S. Nemiz		CLOSED	Fixed
DS-2299[640]	Warn repo admins if they mistakenly leave useProxies set to false[641]	Nov 16, 2014	Mar 15, 2017	Bram Luyten (Atmire)	Bram Luyten (Atmire)		CLOSED	Fixed
DS-2359[642]	Error when depositing large files via browser (over 2Gb)[643]	Dec 11, 2014	Jul 07, 2017	Unassigned	Pauline Ward		CLOSED	Fixed
DS-2748[644]	Cocoon logs overly verbose when 404s occur[645]	Sep 10, 2015	Jun 22, 2017	Unassigned	Chris Wilper		CLOSED	Fixed
DS-2789[646]	Display a "restricted image" for a thumbnail if the bitstream is restricted[647]	Oct 06, 2015	Mar 13, 2017	Terrence W Brady	George Kozak		CLOSED	Fixed
DS-2947[648]	DIM crosswalks repeats authority & confidence values in the metadata values[649]	Dec 11, 2015	Mar 24, 2017	Unassigned	Emilio Lorenzo (Arvo)		CLOSED	Fixed
DS-2952[650]	SOLR: Full text indexing only includes the text on the last bitstream[651]	Dec 14, 2015	Feb 22, 2017	Unassigned	vtown		CLOSED	Fixed
DS-3108[652]	Support Shibboleth Authentication in the REST API[653]	Mar 23, 2016	Mar 23, 2017	Unassigned	Terrence W Brady		CLOSED	Fixed

636 https://jira.lyrasis.org/browse/DS-2227?src=confmacro
637 https://jira.lyrasis.org/browse/DS-2227?src=confmacro
638 https://jira.lyrasis.org/browse/DS-2291?src=confmacro
639 https://jira.lyrasis.org/browse/DS-2291?src=confmacro
640 https://jira.lyrasis.org/browse/DS-2299?src=confmacro
641 https://jira.lyrasis.org/browse/DS-2299?src=confmacro
642 https://jira.lyrasis.org/browse/DS-2359?src=confmacro
643 https://jira.lyrasis.org/browse/DS-2359?src=confmacro
644 https://jira.lyrasis.org/browse/DS-2748?src=confmacro
645 https://jira.lyrasis.org/browse/DS-2748?src=confmacro
646 https://jira.lyrasis.org/browse/DS-2789?src=confmacro
647 https://jira.lyrasis.org/browse/DS-2789?src=confmacro
648 https://jira.lyrasis.org/browse/DS-2947?src=confmacro
649 https://jira.lyrasis.org/browse/DS-2947?src=confmacro
650 https://jira.lyrasis.org/browse/DS-2952?src=confmacro
651 https://jira.lyrasis.org/browse/DS-2952?src=confmacro
652 https://jira.lyrasis.org/browse/DS-3108?src=confmacro
653 https://jira.lyrasis.org/browse/DS-3108?src=confmacro

DS-3164[654]	Item statistic displays UUID of bitstreams instead of name[655]		Apr 26, 2016	May 10, 2017	Unassigned	Claudia Jürgen		CLOSED	Fixed
DS-3245[656]	CSV linebreaks not supported by Bulkedit[657]		Jun 15, 2016	Aug 17, 2018	Unassigned	None		CLOSED	Fixed
DS-3281[658]	Submission made via REST API does not trigger collection workflow approval process.[659]		Aug 03, 2016	May 31, 2017	Unassigned	Emilio Lorenzo (Arvo)		CLOSED	Fixed
DS-3282[660]	Mirage2: Advanced Search Filters Do Not Display if one contains a "_"[661]		Aug 08, 2016	Jul 13, 2017	Terrence W Brady	Terrence W Brady		CLOSED	Fixed
DS-3283[662]	Mirage2: Edit Collection Source - No Field Label for Set Id[663]		Aug 08, 2016	Jul 13, 2017	Terrence W Brady	Terrence W Brady		CLOSED	Fixed
DS-3286[664]	Slow batch operations due to Hibernate caching[665]		Aug 11, 2016	Aug 17, 2017	Unassigned	Chris Wilper		CLOSED	Fixed
DS-3289[666]	Mirage 2 with Jetty 9 has broken image links[667]		Aug 12, 2016	Jan 09, 2017	Bram Luyten (Atmire)	Ilja Sidoroff		CLOSED	Fixed
DS-3331[668]	Multi-shard solr statistics queries suppress key information[669]		Sep 19, 2016	Feb 22, 2017	Unassigned	Terrence W Brady		CLOSED	Fixed
DS-3334[670]	XMLUI Archived Submissions display in chronological order (in		Sep 22, 2016	Mar 08, 2017	Unassigned	Terrence W Brady		CLOSED	Fixed

654 https://jira.lyrasis.org/browse/DS-3164?src=confmacro
655 https://jira.lyrasis.org/browse/DS-3164?src=confmacro
656 https://jira.lyrasis.org/browse/DS-3245?src=confmacro
657 https://jira.lyrasis.org/browse/DS-3245?src=confmacro
658 https://jira.lyrasis.org/browse/DS-3281?src=confmacro
659 https://jira.lyrasis.org/browse/DS-3281?src=confmacro
660 https://jira.lyrasis.org/browse/DS-3282?src=confmacro
661 https://jira.lyrasis.org/browse/DS-3282?src=confmacro
662 https://jira.lyrasis.org/browse/DS-3283?src=confmacro
663 https://jira.lyrasis.org/browse/DS-3283?src=confmacro
664 https://jira.lyrasis.org/browse/DS-3286?src=confmacro
665 https://jira.lyrasis.org/browse/DS-3286?src=confmacro
666 https://jira.lyrasis.org/browse/DS-3289?src=confmacro
667 https://jira.lyrasis.org/browse/DS-3289?src=confmacro
668 https://jira.lyrasis.org/browse/DS-3331?src=confmacro
669 https://jira.lyrasis.org/browse/DS-3331?src=confmacro
670 https://jira.lyrasis.org/browse/DS-3334?src=confmacro

	DSpace 5 it was reverse chronological)[671]								
DS-3336[672]	When moving an item, list of collections sorts by collection instead of community[673]		Sep 25, 2016	Mar 08, 2017	Andrea Schweer	Deborah Fitchett		CLOSED	Fixed

Showing 20 out of 77 issues[674]

7.6.2.4 Changes in DSpace 6.0

New Features in 6.0

Key	Summary	T	Created	Updated	Due	Assignee	Reporter	P	Status	Resolution
DS-1262[675]	CSV export of search results in XMLUI[676]		Sep 12, 2012	Feb 03, 2016		Ivan Masár	William Welling		CLOSED	Fixed
DS-1782[677]	DSpace needs local object identifiers[678]		Nov 12, 2013	Feb 03, 2016		Mark H. Wood	Mark H. Wood		CLOSED	Fixed
DS-2539[679]	Provide REST API function to dump the metadata repository[680]		Apr 06, 2015	Apr 11, 2016		Terrence W Brady	Terrence W Brady		CLOSED	Fixed
DS-2583[681]	Extend DSpace REST API verbs to support enhanced reporting[682]		May 19, 2015	Apr 11, 2016		Terrence W Brady	Terrence W Brady		CLOSED	Fixed

671 https://jira.lyrasis.org/browse/DS-3334?src=confmacro
672 https://jira.lyrasis.org/browse/DS-3336?src=confmacro
673 https://jira.lyrasis.org/browse/DS-3336?src=confmacro
674 https://jira.lyrasis.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project+%3D+DS+AND+issuetype+%3D+Bug+AND+resolution+%3D+Fixed+AND+fixVersion+%3D+%226.1%22+ORDER+BY+key+ASC++++&src=confmacro
675 https://jira.lyrasis.org/browse/DS-1262?src=confmacro
676 https://jira.lyrasis.org/browse/DS-1262?src=confmacro
677 https://jira.lyrasis.org/browse/DS-1782?src=confmacro
678 https://jira.lyrasis.org/browse/DS-1782?src=confmacro
679 https://jira.lyrasis.org/browse/DS-2539?src=confmacro
680 https://jira.lyrasis.org/browse/DS-2539?src=confmacro
681 https://jira.lyrasis.org/browse/DS-2583?src=confmacro
682 https://jira.lyrasis.org/browse/DS-2583?src=confmacro

DS-2629[683]	Add ability to filter Excel (xls and xlsx) files for full text searching[684]		Jun 22, 2015	Apr 15, 2016	Kevin Van de Velde (Atmire)	Ed Goulet	CLOSED	Fixed
DS-2648[685]	Fulltext available sidebar facet for Discovery[686]		Jul 09, 2015	Dec 17, 2015	Ivan Masár	Christian Scheible	CLOSED	Fixed
DS-2653[687]	Java 8 Support for DSpace[688]		Jul 13, 2015	Aug 16, 2016	Tim Donohue	Tim Donohue	CLOSED	Fixed
DS-2654[689]	Reloadable Configurations via Apache Commons Configuration[690]		Jul 14, 2015	Jul 04, 2016	Tim Donohue	Tim Donohue	CLOSED	Fixed
DS-2659[691]	Add configurable "healthcheck" system emailing internals of the repository on regular basis[692]		Jul 17, 2015	Jan 28, 2016	Ivan Masár	Jozef (@lindat)	CLOSED	Fixed
DS-2693[693]	Spanish translation update[694]		Aug 04, 2015	Nov 17, 2015	Ivan Masár	Miguel Carro Pellicer	CLOSED	Fixed
DS-2701[695]	Adopt Service-based API refactor of existing Java API[696]		Aug 12, 2015	Jul 27, 2016	Kevin Van de Velde (Atmire)	Tim Donohue	CLOSED	Fixed
DS-2876[697]	Framework to better support metadata import from external sources[698]		Nov 12, 2015	Jul 30, 2021	Unassigned	Roeland Dillen	CLOSED	Fixed
DS-2880[699]	Pubmed integration into XMLUI submission[700]		Nov 12, 2015	Aug 03, 2016	Unassigned	Roeland Dillen	CLOSED	Fixed

683 https://jira.lyrasis.org/browse/DS-2629?src=confmacro
684 https://jira.lyrasis.org/browse/DS-2629?src=confmacro
685 https://jira.lyrasis.org/browse/DS-2648?src=confmacro
686 https://jira.lyrasis.org/browse/DS-2648?src=confmacro
687 https://jira.lyrasis.org/browse/DS-2653?src=confmacro
688 https://jira.lyrasis.org/browse/DS-2653?src=confmacro
689 https://jira.lyrasis.org/browse/DS-2654?src=confmacro
690 https://jira.lyrasis.org/browse/DS-2654?src=confmacro
691 https://jira.lyrasis.org/browse/DS-2659?src=confmacro
692 https://jira.lyrasis.org/browse/DS-2659?src=confmacro
693 https://jira.lyrasis.org/browse/DS-2693?src=confmacro
694 https://jira.lyrasis.org/browse/DS-2693?src=confmacro
695 https://jira.lyrasis.org/browse/DS-2701?src=confmacro
696 https://jira.lyrasis.org/browse/DS-2701?src=confmacro
697 https://jira.lyrasis.org/browse/DS-2876?src=confmacro
698 https://jira.lyrasis.org/browse/DS-2876?src=confmacro
699 https://jira.lyrasis.org/browse/DS-2880?src=confmacro
700 https://jira.lyrasis.org/browse/DS-2880?src=confmacro

DS-2888[701]	JSPUI: Let users add language tags in submission's edit metadata step[702]		Nov 13, 2015	Sep 29, 2017	Pascal-Nicolas Becker	Pascal-Nicolas Becker		CLOSED	Fixed
DS-2890[703]	org.dspace.app.util.Util.getUUIDParameter(...) is quite noisy and produces a lot of NPEs[704]		Nov 13, 2015	Nov 13, 2015	Pascal-Nicolas Becker	Pascal-Nicolas Becker		CLOSED	Fixed
DS-3055[705]	Remove support for SRB (Storage Resource Broker) as it's unmaintained and outdated[706]		Feb 11, 2016	Mar 23, 2016	Peter Dietz	Tim Donohue		CLOSED	Fixed

16 issues[707]

Improvements in 6.0

Key	Summary	T	Created	Updated	Due	Assignee	Reporter	P	Status	Resolution
DS-79[708]	Pluggable storage / S3 - ID: 2561561[709]		Mar 06, 2009	Apr 02, 2016		Peter Dietz	Charles Kiplagat		CLOSED	Fixed
DS-950[710]	Update OAI-PMH to fully obey 'metadata.hide.SCHEMA.ELEMENT.QUALIFIER' configuration settings[711]		Jul 11, 2011	Jun 22, 2018		Pascal-Nicolas Becker	Tim Donohue		CLOSED	Fixed
DS-1187[712]	Full-text indexing of right-to-left PDF files[713]		Jun 07, 2012	Feb 03, 2016		Ivan Masár	Saiful Amin		CLOSED	Fixed

701 https://jira.lyrasis.org/browse/DS-2888?src=confmacro
702 https://jira.lyrasis.org/browse/DS-2888?src=confmacro
703 https://jira.lyrasis.org/browse/DS-2890?src=confmacro
704 https://jira.lyrasis.org/browse/DS-2890?src=confmacro
705 https://jira.lyrasis.org/browse/DS-3055?src=confmacro
706 https://jira.lyrasis.org/browse/DS-3055?src=confmacro
70 https://jira.lyrasis.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project+%3D+DS+AND+issuetype+%3D+%22New+Feature%22+AND+resolution+%3D+Fixed+AND+fixVersion+%3D+%226.0%22+ORDER+BY+key+ASC++&src=confmacro
708 https://jira.lyrasis.org/browse/DS-79?src=confmacro
709 https://jira.lyrasis.org/browse/DS-79?src=confmacro
710 https://jira.lyrasis.org/browse/DS-950?src=confmacro
711 https://jira.lyrasis.org/browse/DS-950?src=confmacro
712 https://jira.lyrasis.org/browse/DS-1187?src=confmacro
713 https://jira.lyrasis.org/browse/DS-1187?src=confmacro

DS-1390[714]	DSpace has too many configurations[715]		Nov 15, 2012	Apr 27, 2016	Mark H. Wood	Mark H. Wood		CLOSED	Fixed
DS-1518[716]	Support StartTLS in LDAPAuthentication[717]		Mar 13, 2013	Feb 17, 2016	Ivan Masár	Ivan Masár		CLOSED	Fixed
DS-1814[718]	Allow submitter to create a new version of an item[719]		Dec 01, 2013	Mar 07, 2017	Pascal-Nicolas Becker	Andrea Bollini (4Science)		CLOSED	Fixed
DS-1837[720]	Create a separate Maven artifact for the MultiRemoteDSpaceRepositoryHandlePlugin[721]		Dec 11, 2013	Feb 18, 2016	Mark H. Wood	Mark H. Wood		CLOSED	Fixed
DS-2022[722]	log4j-handle-plugin.properties still points to dspace/config/templates[723]		Jun 09, 2014	Mar 18, 2018	Mark H. Wood	Christian Völker		CLOSED	Fixed
DS-2115[724]	Rewrite ConfigurationManager methods to wrap ConfigurationService[725]		Aug 23, 2014	Feb 15, 2016	Tim Donohue	Mark H. Wood		CLOSED	Fixed
DS-2124[726]	Move LNI to a separate GitHub project[727]		Aug 28, 2014	Feb 03, 2016	Robin Taylor	Tim Donohue		CLOSED	Fixed
DS-2159[728]	Deprecate XPDF in DSpace 5, remove in DSpace 6[729]		Sep 24, 2014	May 04, 2016	Mark H. Wood	Peter Dietz		CLOSED	Fixed

714 https://jira.lyrasis.org/browse/DS-1390?src=confmacro
715 https://jira.lyrasis.org/browse/DS-1390?src=confmacro
716 https://jira.lyrasis.org/browse/DS-1518?src=confmacro
717 https://jira.lyrasis.org/browse/DS-1518?src=confmacro
718 https://jira.lyrasis.org/browse/DS-1814?src=confmacro
719 https://jira.lyrasis.org/browse/DS-1814?src=confmacro
720 https://jira.lyrasis.org/browse/DS-1837?src=confmacro
721 https://jira.lyrasis.org/browse/DS-1837?src=confmacro
722 https://jira.lyrasis.org/browse/DS-2022?src=confmacro
723 https://jira.lyrasis.org/browse/DS-2022?src=confmacro
724 https://jira.lyrasis.org/browse/DS-2115?src=confmacro
725 https://jira.lyrasis.org/browse/DS-2115?src=confmacro
726 https://jira.lyrasis.org/browse/DS-2124?src=confmacro
727 https://jira.lyrasis.org/browse/DS-2124?src=confmacro
728 https://jira.lyrasis.org/browse/DS-2159?src=confmacro
729 https://jira.lyrasis.org/browse/DS-2159?src=confmacro

DS-2160[730]	Remove Lucene search index support and DBMS browse support from DSpace[731]		Sep 24, 2014	Jan 12, 2017	Tim Donohue	Tim Donohue		CLOSED	Fixed
DS-2187[732]	Remove deprecated Lucene search index support[733]		Oct 10, 2014	Feb 22, 2016	Tim Donohue	Mark H. Wood		CLOSED	Fixed
DS-2188[734]	Remove deprecated DBMS browse support[735]		Oct 10, 2014	Apr 05, 2016	Tim Donohue	Mark H. Wood		CLOSED	Fixed
DS-2315[736]	export without bitstreams[737]		Nov 18, 2014	Dec 11, 2015	Ivan Masár	Petya Kohts		CLOSED	Fixed
DS-2318[738]	Add border on Thumbnail Item[739]		Nov 21, 2014	Mar 01, 2016	Unassigned	Wesley Bastos		CLOSED	Fixed
DS-2372[740]	Export for ORCID authority cache[741]		Dec 17, 2014	Jul 24, 2019	Unassigned	Bram Luyten (Atmire)		CLOSED	Fixed
DS-2397[742]	Sort out Unit vs. Integration tests, and run them separately[743]		Jan 09, 2015	Feb 03, 2016	Mark H. Wood	Mark H. Wood		CLOSED	Fixed
DS-2426[744]	OAI transformers (xsl) are not able to use relative paths for import[745]		Jan 26, 2015	Jan 31, 2016	Ivan Masár	Christian Scheible		CLOSED	Fixed
DS-2452[746]	Upgrade to Apache Commons DBCP v2.x, and Apache Commons Pool 2.x[747]		Feb 11, 2015	Jan 15, 2016	Tim Donohue	Tim Donohue		CLOSED	Fixed

730 https://jira.lyrasis.org/browse/DS-2160?src=confmacro
731 https://jira.lyrasis.org/browse/DS-2160?src=confmacro
732 https://jira.lyrasis.org/browse/DS-2187?src=confmacro
733 https://jira.lyrasis.org/browse/DS-2187?src=confmacro
734 https://jira.lyrasis.org/browse/DS-2188?src=confmacro
735 https://jira.lyrasis.org/browse/DS-2188?src=confmacro
736 https://jira.lyrasis.org/browse/DS-2315?src=confmacro
737 https://jira.lyrasis.org/browse/DS-2315?src=confmacro
738 https://jira.lyrasis.org/browse/DS-2318?src=confmacro
739 https://jira.lyrasis.org/browse/DS-2318?src=confmacro
740 https://jira.lyrasis.org/browse/DS-2372?src=confmacro
741 https://jira.lyrasis.org/browse/DS-2372?src=confmacro
742 https://jira.lyrasis.org/browse/DS-2397?src=confmacro
743 https://jira.lyrasis.org/browse/DS-2397?src=confmacro
744 https://jira.lyrasis.org/browse/DS-2426?src=confmacro
745 https://jira.lyrasis.org/browse/DS-2426?src=confmacro
746 https://jira.lyrasis.org/browse/DS-2452?src=confmacro
747 https://jira.lyrasis.org/browse/DS-2452?src=confmacro

Showing 20 out of 100 issues[748]

Bug Fixes in 6.0

Key	Summary	T	Created	Updated	Due	Assignee	Reporter	P	Status	Resolution
DS-444[749]	java.lang.ClassCastException: org.mozilla.javascript.NativeContinuation cannot be cast to org.mozilla.javascript.continuations.Continuation[750]		Dec 29, 2009	Aug 22, 2016		Tim Donohue	Lee Li		CLOSED	Fixed
DS-850[751]	Vocabulary with closed="true" not save value filled[752]		Mar 19, 2011	Apr 25, 2016		Kevin Van de Velde (Atmire)	Onivaldo Rosa Junior		CLOSED	Fixed
DS-1349[753]	Item level versioning exposes personal data (name and email of submitter, versioning creator)[754]		Oct 23, 2012	May 07, 2018		Pascal-Nicolas Becker	Claudia Jürgen		CLOSED	Fixed
DS-1698[755]	saving the Edit policy form without specifying action results in stacktrace (regression)[756]		Oct 14, 2013	Mar 23, 2016		Unassigned	Ivan Masár		CLOSED	Fixed

8 https://jira.lyrasis.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project+%3D+DS+AND+issuetype+in+%28Task%2C+Improvement%2C+%22Code+Task%22%2C+Documentation%2C+Sub-task%29+AND+resolution+%3D+Fixed+AND+fixVersion+%3D+%226.0%22+ORDER+BY+key+ASC+&src=confmacro

749 https://jira.lyrasis.org/browse/DS-444?src=confmacro
750 https://jira.lyrasis.org/browse/DS-444?src=confmacro
751 https://jira.lyrasis.org/browse/DS-850?src=confmacro
752 https://jira.lyrasis.org/browse/DS-850?src=confmacro
753 https://jira.lyrasis.org/browse/DS-1349?src=confmacro
754 https://jira.lyrasis.org/browse/DS-1349?src=confmacro
755 https://jira.lyrasis.org/browse/DS-1698?src=confmacro
756 https://jira.lyrasis.org/browse/DS-1698?src=confmacro

DS-1805[757]	Maven filtering broken for SOLR artifact[758]		Nov 26, 2013	Feb 15, 2016	Unassigned	Bram Luyten (Atmire)		CLOSED	Fixed
DS-1818[759]	Ensure DSpace works with Creative Commons 4.0 licenses[760]		Dec 02, 2013	Aug 24, 2018	Unassigned	Tim Donohue		CLOSED	Fixed
DS-1865[761]	XPDF requires manually installing a JAR which is NOT available in Maven Central[762]		Jan 15, 2014	Feb 17, 2016	Mark H. Wood	Tim Donohue		CLOSED	Fixed
DS-1924[763]	currentLocale in DRI document is not changed[764]		Feb 23, 2014	Oct 06, 2015	Andrea Schweer	Supasate Choochaisri		CLOSED	Fixed
DS-1929[765]	editing bitstream description changes bitstream resourcepolicies[766]		Feb 28, 2014	Nov 17, 2016	Unassigned	Jose Blanco		CLOSED	Fixed
DS-1955[767]	Embargo reason field max input length[768]		Apr 01, 2014	Apr 20, 2016	Kevin Van de Velde (Atmire)	Michael Hicke		CLOSED	Fixed

757 https://jira.lyrasis.org/browse/DS-1805?src=confmacro
758 https://jira.lyrasis.org/browse/DS-1805?src=confmacro
759 https://jira.lyrasis.org/browse/DS-1818?src=confmacro
760 https://jira.lyrasis.org/browse/DS-1818?src=confmacro
761 https://jira.lyrasis.org/browse/DS-1865?src=confmacro
762 https://jira.lyrasis.org/browse/DS-1865?src=confmacro
763 https://jira.lyrasis.org/browse/DS-1924?src=confmacro
764 https://jira.lyrasis.org/browse/DS-1924?src=confmacro
765 https://jira.lyrasis.org/browse/DS-1929?src=confmacro
766 https://jira.lyrasis.org/browse/DS-1929?src=confmacro
767 https://jira.lyrasis.org/browse/DS-1955?src=confmacro
768 https://jira.lyrasis.org/browse/DS-1955?src=confmacro

DS-2020[769]	NullPointerException in org.dspace.xoai.filter.DSpaceSetSpecFilter[770]		Jun 05, 2014	May 18, 2015	João Melo	Ondřej Košarko		CLOSED	Fixed
DS-2268[771]	OpenSearch is still dependent on Lucene indexing, which might not be enabled[772]		Nov 07, 2014	Feb 22, 2016	Tim Donohue	Mark H. Wood		CLOSED	Fixed
DS-2358[773]	Item-level versioning discards embargo[774]		Dec 10, 2014	Jul 28, 2015	Pascal-Nicolas Becker	Andrea Schweer		CLOSED	Fixed
DS-2379[775]	command usage output for dspace (from launcher.xml) is unsorted[776]		Dec 22, 2014	Aug 01, 2016	Mark H. Wood	Hardy Pottinger		CLOSED	Fixed
DS-2403[777]	RDFConsumer tries to index unpublished WorkspaceItems[778]		Jan 13, 2015	May 08, 2015	Pascal-Nicolas Becker	Pascal-Nicolas Becker		CLOSED	Fixed
DS-2412[779]	The oai "Show more" link in stylesheet has a fixed verb[780]		Jan 16, 2015	Jan 20, 2015	Ivan Masár	Ondřej Košarko		CLOSED	Fixed
DS-2423[781]	No longer possibile to create additional Filter for OAI-PMH interface[782]		Jan 26, 2015	May 12, 2015	João Melo	Christian Scheible		CLOSED	Fixed
DS-2437[783]	Stop relying on alphabetical loading of jars in WEB-INF/lib[784]		Feb 03, 2015	May 30, 2018	Tim Donohue	Bram Luyten (Atmire)		CLOSED	Fixed

769 https://jira.lyrasis.org/browse/DS-2020?src=confmacro
770 https://jira.lyrasis.org/browse/DS-2020?src=confmacro
771 https://jira.lyrasis.org/browse/DS-2268?src=confmacro
772 https://jira.lyrasis.org/browse/DS-2268?src=confmacro
773 https://jira.lyrasis.org/browse/DS-2358?src=confmacro
774 https://jira.lyrasis.org/browse/DS-2358?src=confmacro
775 https://jira.lyrasis.org/browse/DS-2379?src=confmacro
776 https://jira.lyrasis.org/browse/DS-2379?src=confmacro
777 https://jira.lyrasis.org/browse/DS-2403?src=confmacro
778 https://jira.lyrasis.org/browse/DS-2403?src=confmacro
779 https://jira.lyrasis.org/browse/DS-2412?src=confmacro
780 https://jira.lyrasis.org/browse/DS-2412?src=confmacro
781 https://jira.lyrasis.org/browse/DS-2423?src=confmacro
782 https://jira.lyrasis.org/browse/DS-2423?src=confmacro
783 https://jira.lyrasis.org/browse/DS-2437?src=confmacro
784 https://jira.lyrasis.org/browse/DS-2437?src=confmacro

DS-2446[785]	altmetrics.field property is not read[786]		Feb 06, 2015	Mar 30, 2016	Mark H. Wood	Àlex Magaz Graça		CLOSED	Fixed
DS-2463[787]	iplists.com-non_engines.txt is outdated and slow[788]		Feb 20, 2015	Apr 15, 2016	Mark H. Wood	Mark H. Wood		CLOSED	Fixed

Showing 20 out of 274 issues[789]

7.6.3 Changes in 5.x

- Changes in DSpace 5.9(see page 716)
- Changes in DSpace 5.8(see page 719)
- Changes in DSpace 5.7(see page 719)
- Changes in DSpace 5.6(see page 722)
- Changes in DSpace 5.5(see page 725)
- Changes in DSpace 5.4(see page 726)
- Changes in DSpace 5.3(see page 729)
- Changes in DSpace 5.2(see page 732)
- Changes in DSpace 5.1(see page 736)
- Changes in DSpace 5.0(see page 740)

7.6.3.1 Changes in DSpace 5.9

Key	Summary	T	Created	Updated	Due	Assignee	Reporter	P	Status	Resolution

785 https://jira.lyrasis.org/browse/DS-2446?src=confmacro
786 https://jira.lyrasis.org/browse/DS-2446?src=confmacro
787 https://jira.lyrasis.org/browse/DS-2463?src=confmacro
788 https://jira.lyrasis.org/browse/DS-2463?src=confmacro
789 https://jira.lyrasis.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project+%3D+DS+AND+issuetype+%3D+Bug+AND+resolution+%3D+Fixed+AND+fixVersion+%3D+%226.0%22+ORDER+BY+key+ASC+++&src=confmacro

DS-950[790]	Update OAI-PMH to fully obey 'metadata.hide.SCHEMA.ELEMENT.QUALIFIER' configuration settings[791]		Jul 11, 2011	Jun 22, 2018	Pascal-Nicolas Becker	Tim Donohue		CLOSED	Fixed
DS-1818[792]	Ensure DSpace works with Creative Commons 4.0 licenses[793]		Dec 02, 2013	Aug 24, 2018	Unassigned	Tim Donohue		CLOSED	Fixed
DS-3245[794]	CSV linebreaks not supported by Bulkedit[795]		Jun 15, 2016	Aug 17, 2018	Unassigned	None		CLOSED	Fixed
DS-3447[796]	Transition ORCID integration to ORCID API 2.0[797]		Jan 11, 2017	Aug 29, 2018	Kevin Van de Velde (Atmire)	Bram Luyten (Atmire)		CLOSED	Fixed
DS-3560[798]	MathJax - Mirage MathJax depends on CDN that is being shut down[799]		Apr 10, 2017	Jun 22, 2018	Kim Shepherd	Tom Shorock		CLOSED	Fixed
DS-3705[800]	Recent Submissions in Reference theme completely covered up by navigation[801]		Oct 04, 2017	Jun 22, 2018	Ivan Masár	Ivan Masár		CLOSED	Fixed
DS-3757[802]	Default setting for ClamAV Socket Timeout is too short[803]		Nov 16, 2017	Nov 27, 2017	Unassigned	Hardy Pottinger		CLOSED	Fixed
DS-3832[804]	GeoIP-API(com.maxmind.geoip:geoip-api) needs to be replaced by GeoIP2 (com.maxmind.geoip2:geoip2)[805]		Feb 07, 2018	Jun 22, 2018	Mark H. Wood	Sven Soliman		CLOSED	Fixed

790 https://jira.lyrasis.org/browse/DS-950?src=confmacro
791 https://jira.lyrasis.org/browse/DS-950?src=confmacro
792 https://jira.lyrasis.org/browse/DS-1818?src=confmacro
793 https://jira.lyrasis.org/browse/DS-1818?src=confmacro
794 https://jira.lyrasis.org/browse/DS-3245?src=confmacro
795 https://jira.lyrasis.org/browse/DS-3245?src=confmacro
796 https://jira.lyrasis.org/browse/DS-3447?src=confmacro
797 https://jira.lyrasis.org/browse/DS-3447?src=confmacro
798 https://jira.lyrasis.org/browse/DS-3560?src=confmacro
799 https://jira.lyrasis.org/browse/DS-3560?src=confmacro
800 https://jira.lyrasis.org/browse/DS-3705?src=confmacro
801 https://jira.lyrasis.org/browse/DS-3705?src=confmacro
802 https://jira.lyrasis.org/browse/DS-3757?src=confmacro
803 https://jira.lyrasis.org/browse/DS-3757?src=confmacro
804 https://jira.lyrasis.org/browse/DS-3832?src=confmacro
805 https://jira.lyrasis.org/browse/DS-3832?src=confmacro

DS-3835[806]	XMLUI Search Results Lose Collection/Community Context when Sorting with the Gear Icon[807]		Feb 08, 2018	Jun 22, 2018	Terrence W Brady	Terrence W Brady		CLOSED	Fixed
DS-3839[808]	Imagemagic thumbnails have the wrong orientation[809]		Feb 15, 2018	Feb 20, 2018	Hardy Pottinger	Hardy Pottinger		CLOSED	Fixed
DS-3840[810]	CSV Metadata exports ignores metadata.hide.* and exposes sensitive metadata[811]		Feb 16, 2018	Jun 27, 2018	Kim Shepherd	Eike Kleiner		CLOSED	Fixed
DS-3852[812]	OAI indexer message not helpful in locating problems[813]		Feb 24, 2018	Feb 26, 2018	Mark H. Wood	Mark H. Wood		CLOSED	Fixed
DS-3853[814]	PMH Responder returns 500 Internal Error if item is in no Community or no Collection[815]		Feb 24, 2018	Mar 18, 2018	Mark H. Wood	Mark H. Wood		CLOSED	Fixed
DS-3854[816]	DSpace 5 PostgreSQL JDBC Driver doesn't work well with Postgres 10[817]		Feb 26, 2018	Aug 15, 2019	Tim Donohue	Tim Donohue		CLOSED	Fixed
DS-3866[818]	JSPUI EPerson name and Group name JS injection vulnerability[819]		Mar 09, 2018	Jun 30, 2018	Kim Shepherd	Julio Brafman		CLOSED	Fixed
DS-3883[820]	DSpace 6x Performance Issues for Items with 100+ bitstreams[821]		Apr 04, 2018	Jun 24, 2019	Tim Donohue	Ying Jin		CLOSED	Fixed
DS-3886[822]	Tests in DSpace 5 fail if either versioning and/or DOIIdentifierProvider are activated[823]		Apr 12, 2018	Jun 25, 2018	Pascal-Nicolas Becker	Pascal-Nicolas Becker		CLOSED	Fixed

806 https://jira.lyrasis.org/browse/DS-3835?src=confmacro
807 https://jira.lyrasis.org/browse/DS-3835?src=confmacro
808 https://jira.lyrasis.org/browse/DS-3839?src=confmacro
809 https://jira.lyrasis.org/browse/DS-3839?src=confmacro
810 https://jira.lyrasis.org/browse/DS-3840?src=confmacro
811 https://jira.lyrasis.org/browse/DS-3840?src=confmacro
812 https://jira.lyrasis.org/browse/DS-3852?src=confmacro
813 https://jira.lyrasis.org/browse/DS-3852?src=confmacro
814 https://jira.lyrasis.org/browse/DS-3853?src=confmacro
815 https://jira.lyrasis.org/browse/DS-3853?src=confmacro
816 https://jira.lyrasis.org/browse/DS-3854?src=confmacro
817 https://jira.lyrasis.org/browse/DS-3854?src=confmacro
818 https://jira.lyrasis.org/browse/DS-3866?src=confmacro
819 https://jira.lyrasis.org/browse/DS-3866?src=confmacro
820 https://jira.lyrasis.org/browse/DS-3883?src=confmacro
821 https://jira.lyrasis.org/browse/DS-3883?src=confmacro
822 https://jira.lyrasis.org/browse/DS-3886?src=confmacro
823 https://jira.lyrasis.org/browse/DS-3886?src=confmacro

DS-3936[824]	Bower registry used by Mirage2 is no longer serving requests[825]		Jun 24, 2018	Jun 25, 2018	Kim Shepherd	Kim Shepherd		CLOSED	Fixed

18 issues[826]

7.6.3.2 Changes in DSpace 5.8

Key	Summary	T	Created	Updated	Due	Assignee	Reporter	P	Status	Resolution
DS-1818[827]	Ensure DSpace works with Creative Commons 4.0 licenses[828]		Dec 02, 2013	Aug 24, 2018		Unassigned	Tim Donohue		CLOSED	Fixed
DS-3661[829]	ImageMagick PDF Processing Degraded with Changes in 5.7 release[830]		Aug 02, 2017	Sep 06, 2017		Unassigned	Terrence W Brady		CLOSED	Fixed
DS-3674[831]	PluginServiceTest.java does not have a provided testing config and thus tests a live config[832]		Aug 15, 2017	Aug 15, 2017		Hardy Pottinger	Hardy Pottinger		CLOSED	Fixed

3 issues[833]

7.6.3.3 Changes in DSpace 5.7

Key	Summary	T	Created	Updated	Due	Assignee	Reporter	P	Status	Resolution

824 https://jira.lyrasis.org/browse/DS-3936?src=confmacro
825 https://jira.lyrasis.org/browse/DS-3936?src=confmacro
62 https://jira.lyrasis.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project+%3D+DS+AND+resolution+%3D+Fixed+AND+fixVersion+%3D+%225.9%22+ORDER+BY+key+ASC++++++++++++++&src=confmacro
827 https://jira.lyrasis.org/browse/DS-1818?src=confmacro
828 https://jira.lyrasis.org/browse/DS-1818?src=confmacro
829 https://jira.lyrasis.org/browse/DS-3661?src=confmacro
830 https://jira.lyrasis.org/browse/DS-3661?src=confmacro
831 https://jira.lyrasis.org/browse/DS-3674?src=confmacro
832 https://jira.lyrasis.org/browse/DS-3674?src=confmacro
83 https://jira.lyrasis.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project+%3D+DS+AND+resolution+%3D+Fixed+AND+fixVersion+%3D+%225.8%22+ORDER+BY+key+ASC++++++++++++++&src=confmacro

DS-1818[834]	Ensure DSpace works with Creative Commons 4.0 licenses[835]		Dec 02, 2013	Aug 24, 2018	Unassigned	Tim Donohue		CLOSED	Fixed
DS-2227[836]	failed AIP import leaves files in assetstore[837]		Oct 29, 2014	May 11, 2017	Hardy Pottinger	Ivan Masár		CLOSED	Fixed
DS-2359[838]	Error when depositing large files via browser (over 2Gb)[839]		Dec 11, 2014	Jul 07, 2017	Unassigned	Pauline Ward		CLOSED	Fixed
DS-2748[840]	Cocoon logs overly verbose when 404s occur[841]		Sep 10, 2015	Jun 22, 2017	Unassigned	Chris Wilper		CLOSED	Fixed
DS-2840[842]	Changes INFO level sidebar facet transformer log entries to DEBUG[843]		Oct 25, 2015	Feb 18, 2017	Bram Luyten (Atmire)	Bram Luyten (Atmire)		CLOSED	Fixed
DS-3281[844]	Submission made via REST API does not trigger collection workflow approval process.[845]		Aug 03, 2016	May 31, 2017	Unassigned	Emilio Lorenzo (Arvo)		CLOSED	Fixed
DS-3289[846]	Mirage 2 with Jetty 9 has broken image links[847]		Aug 12, 2016	Jan 09, 2017	Bram Luyten (Atmire)	Ilja Sidoroff		CLOSED	Fixed
DS-3356[848]	DatabaseUtils should turn off the authz system[849]		Oct 07, 2016	Feb 15, 2017	Luigi Andrea Pascarelli (4Science)	Luigi Andrea Pascarelli (4Science)		CLOSED	Fixed

834 https://jira.lyrasis.org/browse/DS-1818?src=confmacro
835 https://jira.lyrasis.org/browse/DS-1818?src=confmacro
836 https://jira.lyrasis.org/browse/DS-2227?src=confmacro
837 https://jira.lyrasis.org/browse/DS-2227?src=confmacro
838 https://jira.lyrasis.org/browse/DS-2359?src=confmacro
839 https://jira.lyrasis.org/browse/DS-2359?src=confmacro
840 https://jira.lyrasis.org/browse/DS-2748?src=confmacro
841 https://jira.lyrasis.org/browse/DS-2748?src=confmacro
842 https://jira.lyrasis.org/browse/DS-2840?src=confmacro
843 https://jira.lyrasis.org/browse/DS-2840?src=confmacro
844 https://jira.lyrasis.org/browse/DS-3281?src=confmacro
845 https://jira.lyrasis.org/browse/DS-3281?src=confmacro
846 https://jira.lyrasis.org/browse/DS-3289?src=confmacro
847 https://jira.lyrasis.org/browse/DS-3289?src=confmacro
848 https://jira.lyrasis.org/browse/DS-3356?src=confmacro
849 https://jira.lyrasis.org/browse/DS-3356?src=confmacro

DS-3366[850]	/handleresolver provides no data[851]		Oct 24, 2016	Mar 08, 2017	Unassigned	Peter Dietz		CLOSED	Fixed
DS-3405[852]	bad typo in Messages_de.properties in jsp.search.facet.refine.previous[853]		Nov 28, 2016	Nov 28, 2016	Claudia Jürgen	Georg Bastian		CLOSED	Fixed
DS-3415[854]	administrative.js doEditCommunity wrong parameter name[855]		Dec 07, 2016	Feb 21, 2017	Unassigned	Samuel Cambien (Atmire)		CLOSED	Fixed
DS-3425[856]	outputstream gets closed in JSONDiscoverySearcher[857]		Dec 16, 2016	Feb 21, 2017	Unassigned	Samuel Cambien (Atmire)		CLOSED	Fixed
DS-3431[858]	BasicWorkflow system is vulnerable to unauthorized manipulations[859]		Dec 28, 2016	Jul 12, 2017	Pascal-Nicolas Becker	Pascal-Nicolas Becker		CLOSED	Fixed
DS-3436[860]	Statistics Shard Corrupts owningComm field[861]		Jan 03, 2017	Feb 09, 2017	Terrence W Brady	Terrence W Brady		CLOSED	Fixed
DS-3441[862]	READ permssion on the Collection object not respected by the JSPUI[863]		Jan 05, 2017	Jan 06, 2017	Andrea Bollini (4Science)	Andrea Bollini (4Science)		CLOSED	Fixed
DS-3448[864]	Multi-Select in submission page[865]		Jan 11, 2017	Feb 22, 2017	Unassigned	Jonas Van Goolen (Atmire)		CLOSED	Fixed
DS-3456[866]	solr-export-statistics / solr-import-statistics inconsistencies[867]		Jan 13, 2017	May 23, 2019	Terrence W Brady	Terrence W Brady		CLOSED	Fixed

850 https://jira.lyrasis.org/browse/DS-3366?src=confmacro
851 https://jira.lyrasis.org/browse/DS-3366?src=confmacro
852 https://jira.lyrasis.org/browse/DS-3405?src=confmacro
853 https://jira.lyrasis.org/browse/DS-3405?src=confmacro
854 https://jira.lyrasis.org/browse/DS-3415?src=confmacro
855 https://jira.lyrasis.org/browse/DS-3415?src=confmacro
856 https://jira.lyrasis.org/browse/DS-3425?src=confmacro
857 https://jira.lyrasis.org/browse/DS-3425?src=confmacro
858 https://jira.lyrasis.org/browse/DS-3431?src=confmacro
859 https://jira.lyrasis.org/browse/DS-3431?src=confmacro
860 https://jira.lyrasis.org/browse/DS-3436?src=confmacro
861 https://jira.lyrasis.org/browse/DS-3436?src=confmacro
862 https://jira.lyrasis.org/browse/DS-3441?src=confmacro
863 https://jira.lyrasis.org/browse/DS-3441?src=confmacro
864 https://jira.lyrasis.org/browse/DS-3448?src=confmacro
865 https://jira.lyrasis.org/browse/DS-3448?src=confmacro
866 https://jira.lyrasis.org/browse/DS-3456?src=confmacro
867 https://jira.lyrasis.org/browse/DS-3456?src=confmacro

DS-3458[868]	Allow Shard Resumption - Append to an existing shard if present[869]		Jan 13, 2017	Feb 09, 2017	Terrence W Brady	Terrence W Brady		CLOSED	Fixed
DS-3468[870]	Add "bin/" to gitignore resources for Eclipse[871]		Jan 25, 2017	Feb 03, 2017	Terrence W Brady	Terrence W Brady		CLOSED	Fixed
DS-3479[872]	Item Import SAF might create empty metadata values[873]		Feb 03, 2017	May 03, 2017	Claudia Jürgen	Claudia Jürgen		CLOSED	Fixed

Showing 20 out of 32 issues[874]

7.6.3.4 Changes in DSpace 5.6

Key	Summary	T	Created	Updated	Due	Assignee	Reporter	P	Status	Resolution
DS-1818[875]	Ensure DSpace works with Creative Commons 4.0 licenses[876]		Dec 02, 2013	Aug 24, 2018		Unassigned	Tim Donohue		CLOSED	Fixed
DS-2604[877]	Creative Commons license assignment silently fails in JSPUI[878]		Jun 05, 2015	Nov 11, 2016		Luigi Andrea Pascarelli (4Science)	Dan Ishimitsu		CLOSED	Fixed
DS-2623[879]	Integrity error on file uploads[880]		Jun 18, 2015	Oct 05, 2016		Pascal-Nicolas Becker	Eike Kleiner		CLOSED	Fixed

868 https://jira.lyrasis.org/browse/DS-3458?src=confmacro
869 https://jira.lyrasis.org/browse/DS-3458?src=confmacro
870 https://jira.lyrasis.org/browse/DS-3468?src=confmacro
871 https://jira.lyrasis.org/browse/DS-3468?src=confmacro
872 https://jira.lyrasis.org/browse/DS-3479?src=confmacro
873 https://jira.lyrasis.org/browse/DS-3479?src=confmacro
874 https://jira.lyrasis.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project+%3D+DS+AND+resolution+%3D+Fixed+AND+fixVersion+%3D+%225.7%22+ORDER+BY+key+ASC++++++++++++&src=confmacro
875 https://jira.lyrasis.org/browse/DS-1818?src=confmacro
876 https://jira.lyrasis.org/browse/DS-1818?src=confmacro
877 https://jira.lyrasis.org/browse/DS-2604?src=confmacro
878 https://jira.lyrasis.org/browse/DS-2604?src=confmacro
879 https://jira.lyrasis.org/browse/DS-2623?src=confmacro
880 https://jira.lyrasis.org/browse/DS-2623?src=confmacro

DS-2702[881]	Cannot send email using SSL[882]		Aug 13, 2015	Oct 13, 2016	Bram Luyten (Atmire)	Roeland Dillen		CLOSED	Fixed
DS-2874[883]	error when missing Context Description in xoai.xml[884]		Nov 11, 2015	Sep 30, 2016	Ivan Masár	Ivan Masár		CLOSED	Fixed
DS-2895[885]	Any registered user can modify inprogress submission[886]		Nov 18, 2015	Oct 13, 2016	Andrea Bollini (4Science)	Andrea Bollini (4Science)		CLOSED	Fixed
DS-3097[887]	Bitstreams of embargoed and/or withdrawn items can be accessed by anyone[888]		Mar 10, 2016	Dec 08, 2016	Andrea Bollini (4Science)	Mark H. Wood		CLOSED	Fixed
DS-3140[889]	METSRightsCrosswalk NPE During AIP Restore - No Anonymous Read[890]		Apr 21, 2016	Aug 17, 2016	Unassigned	Michael Marttila		CLOSED	Fixed
DS-3206[891]	Policy form merge field values when perform group search[892]		May 06, 2016	Aug 11, 2016	Ivan Masár	Oriol		CLOSED	Fixed
DS-3246[893]	Recyclable Cocoon components should clear local variables[894]		Jun 15, 2016	Dec 16, 2016	Hardy Pottinger	Andrea Schweer		CLOSED	Fixed
DS-3248[895]	expand parameter was not passed during new item creation when accessing find-by-metadata-field[896]		Jun 22, 2016	Sep 30, 2016	Unassigned	Terrence W Brady		CLOSED	Fixed

881 https://jira.lyrasis.org/browse/DS-2702?src=confmacro
882 https://jira.lyrasis.org/browse/DS-2702?src=confmacro
883 https://jira.lyrasis.org/browse/DS-2874?src=confmacro
884 https://jira.lyrasis.org/browse/DS-2874?src=confmacro
885 https://jira.lyrasis.org/browse/DS-2895?src=confmacro
886 https://jira.lyrasis.org/browse/DS-2895?src=confmacro
887 https://jira.lyrasis.org/browse/DS-3097?src=confmacro
888 https://jira.lyrasis.org/browse/DS-3097?src=confmacro
889 https://jira.lyrasis.org/browse/DS-3140?src=confmacro
890 https://jira.lyrasis.org/browse/DS-3140?src=confmacro
891 https://jira.lyrasis.org/browse/DS-3206?src=confmacro
892 https://jira.lyrasis.org/browse/DS-3206?src=confmacro
893 https://jira.lyrasis.org/browse/DS-3246?src=confmacro
894 https://jira.lyrasis.org/browse/DS-3246?src=confmacro
895 https://jira.lyrasis.org/browse/DS-3248?src=confmacro
896 https://jira.lyrasis.org/browse/DS-3248?src=confmacro

DS-3250[897]	SQL Injection Vulnerability in 5.x REST API[898]		Jun 27, 2016	Oct 13, 2016	Unassigned	Bram Luyten (Atmire)	CLOSED	Fixed
DS-3266[899]	AIP Restore is not respecting access restrictions (on Items)[900]		Jul 14, 2016	Aug 17, 2016	Tim Donohue	Tim Donohue	CLOSED	Fixed
DS-3294[901]	Request a copy for "all restricted files" not working.[902]		Aug 19, 2016	Aug 25, 2016	Tim Donohue	Bill Tantzen	CLOSED	Fixed
DS-3309[903]	XML External Entity (XXE) vulnerability in pdfbox[904]		Sep 07, 2016	Oct 13, 2016	Unassigned	Seth Robbins	CLOSED	Fixed
DS-3326[905]	XMLUI: Creative Commons store a wrong xml for rdf license[906]		Sep 16, 2016	May 03, 2021	Luigi Andrea Pascarelli (4Science)	Luigi Andrea Pascarelli (4Science)	CLOSED	Fixed
DS-3340[907]	Some test fails during the preparation of DSpace 5.6 release[908]		Sep 26, 2016	Sep 26, 2016	Luigi Andrea Pascarelli (4Science)	Luigi Andrea Pascarelli (4Science)	CLOSED	Fixed
DS-3347[909]	Authorization denied for Anonymous access to restricted bistream[910]		Sep 29, 2016	Oct 11, 2016	Luigi Andrea Pascarelli (4Science)	Ivan Masár	CLOSED	Fixed

18 issues[911]

897 https://jira.lyrasis.org/browse/DS-3250?src=confmacro
898 https://jira.lyrasis.org/browse/DS-3250?src=confmacro
899 https://jira.lyrasis.org/browse/DS-3266?src=confmacro
900 https://jira.lyrasis.org/browse/DS-3266?src=confmacro
901 https://jira.lyrasis.org/browse/DS-3294?src=confmacro
902 https://jira.lyrasis.org/browse/DS-3294?src=confmacro
903 https://jira.lyrasis.org/browse/DS-3309?src=confmacro
904 https://jira.lyrasis.org/browse/DS-3309?src=confmacro
905 https://jira.lyrasis.org/browse/DS-3326?src=confmacro
906 https://jira.lyrasis.org/browse/DS-3326?src=confmacro
907 https://jira.lyrasis.org/browse/DS-3340?src=confmacro
908 https://jira.lyrasis.org/browse/DS-3340?src=confmacro
909 https://jira.lyrasis.org/browse/DS-3347?src=confmacro
910 https://jira.lyrasis.org/browse/DS-3347?src=confmacro
911 https://jira.lyrasis.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project+%3D+DS+AND+issuetype+%3D+Bug+AND+resolution+%3D+Fixed+AND+fixVersion+%3D+%225.6%22+ORDER+BY+key+ASC++++++++++&src=confmacro

7.6.3.5 Changes in DSpace 5.5

Key	Summary	T	Created	Updated	Due	Assignee	Reporter	P	Status	Resolution
DS-1818[912]	Ensure DSpace works with Creative Commons 4.0 licenses[913]		Dec 02, 2013	Aug 24, 2018		Unassigned	Tim Donohue		CLOSED	Fixed
DS-2517[914]	Item.findByMetadataFieldAuthority wrong sql query[915]		Mar 18, 2015	Feb 15, 2016		Ivan Masár	Ondřej Košarko		CLOSED	Fixed
DS-2820[916]	"NOT" XOAI filter not working for SOLR configuration[917]		Oct 16, 2015	Feb 11, 2016		Ivan Masár	Patricio Marrone		CLOSED	Fixed
DS-2893[918]	Mirage2: printing an item page includes the URL to the bitstreams[919]		Nov 17, 2015	Aug 01, 2016		Hardy Pottinger	Hardy Pottinger		CLOSED	Fixed
DS-2923[920]	Update DataCite default configuration[921]		Nov 27, 2015	Nov 27, 2015		Pascal-Nicolas Becker	Pascal-Nicolas Becker		CLOSED	Fixed
DS-2936[922]	REST-API /handle endpoint broken[923]		Dec 06, 2015	Dec 15, 2015		Bram Luyten (Atmire)	Bram Luyten (Atmire)		CLOSED	Fixed
DS-2946[924]	Missing DSpaceWebapp in dspace-rest module[925]		Dec 10, 2015	Dec 11, 2015		Bram Luyten (Atmire)	Bram Luyten (Atmire)		CLOSED	Fixed

912 https://jira.lyrasis.org/browse/DS-1818?src=confmacro
913 https://jira.lyrasis.org/browse/DS-1818?src=confmacro
914 https://jira.lyrasis.org/browse/DS-2517?src=confmacro
915 https://jira.lyrasis.org/browse/DS-2517?src=confmacro
916 https://jira.lyrasis.org/browse/DS-2820?src=confmacro
917 https://jira.lyrasis.org/browse/DS-2820?src=confmacro
918 https://jira.lyrasis.org/browse/DS-2893?src=confmacro
919 https://jira.lyrasis.org/browse/DS-2893?src=confmacro
920 https://jira.lyrasis.org/browse/DS-2923?src=confmacro
921 https://jira.lyrasis.org/browse/DS-2923?src=confmacro
922 https://jira.lyrasis.org/browse/DS-2936?src=confmacro
923 https://jira.lyrasis.org/browse/DS-2936?src=confmacro
924 https://jira.lyrasis.org/browse/DS-2946?src=confmacro
925 https://jira.lyrasis.org/browse/DS-2946?src=confmacro

DS-2998[926]	Incorrect metadata element "dcterms.comformsTo" in dspace registry configuration[927]		Jan 21, 2016	Jan 22, 2016	Ivan Masár	Jing Pu		CLOSED	Fixed
DS-3050[928]	XOAI wrong URL encoding[929]		Feb 08, 2016	Feb 09, 2016	Ivan Masár	Claudia Jürgen		CLOSED	Fixed
DS-3063[930]	JSPUI Edit News feature can be used to view/edit other files readable to Tomcat user[931]		Feb 15, 2016	May 16, 2016	Andrea Bollini (4Science)	Tim Donohue		CLOSED	Fixed
DS-3085[932]	Sherpa/Romeo ungraded journal (gray) shows error[933]		Feb 29, 2016	Feb 29, 2016	Ivan Masár	Jing Pu		CLOSED	Fixed
DS-3094[934]	XMLUI Directory Traversal Vulnerability in Themes[935]		Mar 07, 2016	May 16, 2016	Tim Donohue	Tim Donohue		CLOSED	Fixed

12 issues[936]

7.6.3.6 Changes in DSpace 5.4

Key	Summary	T	Created	Updated	Due	Assignee	Reporter	P	Status	Resolution
DS-1207[937]	ResourceNotFoundException on redirect[938]		Jul 05, 2012	Nov 05, 2015		Unassigned	Nestor Oviedo		CLOSED	Fixed

926 https://jira.lyrasis.org/browse/DS-2998?src=confmacro
927 https://jira.lyrasis.org/browse/DS-2998?src=confmacro
928 https://jira.lyrasis.org/browse/DS-3050?src=confmacro
929 https://jira.lyrasis.org/browse/DS-3050?src=confmacro
930 https://jira.lyrasis.org/browse/DS-3063?src=confmacro
931 https://jira.lyrasis.org/browse/DS-3063?src=confmacro
932 https://jira.lyrasis.org/browse/DS-3085?src=confmacro
933 https://jira.lyrasis.org/browse/DS-3085?src=confmacro
934 https://jira.lyrasis.org/browse/DS-3094?src=confmacro
935 https://jira.lyrasis.org/browse/DS-3094?src=confmacro
936 https://jira.lyrasis.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project+%3D+DS+AND+issuetype+%3D+Bug+AND+resolution+%3D+Fixed+AND+fixVersion+%3D+%225.5%22+ORDER+BY+key+ASC++++++++++&src=confmacro
937 https://jira.lyrasis.org/browse/DS-1207?src=confmacro
938 https://jira.lyrasis.org/browse/DS-1207?src=confmacro

DS-1818[939]	Ensure DSpace works with Creative Commons 4.0 licenses[940]		Dec 02, 2013	Aug 24, 2018	Unassigned	Tim Donohue	CLOSED	Fixed
DS-1924[941]	currentLocale in DRI document is not changed[942]		Feb 23, 2014	Oct 06, 2015	Andrea Schweer	Supasate Choochaisri	CLOSED	Fixed
DS-2408[943]	Cannot create administrator in fresh install - no admin group found[944]		Jan 13, 2015	Nov 02, 2015	Tim Donohue	Andrea Schweer	CLOSED	Fixed
DS-2502[945]	Incorrect dependencies drag javax.servlet:servlet-api into all webapp.s[946]		Mar 09, 2015	Oct 16, 2015	Mark H. Wood	Mark H. Wood	CLOSED	Fixed
DS-2524[947]	XOAI - from/until argument only works with date including the time[948]		Mar 25, 2015	Nov 02, 2015	Unassigned	Christian Scheible	CLOSED	Fixed
DS-2533[949]	Mirage 2 doesn't work with Maven 3.3.x[950]		Apr 01, 2015	Jul 20, 2021	Bram Luyten (Atmire)	Andrea Schweer	CLOSED	Fixed
DS-2542[951]	XOAI does not support non granular YYYY-MM-DD harvesting properly[952]		Apr 10, 2015	Nov 04, 2015	Tim Donohue	Claudia Jürgen	CLOSED	Fixed
DS-2573[953]	OAI Item Record contains all virtual sets independent if the item is in the set or not[954]		May 13, 2015	Nov 03, 2015	Unassigned	Christian Scheible	CLOSED	Fixed
DS-2591[955]	Wrong path to respond.min.js in Mirage 2 page-structure.xsl[956]		May 26, 2015	Oct 16, 2015	Ivan Masár	Andrea Schweer	CLOSED	Fixed

939 https://jira.lyrasis.org/browse/DS-1818?src=confmacro
940 https://jira.lyrasis.org/browse/DS-1818?src=confmacro
941 https://jira.lyrasis.org/browse/DS-1924?src=confmacro
942 https://jira.lyrasis.org/browse/DS-1924?src=confmacro
943 https://jira.lyrasis.org/browse/DS-2408?src=confmacro
944 https://jira.lyrasis.org/browse/DS-2408?src=confmacro
945 https://jira.lyrasis.org/browse/DS-2502?src=confmacro
946 https://jira.lyrasis.org/browse/DS-2502?src=confmacro
947 https://jira.lyrasis.org/browse/DS-2524?src=confmacro
948 https://jira.lyrasis.org/browse/DS-2524?src=confmacro
949 https://jira.lyrasis.org/browse/DS-2533?src=confmacro
950 https://jira.lyrasis.org/browse/DS-2533?src=confmacro
951 https://jira.lyrasis.org/browse/DS-2542?src=confmacro
952 https://jira.lyrasis.org/browse/DS-2542?src=confmacro
953 https://jira.lyrasis.org/browse/DS-2573?src=confmacro
954 https://jira.lyrasis.org/browse/DS-2573?src=confmacro
955 https://jira.lyrasis.org/browse/DS-2591?src=confmacro
956 https://jira.lyrasis.org/browse/DS-2591?src=confmacro

DS-2592[957]	Mirage 2 - wrong URL for OpenSearch description.xml[958]		May 26, 2015	Feb 21, 2017	Ivan Masár	Andrea Schweer		CLOSED	Fixed
DS-2655[959]	REST api collection items 'offset' does not function correctly[960]		Jul 14, 2015	Nov 04, 2015	Unassigned	Ed Goulet		CLOSED	Fixed
DS-2679[961]	Google Scholar ordering of metadata tags with multiple values (like authors) broken[962]		Jul 28, 2015	Oct 13, 2015	Bram Luyten (Atmire)	Christian Scheible		CLOSED	Fixed
DS-2692[963]	REST-API /handle not reflecting updates[964]		Aug 03, 2015	Dec 06, 2015	Peter Dietz	Ondřej Košarko		CLOSED	Fixed
DS-2698[965]	XMLUI metadata browse cache validity doesn't consider value count[966]		Aug 11, 2015	Oct 28, 2015	Unassigned	Andrea Schweer		CLOSED	Fixed
DS-2699[967]	Search not working as expected[968]		Aug 12, 2015	Nov 04, 2015	Andrea Schweer	Andrea Schweer		CLOSED	Fixed
DS-2706[969]	Solr core "authority" spews failure log entries for libraries not used[970]		Aug 18, 2015	Aug 01, 2016	Hardy Pottinger	Mark H. Wood		CLOSED	Fixed
DS-2719[971]	rest /colections/<id>/items ignores offset parameter[972]		Aug 24, 2015	Nov 04, 2015	Unassigned	Monika Mevenkamp		CLOSED	Fixed
DS-2733[973]	Erroneous String Compare[974]		Aug 31, 2015	Oct 28, 2015	Tim Donohue	KimKM		CLOSED	Fixed

957 https://jira.lyrasis.org/browse/DS-2592?src=confmacro
958 https://jira.lyrasis.org/browse/DS-2592?src=confmacro
959 https://jira.lyrasis.org/browse/DS-2655?src=confmacro
960 https://jira.lyrasis.org/browse/DS-2655?src=confmacro
961 https://jira.lyrasis.org/browse/DS-2679?src=confmacro
962 https://jira.lyrasis.org/browse/DS-2679?src=confmacro
963 https://jira.lyrasis.org/browse/DS-2692?src=confmacro
964 https://jira.lyrasis.org/browse/DS-2692?src=confmacro
965 https://jira.lyrasis.org/browse/DS-2698?src=confmacro
966 https://jira.lyrasis.org/browse/DS-2698?src=confmacro
967 https://jira.lyrasis.org/browse/DS-2699?src=confmacro
968 https://jira.lyrasis.org/browse/DS-2699?src=confmacro
969 https://jira.lyrasis.org/browse/DS-2706?src=confmacro
970 https://jira.lyrasis.org/browse/DS-2706?src=confmacro
971 https://jira.lyrasis.org/browse/DS-2719?src=confmacro
972 https://jira.lyrasis.org/browse/DS-2719?src=confmacro
973 https://jira.lyrasis.org/browse/DS-2733?src=confmacro
974 https://jira.lyrasis.org/browse/DS-2733?src=confmacro

DS-2736[975]	XSS in JSPUI search form[976]		Sep 02, 2015	May 16, 2016	Tim Donohue	Genaro Contreras		CLOSED	Fixed

Showing 20 out of 33 issues[977]

7.6.3.7 Changes in DSpace 5.3

Key	Summary	T	Created	Updated	Due	Assignee	Reporter	P	Status	Resolution
DS-1818[978]	Ensure DSpace works with Creative Commons 4.0 licenses[979]		Dec 02, 2013	Aug 24, 2018		Unassigned	Tim Donohue		CLOSED	Fixed
DS-2339[980]	Searching produces ParseException with Lucene special char as input[981]		Dec 01, 2014	Jul 15, 2015		Tim Donohue	Sean Xiao		CLOSED	Fixed
DS-2358[982]	Item-level versioning discards embargo[983]		Dec 10, 2014	Jul 28, 2015		Pascal-Nicolas Becker	Andrea Schweer		CLOSED	Fixed
DS-2461[984]	Discovery Search NPE or Solr SyntaxError if search string contains a colon[985]		Feb 20, 2015	Jul 15, 2015		Tim Donohue	Raul Ruiz		CLOSED	Fixed
DS-2472[986]	Simple Search with two colons throws NullPointerException[987]		Feb 25, 2015	Jul 15, 2015		Tim Donohue	Oliver Goldschmidt		CLOSED	Fixed

975 https://jira.lyrasis.org/browse/DS-2736?src=confmacro
976 https://jira.lyrasis.org/browse/DS-2736?src=confmacro
977 https://jira.lyrasis.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project+%3D+DS+AND+issuetype+%3D+Bug+AND+resolution+%3D+Fixed+AND+fixVersion+%3D+%225.4%22+ORDER+BY+key+ASC++++++++&src=confmacro
978 https://jira.lyrasis.org/browse/DS-1818?src=confmacro
979 https://jira.lyrasis.org/browse/DS-1818?src=confmacro
980 https://jira.lyrasis.org/browse/DS-2339?src=confmacro
981 https://jira.lyrasis.org/browse/DS-2339?src=confmacro
982 https://jira.lyrasis.org/browse/DS-2358?src=confmacro
983 https://jira.lyrasis.org/browse/DS-2358?src=confmacro
984 https://jira.lyrasis.org/browse/DS-2461?src=confmacro
985 https://jira.lyrasis.org/browse/DS-2461?src=confmacro
986 https://jira.lyrasis.org/browse/DS-2472?src=confmacro
987 https://jira.lyrasis.org/browse/DS-2472?src=confmacro

DS-2543[988]	OAI full import does not clean the cached responses[989]		Apr 13, 2015	Jul 06, 2015	Claudia Jürgen	Claudia Jürgen		CLOSED	Fixed
DS-2554[990]	NPE when indexing harvested items for OAI[991]		Apr 27, 2015	Jul 06, 2015	Tim Donohue	Andrea Schweer		CLOSED	Fixed
DS-2560[992]	Did you mean option missing in Mirage2[993]		Apr 29, 2015	Jul 06, 2015	Bram Luyten (Atmire)	Mini Pillai (@mire)		CLOSED	Fixed
DS-2571[994]	Jump to value in descending browse jumps too far[995]		May 11, 2015	Aug 01, 2016	Hardy Pottinger	Andrea Schweer		CLOSED	Fixed
DS-2587[996]	Resource policies rptype is null after upgrading[997]		May 21, 2015	May 21, 2015	Ivan Masár	Ondřej Košarko		CLOSED	Fixed
DS-2590[998]	Assembly configurations for producing distribution files have multiple issues[999]		May 22, 2015	Aug 01, 2016	Mark H. Wood	Hardy Pottinger		CLOSED	Fixed

988 https://jira.lyrasis.org/browse/DS-2543?src=confmacro
989 https://jira.lyrasis.org/browse/DS-2543?src=confmacro
990 https://jira.lyrasis.org/browse/DS-2554?src=confmacro
991 https://jira.lyrasis.org/browse/DS-2554?src=confmacro
992 https://jira.lyrasis.org/browse/DS-2560?src=confmacro
993 https://jira.lyrasis.org/browse/DS-2560?src=confmacro
994 https://jira.lyrasis.org/browse/DS-2571?src=confmacro
995 https://jira.lyrasis.org/browse/DS-2571?src=confmacro
996 https://jira.lyrasis.org/browse/DS-2587?src=confmacro
997 https://jira.lyrasis.org/browse/DS-2587?src=confmacro
998 https://jira.lyrasis.org/browse/DS-2590?src=confmacro
999 https://jira.lyrasis.org/browse/DS-2590?src=confmacro

DS-2593[1000]	Withdrawn items remain in OAI-PMH until the next full re-import[1001]		May 26, 2015	Aug 07, 2015	Tim Donohue	Tim Donohue	CLOSED	Fixed
DS-2594[1002]	Long file names overlap the second column of item metadata in Mirage 2[1003]		May 27, 2015	Jun 16, 2015	Ivan Masár	Àlex Magaz Graça	CLOSED	Fixed
DS-2598[1004]	OAI-PMH mets format doesn't expose dc.date.available[1005]		Jun 01, 2015	Jun 02, 2015	Ivan Masár	Andrea Schweer	CLOSED	Fixed
DS-2602[1006]	Clicking on a letter when browsing by title or year when browsing by date does not work[1007]		Jun 05, 2015	Jul 15, 2015	Tim Donohue	Anonymous (No Reply)	CLOSED	Fixed
DS-2603[1008]	The citation_pdf_url metadata is null when it shouldn't[1009]		Jun 05, 2015	Jan 04, 2019	Mark H. Wood	Nicolas Schwab (SEDICI)	CLOSED	Fixed
DS-2614[1010]	Expired embargos make bitstreams and items accessible even if they did not pass the workflow (yet)[1011]		Jun 12, 2015	May 16, 2016	Pascal-Nicolas Becker	Pascal-Nicolas Becker	CLOSED	Fixed
DS-2618[1012]	Test Email reports mail sent successfully if mail.server.disabled[1013]		Jun 16, 2015	Jun 16, 2015	Ivan Masár	Roeland Dillen	CLOSED	Fixed
DS-2620[1014]	cocoon misspelled as coocon[1015]		Jun 18, 2015	Jun 20, 2015	Bram Luyten (Atmire)	Ondřej Košarko	CLOSED	Fixed

1000 https://jira.lyrasis.org/browse/DS-2593?src=confmacro
1001 https://jira.lyrasis.org/browse/DS-2593?src=confmacro
1002 https://jira.lyrasis.org/browse/DS-2594?src=confmacro
1003 https://jira.lyrasis.org/browse/DS-2594?src=confmacro
1004 https://jira.lyrasis.org/browse/DS-2598?src=confmacro
1005 https://jira.lyrasis.org/browse/DS-2598?src=confmacro
1006 https://jira.lyrasis.org/browse/DS-2602?src=confmacro
1007 https://jira.lyrasis.org/browse/DS-2602?src=confmacro
1008 https://jira.lyrasis.org/browse/DS-2603?src=confmacro
1009 https://jira.lyrasis.org/browse/DS-2603?src=confmacro
1010 https://jira.lyrasis.org/browse/DS-2614?src=confmacro
1011 https://jira.lyrasis.org/browse/DS-2614?src=confmacro
1012 https://jira.lyrasis.org/browse/DS-2618?src=confmacro
1013 https://jira.lyrasis.org/browse/DS-2618?src=confmacro
1014 https://jira.lyrasis.org/browse/DS-2620?src=confmacro
1015 https://jira.lyrasis.org/browse/DS-2620?src=confmacro

DS-2658[1016]	Wrong mapping for dc metadata in html head[1017]		Jul 17, 2015	Jul 22, 2015	Unassigned	Ondřej Košarko		CLOSED	Fixed

Showing 20 out of 21 issues[1018]

7.6.3.8 Changes in DSpace 5.2

Key	Summary	T	Created	Updated	Due	Assignee	Reporter	P	Status	Resolution
DS-2372[1019]	Export for ORCID authority cache[1020]		Dec 17, 2014	Jul 24, 2019		Unassigned	Bram Luyten (Atmire)		CLOSED	Fixed
DS-2420[1021]	update entity relationship diagram in 5.x docs[1022]		Jan 23, 2015	Apr 23, 2015		Ivan Masár	Ivan Masár		CLOSED	Fixed
DS-2487[1023]	Pre-5 Solr usage stats geo information may get lost when upgrading to 5[1024]		Mar 04, 2015	May 20, 2015		Andrea Schweer	Andrea Schweer		CLOSED	Fixed
DS-2489[1025]	Pre-3 Solr usage stats do not contain uid field[1026]		Mar 04, 2015	May 20, 2015		Unassigned	Andrea Schweer		CLOSED	Fixed

1016 https://jira.lyrasis.org/browse/DS-2658?src=confmacro
1017 https://jira.lyrasis.org/browse/DS-2658?src=confmacro
1018 https://jira.lyrasis.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project+%3D+DS+AND+issuetype+%3D+Bug+AND+resolution+%3D+Fixed+AND+fixVersion+%3D+%225.3%22+ORDER+BY+key+ASC+++++++&src=confmacro
1019 https://jira.lyrasis.org/browse/DS-2372?src=confmacro
1020 https://jira.lyrasis.org/browse/DS-2372?src=confmacro
1021 https://jira.lyrasis.org/browse/DS-2420?src=confmacro
1022 https://jira.lyrasis.org/browse/DS-2420?src=confmacro
1023 https://jira.lyrasis.org/browse/DS-2487?src=confmacro
1024 https://jira.lyrasis.org/browse/DS-2487?src=confmacro
1025 https://jira.lyrasis.org/browse/DS-2489?src=confmacro
1026 https://jira.lyrasis.org/browse/DS-2489?src=confmacro

DS-2509[1027]	Update REST api README.md in GitHub[1028]		Mar 16, 2015	Mar 16, 2015	Ivan Masár	CTU Developers		CLOSED	Fixed
DS-2531[1029]	New entries for the robots user agent list[1030]		Apr 01, 2015	May 08, 2015	Ivan Masár	Bram Luyten (Atmire)		CLOSED	Fixed

6 issues[1031]

Key	Summary	T	Created	Updated	Due	Assignee	Reporter	P	Status	Resolution
DS-1818[1032]	Ensure DSpace works with Creative Commons 4.0 licenses[1033]		Dec 02, 2013	Aug 24, 2018		Unassigned	Tim Donohue		CLOSED	Fixed
DS-1965[1034]	JSPUI Embargo functionality uncompleted[1035]		Apr 07, 2014	May 14, 2015		Pascal-Nicolas Becker	Denis Fdz		CLOSED	Fixed
DS-2020[1036]	NullPointerException in org.dspace.xoai.filter.DSpaceSetSpecFilter[1037]		Jun 05, 2014	May 18, 2015		João Melo	Ondřej Košarko		CLOSED	Fixed

1027 https://jira.lyrasis.org/browse/DS-2509?src=confmacro
1028 https://jira.lyrasis.org/browse/DS-2509?src=confmacro
1029 https://jira.lyrasis.org/browse/DS-2531?src=confmacro
1030 https://jira.lyrasis.org/browse/DS-2531?src=confmacro
1031 https://jira.lyrasis.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project+%3D+DS+AND+issuetype+in+%28Task%2C+Improvement%2C+%22Code+Task%22%2C+Documentation%2C+Sub-task%29+AND+resolution+%3D+Fixed+AND+fixVersion+%3D+%225.2%22+ORDER+BY+key+ASC++++&src=confmacro
1032 https://jira.lyrasis.org/browse/DS-1818?src=confmacro
1033 https://jira.lyrasis.org/browse/DS-1818?src=confmacro
1034 https://jira.lyrasis.org/browse/DS-1965?src=confmacro
1035 https://jira.lyrasis.org/browse/DS-1965?src=confmacro
1036 https://jira.lyrasis.org/browse/DS-2020?src=confmacro
1037 https://jira.lyrasis.org/browse/DS-2020?src=confmacro

DS-2131[1038]	SWORDv2 ingestion fails with NullPointerException[1039]		Sep 04, 2014	May 14, 2015	Kevin Van de Velde (Atmire)	Jan Lievens	CLOSED	Fixed
DS-2186[1040]	Request item copy doesn't always use RequestItemAuthorExtractor[1041]		Oct 10, 2014	May 14, 2015	Unassigned	Àlex Magaz Graça	CLOSED	Fixed
DS-2212[1042]	Statistics Shard not working, version conflict[1043]		Oct 22, 2014	May 20, 2015	Unassigned	Terrence W Brady	CLOSED	Fixed
DS-2218[1044]	Unable to use command "update-handle-prefix"[1045]		Oct 25, 2014	May 08, 2015	Ivan Masár	CTU Developers	CLOSED	Fixed
DS-2379[1046]	command usage output for dspace (from launcher.xml) is unsorted[1047]		Dec 22, 2014	Aug 01, 2016	Mark H. Wood	Hardy Pottinger	CLOSED	Fixed
DS-2403[1048]	RDFConsumer tries to index unpublished WorkspaceItems[1049]		Jan 13, 2015	May 08, 2015	Pascal-Nicolas Becker	Pascal-Nicolas Becker	CLOSED	Fixed
DS-2418[1050]	Incompatible oracle sql method on collection.java (used by REST API)[1051]		Jan 22, 2015	May 08, 2015	Ivan Masár	Raul Ruiz	CLOSED	Fixed
DS-2423[1052]	No longer possibile to create additional Filter for OAI-PMH interface[1053]		Jan 26, 2015	May 12, 2015	João Melo	Christian Scheible	CLOSED	Fixed
DS-2424[1054]	Context Filter not working in OAI interface[1055]		Jan 26, 2015	Apr 22, 2015	João Melo	Christian Scheible	CLOSED	Fixed

1038 https://jira.lyrasis.org/browse/DS-2131?src=confmacro
1039 https://jira.lyrasis.org/browse/DS-2131?src=confmacro
1040 https://jira.lyrasis.org/browse/DS-2186?src=confmacro
1041 https://jira.lyrasis.org/browse/DS-2186?src=confmacro
1042 https://jira.lyrasis.org/browse/DS-2212?src=confmacro
1043 https://jira.lyrasis.org/browse/DS-2212?src=confmacro
1044 https://jira.lyrasis.org/browse/DS-2218?src=confmacro
1045 https://jira.lyrasis.org/browse/DS-2218?src=confmacro
1046 https://jira.lyrasis.org/browse/DS-2379?src=confmacro
1047 https://jira.lyrasis.org/browse/DS-2379?src=confmacro
1048 https://jira.lyrasis.org/browse/DS-2403?src=confmacro
1049 https://jira.lyrasis.org/browse/DS-2403?src=confmacro
1050 https://jira.lyrasis.org/browse/DS-2418?src=confmacro
1051 https://jira.lyrasis.org/browse/DS-2418?src=confmacro
1052 https://jira.lyrasis.org/browse/DS-2423?src=confmacro
1053 https://jira.lyrasis.org/browse/DS-2423?src=confmacro
1054 https://jira.lyrasis.org/browse/DS-2424?src=confmacro
1055 https://jira.lyrasis.org/browse/DS-2424?src=confmacro

DS-2449[1056]	Mirage 2 - Edit Collection - Label item template missing[1057]	Feb 09, 2015	May 13, 2015	Unassigned	Bram Luyten (Atmire)		CLOSED	Fixed
DS-2461[1058]	Discovery Search NPE or Solr SyntaxError if search string contains a colon[1059]	Feb 20, 2015	Jul 15, 2015	Tim Donohue	Raul Ruiz		CLOSED	Fixed
DS-2474[1060]	METS format in OAI includes only the first author[1061]	Feb 25, 2015	May 08, 2015	Ivan Masár	Ivan Masár		CLOSED	Fixed
DS-2482[1062]	Browse and discovery location (community, collection) lost, so one is always browsing the whole site[1063]	Mar 03, 2015	May 08, 2015	Claudia Jürgen	Claudia Jürgen		CLOSED	Fixed
DS-2483[1064]	sword.compatability configuration misspelled in authentication-shibboleth.cfg[1065]	Mar 03, 2015	Mar 05, 2015	Tim Donohue	Tim Donohue		CLOSED	Fixed
DS-2486[1066]	Missing fields in solr statistics data from previous DSpace versions[1067]	Mar 04, 2015	May 21, 2015	Unassigned	Andrea Schweer		CLOSED	Fixed
DS-2491[1068]	advertised OAI deletion mode doesn't correspond to actual mode[1069]	Mar 05, 2015	Feb 03, 2016	Ivan Masár	Ivan Masár		CLOSED	Fixed
DS-2493[1070]	"View more" link is shown even when there aren't more items[1071]	Mar 06, 2015	Mar 06, 2015	Ivan Masár	Àlex Magaz Graça		CLOSED	Fixed

Showing 20 out of 40 issues[1072]

1056 https://jira.lyrasis.org/browse/DS-2449?src=confmacro
1057 https://jira.lyrasis.org/browse/DS-2449?src=confmacro
1058 https://jira.lyrasis.org/browse/DS-2461?src=confmacro
1059 https://jira.lyrasis.org/browse/DS-2461?src=confmacro
1060 https://jira.lyrasis.org/browse/DS-2474?src=confmacro
1061 https://jira.lyrasis.org/browse/DS-2474?src=confmacro
1062 https://jira.lyrasis.org/browse/DS-2482?src=confmacro
1063 https://jira.lyrasis.org/browse/DS-2482?src=confmacro
1064 https://jira.lyrasis.org/browse/DS-2483?src=confmacro
1065 https://jira.lyrasis.org/browse/DS-2483?src=confmacro
1066 https://jira.lyrasis.org/browse/DS-2486?src=confmacro
1067 https://jira.lyrasis.org/browse/DS-2486?src=confmacro
1068 https://jira.lyrasis.org/browse/DS-2491?src=confmacro
1069 https://jira.lyrasis.org/browse/DS-2491?src=confmacro
1070 https://jira.lyrasis.org/browse/DS-2493?src=confmacro
1071 https://jira.lyrasis.org/browse/DS-2493?src=confmacro
1072 https://jira.lyrasis.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project+%3D+DS+AND+issuetype+%3D+Bug+AND+resolution+%3D+Fixed+AND+fixVersion+%3D+%225.2%22+ORDER+BY+key+ASC+++++&src=confmacro

7.6.3.9 Changes in DSpace 5.1

Key	Summary	T	Created	Updated	Due	Assignee	Reporter	P	Status	Resolution
DS-2290[1073]	DSpace5 Estonian translation[1074]		Nov 13, 2014	Feb 19, 2015		Ivan Masár	Heiki Epner		CLOSED	Fixed

1 issue[1075]

Key	Summary	T	Created	Updated	Due	Assignee	Reporter	P	Status	Resolution
DS-2429[1076]	Mirage 2 - No mention of Git prerequisite in documentation[1077]		Jan 28, 2015	Jul 08, 2015		Unassigned	Art Lowel (Atmire)		CLOSED	Fixed

1 issue[1078]

Key	Summary	T	Created	Updated	Due	Assignee	Reporter	P	Status	Resolution
DS-640[1079]	Internal System Error when browsing with wrong argument[1080]		Aug 07, 2010	Nov 10, 2015		Kim Shepherd	Hardik Mishra		CLOSED	Fixed

1073 https://jira.lyrasis.org/browse/DS-2290?src=confmacro
1074 https://jira.lyrasis.org/browse/DS-2290?src=confmacro
1075 https://jira.lyrasis.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project+%3D+DS+AND+issuetype+%3D+%22New+Feature%22+AND+resolution+%3D+Fixed+AND+fixVersion+%3D+%225.1%22+ORDER+BY+key+ASC+&src=confmacro
1076 https://jira.lyrasis.org/browse/DS-2429?src=confmacro
1077 https://jira.lyrasis.org/browse/DS-2429?src=confmacro
1078 https://jira.lyrasis.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project+%3D+DS+AND+issuetype+in+%28Task%2C+Improvement%2C+%22Code+Task%22%2C+Documentation%2C+Sub-task%29+AND+resolution+%3D+Fixed+AND+fixVersion+%3D+%225.1%22+ORDER+BY+key+ASC+&src=confmacro
1079 https://jira.lyrasis.org/browse/DS-640?src=confmacro
1080 https://jira.lyrasis.org/browse/DS-640?src=confmacro

DS-1265[1081]	Sitemap generator does not defend against empty repository[1082]		Sep 13, 2012	Feb 22, 2015	Mark H. Wood	Mark H. Wood	CLOSED	Fixed
DS-1702[1083]	Cross-site scripting (XSS injection) is possible in JSPUI Recent Submissions listings[1084]		Oct 15, 2013	Mar 25, 2015	Luigi Andrea Pascarelli (4Science)	Sean Xiao	CLOSED	Fixed
DS-1818[1085]	Ensure DSpace works with Creative Commons 4.0 licenses[1086]		Dec 02, 2013	Aug 24, 2018	Unassigned	Tim Donohue	CLOSED	Fixed
DS-1896[1087]	XMLUI returns 500 response for most invalid "/static" URLs[1088]		Jan 30, 2014	Feb 24, 2015	Tim Donohue	Tim Donohue	CLOSED	Fixed
DS-2034[1089]	Problem in [Control Panel]->[Dspace Configuration] - org.dspace.app.xmlui.wing.WingInvalidArgument: The 'characters' parameter is required for list items.[1090]		Jun 20, 2014	Aug 01, 2016	Hardy Pottinger	Royopa	CLOSED	Fixed
DS-2044[1091]	Cross-site scripting (XSS injection) is possible in JSPUI Discovery search form[1092]		Jun 30, 2014	Mar 25, 2015	Luigi Andrea Pascarelli (4Science)	Gabriela Mircea	CLOSED	Fixed

1081 https://jira.lyrasis.org/browse/DS-1265?src=confmacro
1082 https://jira.lyrasis.org/browse/DS-1265?src=confmacro
1083 https://jira.lyrasis.org/browse/DS-1702?src=confmacro
1084 https://jira.lyrasis.org/browse/DS-1702?src=confmacro
1085 https://jira.lyrasis.org/browse/DS-1818?src=confmacro
1086 https://jira.lyrasis.org/browse/DS-1818?src=confmacro
1087 https://jira.lyrasis.org/browse/DS-1896?src=confmacro
1088 https://jira.lyrasis.org/browse/DS-1896?src=confmacro
1089 https://jira.lyrasis.org/browse/DS-2034?src=confmacro
1090 https://jira.lyrasis.org/browse/DS-2034?src=confmacro
1091 https://jira.lyrasis.org/browse/DS-2044?src=confmacro
1092 https://jira.lyrasis.org/browse/DS-2044?src=confmacro

DS-2130[1093]	XMLUI allows access to theme XSL files[1094]		Sep 03, 2014	Aug 01, 2016	Tim Donohue	Hardy Pottinger		CLOSED	Fixed
DS-2201[1095]	Unable to complete installation of DSpace with non-empty variable "db.schema" configuration file "build.properties"[1096]		Oct 17, 2014	Jan 28, 2015	Unassigned	CTU Developers		CLOSED	Fixed
DS-2278[1097]	Error pages improperly formatted on certain url patterns[1098]		Nov 11, 2014	Feb 23, 2015	Tim Donohue	Terrence W Brady		CLOSED	Fixed
DS-2355[1099]	bug in SpiderDetector.java[1100]		Dec 10, 2014	Feb 20, 2015	Unassigned	Bill Tantzen		CLOSED	Fixed
DS-2412[1101]	The oai "Show more" link in stylesheet has a fixed verb[1102]		Jan 16, 2015	Jan 20, 2015	Ivan Masár	Ondřej Košarko		CLOSED	Fixed
DS-2415[1103]	cannot add new metadata field to an existing item with Oracle back-end database[1104]		Jan 20, 2015	Aug 01, 2016	Unassigned	Hardy Pottinger		CLOSED	Fixed
DS-2419[1105]	JSP UI ignores authorization.admin.usage[1106]		Jan 23, 2015	Feb 22, 2015	Unassigned	Eike Kleiner		CLOSED	Fixed

1093 https://jira.lyrasis.org/browse/DS-2130?src=confmacro
1094 https://jira.lyrasis.org/browse/DS-2130?src=confmacro
1095 https://jira.lyrasis.org/browse/DS-2201?src=confmacro
1096 https://jira.lyrasis.org/browse/DS-2201?src=confmacro
1097 https://jira.lyrasis.org/browse/DS-2278?src=confmacro
1098 https://jira.lyrasis.org/browse/DS-2278?src=confmacro
1099 https://jira.lyrasis.org/browse/DS-2355?src=confmacro
1100 https://jira.lyrasis.org/browse/DS-2355?src=confmacro
1101 https://jira.lyrasis.org/browse/DS-2412?src=confmacro
1102 https://jira.lyrasis.org/browse/DS-2412?src=confmacro
1103 https://jira.lyrasis.org/browse/DS-2415?src=confmacro
1104 https://jira.lyrasis.org/browse/DS-2415?src=confmacro
1105 https://jira.lyrasis.org/browse/DS-2419?src=confmacro
1106 https://jira.lyrasis.org/browse/DS-2419?src=confmacro

DS-2425[1107]	Typos in xoai.xml prevent filters from being used[1108]		Jan 26, 2015	Jan 27, 2015	Ivan Masár	Christian Scheible	CLOSED	Fixed
DS-2427[1109]	DSpace API does not always filter results by the DB schema of current connection[1110]		Jan 27, 2015	May 08, 2015	Tim Donohue	Tim Donohue	CLOSED	Fixed
DS-2435[1111]	JSPUI should send HTTP 400 Bad Request if specified Browse index is not configured[1112]		Feb 03, 2015	Feb 03, 2015	Pascal-Nicolas Becker	Pascal-Nicolas Becker	CLOSED	Fixed
DS-2438[1113]	OAI indexing fails if a metadata field contains a value with more than 32766 Bytes[1114]		Feb 04, 2015	Feb 04, 2015	Ivan Masár	Christian Scheible	CLOSED	Fixed
DS-2445[1115]	XMLUI Directory Traversal Vulnerability[1116]		Feb 05, 2015	Mar 07, 2016	Tim Donohue	Tim Donohue	CLOSED	Fixed
DS-2448[1117]	JSPUI Path Traversal Vulnerability[1118]		Feb 09, 2015	Mar 25, 2015	Pascal-Nicolas Becker	Pascal-Nicolas Becker	CLOSED	Fixed

1107 https://jira.lyrasis.org/browse/DS-2425?src=confmacro
1108 https://jira.lyrasis.org/browse/DS-2425?src=confmacro
1109 https://jira.lyrasis.org/browse/DS-2427?src=confmacro
1110 https://jira.lyrasis.org/browse/DS-2427?src=confmacro
1111 https://jira.lyrasis.org/browse/DS-2435?src=confmacro
1112 https://jira.lyrasis.org/browse/DS-2435?src=confmacro
1113 https://jira.lyrasis.org/browse/DS-2438?src=confmacro
1114 https://jira.lyrasis.org/browse/DS-2438?src=confmacro
1115 https://jira.lyrasis.org/browse/DS-2445?src=confmacro
1116 https://jira.lyrasis.org/browse/DS-2445?src=confmacro
1117 https://jira.lyrasis.org/browse/DS-2448?src=confmacro
1118 https://jira.lyrasis.org/browse/DS-2448?src=confmacro

20 issues[1119]

7.6.3.10 Changes in DSpace 5.0

key	summary	assignee	reporter

Can't show details. Ask your admin to add this Jira URL to the allowlist.
View these issues in Jira[1120]

key	summary	assignee	reporter

Can't show details. Ask your admin to add this Jira URL to the allowlist.
View these issues in Jira[1121]

key	summary	assignee	reporter

Can't show details. Ask your admin to add this Jira URL to the allowlist.
View these issues in Jira[1122]

1119 https://jira.lyrasis.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project+%3D+DS+AND+issuetype+%3D+Bug+AND+resolution+%3D+Fixed+AND+fixVersion+%3D+%225.1%22+ORDER+BY+key+ASC+&src=confmacro
1120 https://jira.duraspace.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project%20=%20DS%20AND%20issuetype%20=%20%22New%20Feature%22%20AND%20resolution%20=%20Fixed%
1121 https://jira.duraspace.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project%20=%20DS%20AND%20issuetype%20in%20%28Task,%20%22Code%20Task%22,%20Documentation,%20Sub-task%29%20AND%20resolution%20=%20Fixed%20AND%20fixVersion%20=%20%225.0%22%20ORDER%20BY%20key%20ASC&tempMax=1
1122 https://jira.duraspace.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project%20=%20DS%20AND%20issuetype%20=%20Bug%20AND%20resolution%20=%20Fixed%20AND%20fixVersion

7.6.4 Changes in 4.x

7.6.4.1 Changes in DSpace 4.9

Key	Summary	T	Created	Updated	Due	Assignee	Reporter	P	Status	Resolution
DS-3840[1123]	CSV Metadata exports ignores metadata.hide.* and exposes sensitive metadata[1124]		Feb 16, 2018	Jun 27, 2018		Kim Shepherd	Eike Kleiner		CLOSED	Fixed
DS-3866[1125]	JSPUI EPerson name and Group name JS injection vulnerability[1126]		Mar 09, 2018	Jun 30, 2018		Kim Shepherd	Julio Brafman		CLOSED	Fixed

2 issues[1127]

1123 https://jira.lyrasis.org/browse/DS-3840?src=confmacro
1124 https://jira.lyrasis.org/browse/DS-3840?src=confmacro
1125 https://jira.lyrasis.org/browse/DS-3866?src=confmacro
1126 https://jira.lyrasis.org/browse/DS-3866?src=confmacro
1127 https://jira.lyrasis.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project+%3D+DS+AND+issuetype+%3D+Bug+AND+resolution+%3D+Fixed+AND+fixVersion+%3D+%224.9%22+ORDER+BY+key+ASC+++++++++&src=confmacro

7.6.4.2 Changes in DSpace 4.8

Key	Summary	T	Created	Updated	Due	Assignee	Reporter	P	Status	Resolution
DS-3415[1128]	administrative.js doEditCommunity wrong parameter name[1129]		Dec 07, 2016	Feb 21, 2017		Unassigned	Samuel Cambien (Atmire)		CLOSED	Fixed
DS-3431[1130]	BasicWorkflow system is vulnerable to unauthorized manipulations[1131]		Dec 28, 2016	Jul 12, 2017		Pascal-Nicolas Becker	Pascal-Nicolas Becker		CLOSED	Fixed
DS-3519[1132]	PasswordAuthentication getSpecialGroups empty exception catchblock[1133]		Mar 03, 2017	Mar 04, 2017		Unassigned	Jonas Van Goolen (Atmire)		CLOSED	Fixed
DS-3520[1134]	Apache Commons Collections vulnerability (COLLECTIONS-580)[1135]		Mar 06, 2017	Jul 13, 2017		Tim Donohue	Alan Orth		CLOSED	Fixed
DS-3584[1136]	when editing an eperson, trying to change its email address is		Apr 26, 2017	Jul 05, 2017		Unassigned	Samuel Cambien (Atmire)		CLOSED	Fixed

1128 https://jira.lyrasis.org/browse/DS-3415?src=confmacro
1129 https://jira.lyrasis.org/browse/DS-3415?src=confmacro
1130 https://jira.lyrasis.org/browse/DS-3431?src=confmacro
1131 https://jira.lyrasis.org/browse/DS-3431?src=confmacro
1132 https://jira.lyrasis.org/browse/DS-3519?src=confmacro
1133 https://jira.lyrasis.org/browse/DS-3519?src=confmacro
1134 https://jira.lyrasis.org/browse/DS-3520?src=confmacro
1135 https://jira.lyrasis.org/browse/DS-3520?src=confmacro
1136 https://jira.lyrasis.org/browse/DS-3584?src=confmacro

	ignored if another user already has that email address[1137]								
DS-3647[1138]	BasicWorkflow system is vulnerable to unauthorized manipulations (was: DS-3431)[1139]		Jul 12, 2017	Jul 14, 2017	Pascal-Nicolas Becker	Pascal-Nicolas Becker		CLOSED	Fixed

6 issues[1140]

7.6.4.3 Changes in DSpace 4.7

Key	Summary	T	Created	Updated	Due	Assignee	Reporter	P	Status	Resolution
DS-2895[1141]	Any registered user can modify inprogress submission[1142]		Nov 18, 2015	Oct 13, 2016		Andrea Bollini (4Science)	Andrea Bollini (4Science)		CLOSED	Fixed
DS-3097[1143]	Bitstreams of embargoed and/or withdrawn items can be accessed by anyone[1144]		Mar 10, 2016	Dec 08, 2016		Andrea Bollini (4Science)	Mark H. Wood		CLOSED	Fixed

2 issues[1145]

7.6.4.4 Changes in DSpace 4.6

Key	Summary	T	Created	Updated	Due	Assignee	Reporter	P	Status	Resolution

1137 https://jira.lyrasis.org/browse/DS-3584?src=confmacro
1138 https://jira.lyrasis.org/browse/DS-3647?src=confmacro
1139 https://jira.lyrasis.org/browse/DS-3647?src=confmacro
1140 https://jira.lyrasis.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project+%3D+DS+AND+issuetype+%3D+Bug+AND+resolution+%3D+Fixed+AND+fixVersion+%3D+%224.8%22+ORDER+BY+key+ASC++++++++&src=confmacro
1141 https://jira.lyrasis.org/browse/DS-2895?src=confmacro
1142 https://jira.lyrasis.org/browse/DS-2895?src=confmacro
1143 https://jira.lyrasis.org/browse/DS-3097?src=confmacro
1144 https://jira.lyrasis.org/browse/DS-3097?src=confmacro
1145 https://jira.lyrasis.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project+%3D+DS+AND+issuetype+%3D+Bug+AND+resolution+%3D+Fixed+AND+fixVersion+%3D+%224.7%22+ORDER+BY+key+ASC+++++++&src=confmacro

DS-2702[1146]	Cannot send email using SSL[1147]		Aug 13, 2015	Oct 13, 2016	Bram Luyten (Atmire)	Roeland Dillen	CLOSED	Fixed
DS-3309[1148]	XML External Entity (XXE) vulnerability in pdfbox[1149]		Sep 07, 2016	Oct 13, 2016	Unassigned	Seth Robbins	CLOSED	Fixed
DS-3328[1150]	ItemTest and CollectionTest fails in dspace-4_x branch[1151]		Sep 16, 2016	Sep 16, 2016	Pascal-Nicolas Becker	Pascal-Nicolas Becker	CLOSED	Fixed
DS-3330[1152]	Test fails into dspace-4_x[1153]		Sep 19, 2016	Sep 19, 2016	Luigi Andrea Pascarelli (4Science)	Luigi Andrea Pascarelli (4Science)	CLOSED	Fixed

4 issues[1154]

7.6.4.5 Changes in DSpace 4.5

1146 https://jira.lyrasis.org/browse/DS-2702?src=confmacro
1147 https://jira.lyrasis.org/browse/DS-2702?src=confmacro
1148 https://jira.lyrasis.org/browse/DS-3309?src=confmacro
1149 https://jira.lyrasis.org/browse/DS-3309?src=confmacro
1150 https://jira.lyrasis.org/browse/DS-3328?src=confmacro
1151 https://jira.lyrasis.org/browse/DS-3328?src=confmacro
1152 https://jira.lyrasis.org/browse/DS-3330?src=confmacro
1153 https://jira.lyrasis.org/browse/DS-3330?src=confmacro
1154 https://jira.lyrasis.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project+%3D+DS+AND+issuetype+%3D+Bug+AND+resolution+%3D+Fixed+AND+fixVersion+%3D+%224.6%22+ORDER+BY+key+ASC++++++&src=confmacro

Key	Summary	T	Created	Updated	Due	Assignee	Reporter	P	Status	Resolution
DS-3063[1155]	JSPUI Edit News feature can be used to view/edit other files readable to Tomcat user[1156]		Feb 15, 2016	May 16, 2016		Andrea Bollini (4Science)	Tim Donohue		CLOSED	Fixed
DS-3094[1157]	XMLUI Directory Traversal Vulnerability in Themes[1158]		Mar 07, 2016	May 16, 2016		Tim Donohue	Tim Donohue		CLOSED	Fixed

2 issues[1159]

7.6.4.6 Changes in DSpace 4.4

1155 https://jira.lyrasis.org/browse/DS-3063?src=confmacro
1156 https://jira.lyrasis.org/browse/DS-3063?src=confmacro
1157 https://jira.lyrasis.org/browse/DS-3094?src=confmacro
1158 https://jira.lyrasis.org/browse/DS-3094?src=confmacro
1159 https://jira.lyrasis.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project+%3D+DS+AND+issuetype+%3D+Bug+AND+resolution+%3D+Fixed+AND+fixVersion+%3D+%224.5%22+ORDER+BY+key+ASC+++++&src=confmacro

Key	Summary	T	Created	Updated	Due	Assignee	Reporter	P	Status	Resolution
DS-2736[1160]	XSS in JSPUI search form[1161]		Sep 02, 2015	May 16, 2016		Tim Donohue	Genaro Contreras		CLOSED	Fixed
DS-2737[1162]	Expression language Injection in JSPUI search form[1163]		Sep 02, 2015	May 16, 2016		Tim Donohue	Genaro Contreras		CLOSED	Fixed

2 issues[1164]

7.6.4.7 Changes in DSpace 4.3

Key	Summary	T	Created	Updated	Due	Assignee	Reporter	P	Status	Resolution
DS-1702[1165]	Cross-site scripting (XSS injection) is possible in JSPUI Recent Submissions listings[1166]		Oct 15, 2013	Mar 25, 2015		Luigi Andrea Pascarelli (4Science)	Sean Xiao		CLOSED	Fixed

1160 https://jira.lyrasis.org/browse/DS-2736?src=confmacro
1161 https://jira.lyrasis.org/browse/DS-2736?src=confmacro
1162 https://jira.lyrasis.org/browse/DS-2737?src=confmacro
1163 https://jira.lyrasis.org/browse/DS-2737?src=confmacro
1164 https://jira.lyrasis.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project+%3D+DS+AND+issuetype+%3D+Bug+AND+resolution+%3D+Fixed+AND+fixVersion+%3D+%224.4%22+ORDER+BY+key+ASC++++&src=confmacro
1165 https://jira.lyrasis.org/browse/DS-1702?src=confmacro
1166 https://jira.lyrasis.org/browse/DS-1702?src=confmacro

DS-1896[1167]	XMLUI returns 500 response for most invalid "/static" URLs[1168]	Jan 30, 2014	Feb 24, 2015	Tim Donohue	Tim Donohue		CLOSED	Fixed
DS-2044[1169]	Cross-site scripting (XSS injection) is possible in JSPUI Discovery search form[1170]	Jun 30, 2014	Mar 25, 2015	Luigi Andrea Pascarelli (4Science)	Gabriela Mircea		CLOSED	Fixed
DS-2130[1171]	XMLUI allows access to theme XSL files[1172]	Sep 03, 2014	Aug 01, 2016	Tim Donohue	Hardy Pottinger		CLOSED	Fixed
DS-2445[1173]	XMLUI Directory Traversal Vulnerability[1174]	Feb 05, 2015	Mar 07, 2016	Tim Donohue	Tim Donohue		CLOSED	Fixed
DS-2448[1175]	JSPUI Path Traversal Vulnerability[1176]	Feb 09, 2015	Mar 25, 2015	Pascal-Nicolas Becker	Pascal-Nicolas Becker		CLOSED	Fixed

6 issues[1177]

7.6.4.8 Changes in DSpace 4.2

Key	Summary	T	Created	Updated	Due	Assignee	Reporter	P	Status	Resolution
DS-913[1178]	Ukrainian translation for Manakin web interface[1179]		May 30, 2011	Jul 16, 2014		Ivan Masár	Parhomenko Yaroslav		CLOSED	Fixed

1167 https://jira.lyrasis.org/browse/DS-1896?src=confmacro
1168 https://jira.lyrasis.org/browse/DS-1896?src=confmacro
1169 https://jira.lyrasis.org/browse/DS-2044?src=confmacro
1170 https://jira.lyrasis.org/browse/DS-2044?src=confmacro
1171 https://jira.lyrasis.org/browse/DS-2130?src=confmacro
1172 https://jira.lyrasis.org/browse/DS-2130?src=confmacro
1173 https://jira.lyrasis.org/browse/DS-2445?src=confmacro
1174 https://jira.lyrasis.org/browse/DS-2445?src=confmacro
1175 https://jira.lyrasis.org/browse/DS-2448?src=confmacro
1176 https://jira.lyrasis.org/browse/DS-2448?src=confmacro
1177 https://jira.lyrasis.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project+%3D+DS+AND+issuetype+%3D+Bug+AND+resolution+%3D+Fixed+AND+fixVersion+%3D+%224.3%22+ORDER+BY+key+ASC+++&src=confmacro
1178 https://jira.lyrasis.org/browse/DS-913?src=confmacro
1179 https://jira.lyrasis.org/browse/DS-913?src=confmacro

DS-1906[1180]	Shibboleth attributes may need to be reconverted[1181]		Feb 07, 2014	Aug 01, 2016	Hardy Pottinger	Pascal-Nicolas Becker		CLOSED	Fixed
DS-1932[1182]	Complete Update translation of JSPUI pt_BR[1183]		Mar 01, 2014	Mar 06, 2014	Ivan Masár	Washington Ribeiro		CLOSED	Fixed
DS-1943[1184]	Language Selection _ Turkish Option[1185]		Mar 11, 2014	May 29, 2014	Mark H. Wood	Sonmez CELIK		CLOSED	Fixed
DS-2047[1186]	zh_TW language for dspace 4.1 jspui[1187]		Jul 02, 2014	Jul 16, 2014	Ivan Masár	Chunmin Tai		CLOSED	Fixed

5 issues[1188]

Key	Summary	T	Created	Updated	Due	Assignee	Reporter	P	Status	Resolution
DS-1411[1189]	uncaught NPE in stats-log-converter -m[1190]		Dec 03, 2012	Mar 26, 2014		Mark H. Wood	Ivan Masár		CLOSED	Fixed
DS-1584[1191]	XMLUI "Browse by" sorting Bug[1192]		Jun 20, 2013	Apr 23, 2014		Bram Luyten (Atmire)	Denis Fdz		CLOSED	Fixed
DS-1919[1193]	Solr Search Empty FilterQuery bug[1194]		Feb 20, 2014	Aug 01, 2016		Hardy Pottinger	Denis Fdz		CLOSED	Fixed

1180 https://jira.lyrasis.org/browse/DS-1906?src=confmacro
1181 https://jira.lyrasis.org/browse/DS-1906?src=confmacro
1182 https://jira.lyrasis.org/browse/DS-1932?src=confmacro
1183 https://jira.lyrasis.org/browse/DS-1932?src=confmacro
1184 https://jira.lyrasis.org/browse/DS-1943?src=confmacro
1185 https://jira.lyrasis.org/browse/DS-1943?src=confmacro
1186 https://jira.lyrasis.org/browse/DS-2047?src=confmacro
1187 https://jira.lyrasis.org/browse/DS-2047?src=confmacro
1188 https://jira.lyrasis.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project+%3D+DS+AND+issuetype+in+%28Task%2C+Improvement%2C+%22Code+Task%22%2C+Documentation%2C+Sub-task%29+AND+resolution+%3D+Fixed+AND+fixVersion+%3D+%224.2%22+ORDER+BY+key+ASC++&src=confmacro
1189 https://jira.lyrasis.org/browse/DS-1411?src=confmacro
1190 https://jira.lyrasis.org/browse/DS-1411?src=confmacro
1191 https://jira.lyrasis.org/browse/DS-1584?src=confmacro
1192 https://jira.lyrasis.org/browse/DS-1584?src=confmacro
1193 https://jira.lyrasis.org/browse/DS-1919?src=confmacro
1194 https://jira.lyrasis.org/browse/DS-1919?src=confmacro

DS-1928[1195]	OAI-PMH Identify response well-formed but invalid[1196]		Feb 27, 2014	Jul 17, 2014	Ivan Masár	Ondřej Košarko		Fixed
DS-1940[1197]	DS-1867 caused error running "mvn package"[1198]		Mar 08, 2014	Jul 28, 2014	Tim Donohue	Mohsen		Fixed
DS-1944[1199]	http://mobile.demo.dspace.org/xmlui/[1200]		Mar 11, 2014	Aug 01, 2016	Hardy Pottinger	Thomas Misilo		Fixed
DS-1946[1201]	Solr floods catalina.out with unwanted messages[1202]		Mar 12, 2014	Apr 09, 2014	Mark H. Wood	Mark H. Wood		Fixed
DS-1947[1203]	Missing m-tweaks.js for mobile theme[1204]		Mar 13, 2014	Jul 16, 2014	Tim Donohue	Thomas Misilo		Fixed
DS-1957[1205]	incorrect xml workflow script for oracle[1206]		Apr 01, 2014	Jul 09, 2014	Mark H. Wood	Roeland Dillen		Fixed
DS-1958[1207]	Discovery OutOfMemoryError when indexing Large Bitstreams[1208]		Apr 02, 2014	Jul 17, 2014	Mark H. Wood	Mark Diggory		Fixed

1195 https://jira.lyrasis.org/browse/DS-1928?src=confmacro
1196 https://jira.lyrasis.org/browse/DS-1928?src=confmacro
1197 https://jira.lyrasis.org/browse/DS-1940?src=confmacro
1198 https://jira.lyrasis.org/browse/DS-1940?src=confmacro
1199 https://jira.lyrasis.org/browse/DS-1944?src=confmacro
1200 https://jira.lyrasis.org/browse/DS-1944?src=confmacro
1201 https://jira.lyrasis.org/browse/DS-1946?src=confmacro
1202 https://jira.lyrasis.org/browse/DS-1946?src=confmacro
1203 https://jira.lyrasis.org/browse/DS-1947?src=confmacro
1204 https://jira.lyrasis.org/browse/DS-1947?src=confmacro
1205 https://jira.lyrasis.org/browse/DS-1957?src=confmacro
1206 https://jira.lyrasis.org/browse/DS-1957?src=confmacro
1207 https://jira.lyrasis.org/browse/DS-1958?src=confmacro
1208 https://jira.lyrasis.org/browse/DS-1958?src=confmacro

DS-1961[1209]	Use HTTPS with oss.sonatype.org repository[1210]		Apr 04, 2014	Jul 28, 2014	Mark H. Wood	Mark H. Wood	CLOSED	Fixed
DS-1970[1211]	to many open files exception when update lucene index[1212]		Apr 15, 2014	Aug 01, 2016	Hardy Pottinger	Roeland Dillen	CLOSED	Fixed
DS-1971[1213]	'bte-io' (v 0.9.2.3) dependency from EKT has an invalid SNAPSHOT dependency in its POM[1214]		Apr 15, 2014	Aug 01, 2016	Hardy Pottinger	Tim Donohue	CLOSED	Fixed
DS-1986[1215]	REST API holds on to context for too long, should use DB pool[1216]		Apr 27, 2014	Jul 16, 2014	Peter Dietz	Peter Dietz	CLOSED	Fixed
DS-1998[1217]	"dspace classpath" CLI command does nothing, throws error[1218]		May 11, 2014	Jun 05, 2014	Mark H. Wood	Ivan Masár	CLOSED	Fixed
DS-2004[1219]	Catalan translation of Discovery strings[1220]		May 15, 2014	May 16, 2014	Ivan Masár	Àlex Magaz Graça	CLOSED	Fixed

1209 https://jira.lyrasis.org/browse/DS-1961?src=confmacro
1210 https://jira.lyrasis.org/browse/DS-1961?src=confmacro
1211 https://jira.lyrasis.org/browse/DS-1970?src=confmacro
1212 https://jira.lyrasis.org/browse/DS-1970?src=confmacro
1213 https://jira.lyrasis.org/browse/DS-1971?src=confmacro
1214 https://jira.lyrasis.org/browse/DS-1971?src=confmacro
1215 https://jira.lyrasis.org/browse/DS-1986?src=confmacro
1216 https://jira.lyrasis.org/browse/DS-1986?src=confmacro
1217 https://jira.lyrasis.org/browse/DS-1998?src=confmacro
1218 https://jira.lyrasis.org/browse/DS-1998?src=confmacro
1219 https://jira.lyrasis.org/browse/DS-2004?src=confmacro
1220 https://jira.lyrasis.org/browse/DS-2004?src=confmacro

DS-2013[1221]	JSPUI with Oracle DB - Browse items with THUMBNAILS unimplemented[1222]		May 22, 2014	Jun 05, 2014	Mark H. Wood	Denis Fdz		CLOSED	Fixed
DS-2035[1223]	dim crosswalk has missing values[1224]		Jun 24, 2014	Jul 16, 2014	Tim Donohue	Antoine Snyers (Atmire)		CLOSED	Fixed
DS-2036[1225]	DSpace upgrade with oracle database, no discovery results[1226]		Jun 24, 2014	May 08, 2015	Kevin Van de Velde (Atmire)	Kevin Van de Velde (Atmire)		CLOSED	Fixed
DS-2038[1227]	Oracle dspace-schema_3-4.sql upgrade script contains a minor error[1228]		Jun 24, 2014	Aug 01, 2016	Unassigned	Hardy Pottinger		CLOSED	Fixed

Showing 20 out of 22 issues[1229]

7.6.4.9 Changes in DSpace 4.1

Key	Summary	T	Created	Updated	Due	Assignee	Reporter	P	Status	Resolution
DS-1860[1230]	Community-list doesn't show all collections[1231]		Jan 13, 2014	Jul 28, 2014		Ivan Masár	Denis Fdz		CLOSED	Fixed
DS-1866[1232]	"Consuming Web Services" curation task is missing documentation[1233]		Jan 15, 2014	Jan 21, 2014		Richard Rodgers	Tim Donohue		CLOSED	Fixed

1221 https://jira.lyrasis.org/browse/DS-2013?src=confmacro
1222 https://jira.lyrasis.org/browse/DS-2013?src=confmacro
1223 https://jira.lyrasis.org/browse/DS-2035?src=confmacro
1224 https://jira.lyrasis.org/browse/DS-2035?src=confmacro
1225 https://jira.lyrasis.org/browse/DS-2036?src=confmacro
1226 https://jira.lyrasis.org/browse/DS-2036?src=confmacro
1227 https://jira.lyrasis.org/browse/DS-2038?src=confmacro
1228 https://jira.lyrasis.org/browse/DS-2038?src=confmacro
1229 https://jira.lyrasis.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project+%3D+DS+AND+issuetype+%3D+Bug+AND+resolution+%3D+Fixed+AND+fixVersion+%3D+%224.2%22+ORDER+BY+key+ASC++&src=confmacro
1230 https://jira.lyrasis.org/browse/DS-1860?src=confmacro
1231 https://jira.lyrasis.org/browse/DS-1860?src=confmacro
1232 https://jira.lyrasis.org/browse/DS-1866?src=confmacro
1233 https://jira.lyrasis.org/browse/DS-1866?src=confmacro

DS-1911[1234]	Update translation of JSPUI pt_BR[1235]	Feb 14, 2014	Feb 14, 2014	Ivan Masár	Tiago Murakami		CLOSED	Fixed

3 issues[1236]

Key	Summary	T	Created	Updated	Due	Assignee	Reporter	P	Status	Resolution
DS-1445[1237]	XOAI validation issues - ListRecords response gave a noRecordsMatch[1238]		Jan 09, 2013	Jul 24, 2014		João Melo	Christos Rodosthenous		CLOSED	Fixed
DS-1531[1239]	bug in current DSpace (3.1) with log importing[1240]		Apr 06, 2013	Feb 05, 2014		Mark H. Wood	James Halliday		CLOSED	Fixed
DS-1536[1241]	having a DOT in handle prefix causes identifier.uri to be cut off when being created[1242]		Apr 17, 2013	Sep 22, 2014		Ivan Masár	Jose Blanco		CLOSED	Fixed
DS-1744[1243]	Upgrade to latest log4j[1244]		Oct 30, 2013	Dec 20, 2013		Mark H. Wood	Mark H. Wood		CLOSED	Fixed
DS-1756[1245]	Wrongly aligned text on item view in mobile theme[1246]		Nov 05, 2013	Jan 29, 2014		Ivan Masár	Marina Muilwijk		CLOSED	Fixed

1234 https://jira.lyrasis.org/browse/DS-1911?src=confmacro
1235 https://jira.lyrasis.org/browse/DS-1911?src=confmacro
1236 https://jira.lyrasis.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project+%3D+DS+AND+issuetype+in+%28Task%2C+Improvement%2C+%22Code+Task%22%2C+Documentation%2C+Sub-task%29+AND+resolution+%3D+Fixed+AND+fixVersion+%3D+%224.1%22+ORDER+BY+key+ASC++&src=confmacro
1237 https://jira.lyrasis.org/browse/DS-1445?src=confmacro
1238 https://jira.lyrasis.org/browse/DS-1445?src=confmacro
1239 https://jira.lyrasis.org/browse/DS-1531?src=confmacro
1240 https://jira.lyrasis.org/browse/DS-1531?src=confmacro
1241 https://jira.lyrasis.org/browse/DS-1536?src=confmacro
1242 https://jira.lyrasis.org/browse/DS-1536?src=confmacro
1243 https://jira.lyrasis.org/browse/DS-1744?src=confmacro
1244 https://jira.lyrasis.org/browse/DS-1744?src=confmacro
1245 https://jira.lyrasis.org/browse/DS-1756?src=confmacro
1246 https://jira.lyrasis.org/browse/DS-1756?src=confmacro

DS-1757[1247]	Missing images in mobile theme[1248]		Nov 05, 2013	Jan 22, 2014	Ivan Masár	Marina Muilwijk	CLOSED	Fixed
DS-1779[1249]	Pagination link error in JSPUI discovery search[1250]		Nov 11, 2013	Feb 20, 2014	Kim Shepherd	Raul Ruiz	CLOSED	Fixed
DS-1795[1251]	When run command dspace "dspace stat-initial"[1252]		Nov 17, 2013	Feb 26, 2014	Mark H. Wood	Anonymous (No Reply)	CLOSED	Fixed
DS-1816[1253]	Incorrect label for search in community in navigation section.[1254]		Dec 02, 2013	Jan 27, 2014	Ivan Masár	Bavo Van Geit	CLOSED	Fixed
DS-1832[1255]	Proxy configuration set in system properties if empty[1256]		Dec 11, 2013	Dec 21, 2013	Kevin Van de Velde (Atmire)	Kevin Van de Velde (Atmire)	CLOSED	Fixed
DS-1833[1257]	Collection content source harvesting test does not work with ORE[1258]		Dec 11, 2013	Jan 30, 2014	Kevin Van de Velde (Atmire)	Kevin Van de Velde (Atmire)	CLOSED	Fixed
DS-1834[1259]	Collection content source harvesting test does not check sets properly[1260]		Dec 11, 2013	Feb 17, 2014	Kevin Van de Velde (Atmire)	Kevin Van de Velde (Atmire)	CLOSED	Fixed
DS-1835[1261]	Invalid bootstrap CSS[1262]		Dec 11, 2013	Jan 29, 2014	Mark H. Wood	Mark H. Wood	CLOSED	Fixed

1247 https://jira.lyrasis.org/browse/DS-1757?src=confmacro
1248 https://jira.lyrasis.org/browse/DS-1757?src=confmacro
1249 https://jira.lyrasis.org/browse/DS-1779?src=confmacro
1250 https://jira.lyrasis.org/browse/DS-1779?src=confmacro
1251 https://jira.lyrasis.org/browse/DS-1795?src=confmacro
1252 https://jira.lyrasis.org/browse/DS-1795?src=confmacro
1253 https://jira.lyrasis.org/browse/DS-1816?src=confmacro
1254 https://jira.lyrasis.org/browse/DS-1816?src=confmacro
1255 https://jira.lyrasis.org/browse/DS-1832?src=confmacro
1256 https://jira.lyrasis.org/browse/DS-1832?src=confmacro
1257 https://jira.lyrasis.org/browse/DS-1833?src=confmacro
1258 https://jira.lyrasis.org/browse/DS-1833?src=confmacro
1259 https://jira.lyrasis.org/browse/DS-1834?src=confmacro
1260 https://jira.lyrasis.org/browse/DS-1834?src=confmacro
1261 https://jira.lyrasis.org/browse/DS-1835?src=confmacro
1262 https://jira.lyrasis.org/browse/DS-1835?src=confmacro

DS-1846[1263]	Cannot deposit new item via SWORD[1264]		Dec 18, 2013	Jul 28, 2014	Andrea Schweer	Àlex Magaz Graça		CLOSED	Fixed
DS-1848[1265]	OAI harvest issues when starting from control panel/command line[1266]		Dec 20, 2013	Feb 18, 2014	Kevin Van de Velde (Atmire)	Kevin Van de Velde (Atmire)		CLOSED	Fixed
DS-1857[1267]	Unhandled exception in BTE batch import when uploading CSV files with misconfiguration options[1268]		Jan 08, 2014	Jan 31, 2014	Kostas Stamatis	Kostas Stamatis		CLOSED	Fixed
DS-1863[1269]	JSPUI eperson and group selection should use the new theme[1270]		Jan 15, 2014	Feb 20, 2014	Ivan Masár	Denis Fdz		CLOSED	Fixed
DS-1867[1271]	Maven build issues from [src]/dspace/ , error finding target/build.properties[1272]		Jan 15, 2014	Apr 09, 2015	Tim Donohue	Tim Donohue		CLOSED	Fixed
DS-1873[1273]	CheckSum Checker Emailer sends emails for 0 issues, doesn't specify any site info[1274]		Jan 21, 2014	Apr 09, 2015	Tim Donohue	Tim Donohue		CLOSED	Fixed
DS-1878[1275]	XMLUI mobile theme front page search doesn't recognize Discovery[1276]		Jan 23, 2014	Jan 23, 2014	Ivan Masár	Ivan Masár		CLOSED	Fixed

Showing 20 out of 31 issues[1277]

1263 https://jira.lyrasis.org/browse/DS-1846?src=confmacro
1264 https://jira.lyrasis.org/browse/DS-1846?src=confmacro
1265 https://jira.lyrasis.org/browse/DS-1848?src=confmacro
1266 https://jira.lyrasis.org/browse/DS-1848?src=confmacro
1267 https://jira.lyrasis.org/browse/DS-1857?src=confmacro
1268 https://jira.lyrasis.org/browse/DS-1857?src=confmacro
1269 https://jira.lyrasis.org/browse/DS-1863?src=confmacro
1270 https://jira.lyrasis.org/browse/DS-1863?src=confmacro
1271 https://jira.lyrasis.org/browse/DS-1867?src=confmacro
1272 https://jira.lyrasis.org/browse/DS-1867?src=confmacro
1273 https://jira.lyrasis.org/browse/DS-1873?src=confmacro
1274 https://jira.lyrasis.org/browse/DS-1873?src=confmacro
1275 https://jira.lyrasis.org/browse/DS-1878?src=confmacro
1276 https://jira.lyrasis.org/browse/DS-1878?src=confmacro
1277 https://jira.lyrasis.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project+%3D+DS+AND+issuetype+%3D+Bug+AND+resolution+%3D+Fixed+AND+fixVersion+%3D+%224.1%22+ORDER+BY+key+ASC+&src=confmacro

7.6.4.10 Changes in DSpace 4.0

key	summary	assignee	reporter

Can't show details. Ask your admin to add this Jira URL to the allowlist.
View these issues in Jira[1278]

key	summary	assignee	reporter

Can't show details. Ask your admin to add this Jira URL to the allowlist.
View these issues in Jira[1279]

key	summary	assignee	reporter

Can't show details. Ask your admin to add this Jira URL to the allowlist.
View these issues in Jira[1280]

7.6.5 Changes in 3.x

1278https://jira.duraspace.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project%20=%20DS%20AND%20issuetype%20=%20%22New%20Feature%22%20AND%20resolution%20=%20Fixed%

1279https://jira.duraspace.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project%20=%20DS%20AND%20issuetype%20in%20%28Task,%20%22Code%20Task%22,%20Documentation,%20Sub-task%29%20AND%20resolution%20=%20Fixed%20AND%20fixVersion%20=%20%224.0%22%20ORDER%20BY%20key%20ASC&tempMax=1

1280https://jira.duraspace.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project%20=%20DS%20AND%20issuetype%20=%20Bug%20AND%20resolution%20=%20Fixed%20AND%20fixVersion

7.6.5.1 Changes in DSpace 3.6

Key	Summary	T	Created	Updated	Due	Assignee	Reporter	P	Status	Resolution
DS-3094[1281]	XMLUI Directory Traversal Vulnerability in Themes[1282]		Mar 07, 2016	May 16, 2016		Tim Donohue	Tim Donohue		CLOSED	Fixed

1 issue[1283]

7.6.5.2 Changes in DSpace 3.5

Key	Summary	T	Created	Updated	Due	Assignee	Reporter	P	Status	Resolution
DS-2736[1284]	XSS in JSPUI search form[1285]		Sep 02, 2015	May 16, 2016		Tim Donohue	Genaro Contreras		CLOSED	Fixed
DS-2737[1286]	Expression language Injection in JSPUI search form[1287]		Sep 02, 2015	May 16, 2016		Tim Donohue	Genaro Contreras		CLOSED	Fixed

2 issues[1288]

7.6.5.3 Changes in DSpace 3.4

This release includes the following security fixes. All of these tickets require a valid JIRA account to view the details:

1281 https://jira.lyrasis.org/browse/DS-3094?src=confmacro
1282 https://jira.lyrasis.org/browse/DS-3094?src=confmacro
1283 https://jira.lyrasis.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project+%3D+DS+AND+issuetype+%3D+Bug+AND+resolution+%3D+Fixed+AND+fixVersion+%3D+%223.6%22+ORDER+BY+key+ASC+++&src=confmacro
1284 https://jira.lyrasis.org/browse/DS-2736?src=confmacro
1285 https://jira.lyrasis.org/browse/DS-2736?src=confmacro
1286 https://jira.lyrasis.org/browse/DS-2737?src=confmacro
1287 https://jira.lyrasis.org/browse/DS-2737?src=confmacro
1288 https://jira.lyrasis.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project+%3D+DS+AND+issuetype+%3D+Bug+AND+resolution+%3D+Fixed+AND+fixVersion+%3D+%223.5%22+ORDER+BY+key+ASC++&src=confmacro

- DS-1702[1289] - Cross-site scripting (XSS injection) is possible in JSPUI Recent Submissions listings
- DS-2044[1290] - Cross-site scripting (XSS injection) is possible in JSPUI Discovery search form
- DS-2445[1291] - XMLUI Directory Traversal Vulnerabilities
    - Also resolves related, minor theme access issues DS-1896[1292] and DS-2130[1293].
- DS-2448[1294] - JSPUI Directory Traversal Vulnerability

7.6.5.4 Changes in DSpace 3.3

Key	Summary	T	Created	Updated	Due	Assignee	Reporter	P	Status	Resolution
DS-1536[1295]	having a DOT in handle prefix causes identifier.uri to be cut off when being created[1296]		Apr 17, 2013	Sep 22, 2014		Ivan Masár	Jose Blanco		CLOSED	Fixed
DS-1619[1297]	Unable to remove items after enabling SOLRBrowseDAOs[1298]		Aug 10, 2013	Jan 16, 2015		Andrea Bollini (4Science)	Andrea Bollini (4Science)		CLOSED	Fixed
DS-1834[1299]	Collection content source harvesting test does not check sets properly[1300]		Dec 11, 2013	Feb 17, 2014		Kevin Van de Velde (Atmire)	Kevin Van de Velde (Atmire)		CLOSED	Fixed
DS-1893[1301]	Get page refresh after adding a value in the submission forms clears all metadata in XMLUI[1302]		Jan 30, 2014	Jan 30, 2014		Kevin Van de Velde (Atmire)	Kevin Van de Velde (Atmire)		CLOSED	Fixed
DS-1898[1303]	OAI not always closing contexts[1304]		Jan 31, 2014	Jul 28, 2014		Kevin Van de Velde (Atmire)	Kevin Van de Velde (Atmire)		CLOSED	Fixed

1289 https://jira.duraspace.org/browse/DS-1702
1290 https://jira.duraspace.org/browse/DS-2044
1291 https://jira.duraspace.org/browse/DS-2445
1292 https://jira.duraspace.org/browse/DS-1896
1293 https://jira.duraspace.org/browse/DS-2130
1294 https://jira.duraspace.org/browse/DS-2448
1295 https://jira.lyrasis.org/browse/DS-1536?src=confmacro
1296 https://jira.lyrasis.org/browse/DS-1536?src=confmacro
1297 https://jira.lyrasis.org/browse/DS-1619?src=confmacro
1298 https://jira.lyrasis.org/browse/DS-1619?src=confmacro
1299 https://jira.lyrasis.org/browse/DS-1834?src=confmacro
1300 https://jira.lyrasis.org/browse/DS-1834?src=confmacro
1301 https://jira.lyrasis.org/browse/DS-1893?src=confmacro
1302 https://jira.lyrasis.org/browse/DS-1893?src=confmacro
1303 https://jira.lyrasis.org/browse/DS-1898?src=confmacro
1304 https://jira.lyrasis.org/browse/DS-1898?src=confmacro

DS-1958[1305]	Discovery OutOfMemoryError when indexing Large Bitstreams[1306]	Apr 02, 2014	Jul 17, 2014	Mark H. Wood	Mark Diggory		CLOSED	Fixed
DS-1961[1307]	Use HTTPS with oss.sonatype.org repository[1308]	Apr 04, 2014	Jul 28, 2014	Mark H. Wood	Mark H. Wood		CLOSED	Fixed
DS-1998[1309]	"dspace classpath" CLI command does nothing, throws error[1310]	May 11, 2014	Jun 05, 2014	Mark H. Wood	Ivan Masár		CLOSED	Fixed
DS-2013[1311]	JSPUI with Oracle DB - Browse items with THUMBNAILS unimplemented[1312]	May 22, 2014	Jun 05, 2014	Mark H. Wood	Denis Fdz		CLOSED	Fixed

9 issues[1313]

7.6.5.5 Changes in DSpace 3.2

key	summary	assignee	reporter

Can't show details. Ask your admin to add this Jira URL to the allowlist.
View these issues in Jira[1314]

key	summary	assignee	reporter

1305 https://jira.lyrasis.org/browse/DS-1958?src=confmacro
1306 https://jira.lyrasis.org/browse/DS-1958?src=confmacro
1307 https://jira.lyrasis.org/browse/DS-1961?src=confmacro
1308 https://jira.lyrasis.org/browse/DS-1961?src=confmacro
1309 https://jira.lyrasis.org/browse/DS-1998?src=confmacro
1310 https://jira.lyrasis.org/browse/DS-1998?src=confmacro
1311 https://jira.lyrasis.org/browse/DS-2013?src=confmacro
1312 https://jira.lyrasis.org/browse/DS-2013?src=confmacro
13 https://jira.lyrasis.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project+%3D+DS+AND+issuetype+%3D+Bug+AND+resolution+%3D+Fixed+AND+fixVersion+%3D+%223.3%22+ORDER+BY+key+ASC+&src=confmacro
1314 https://jira.duraspace.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project%20=%20DS%20AND%20issuetype%20in%20%28Task,%20%22Code%20Task%22,%20Documentation,%20Sub-task%29%20AND%20resolution%20=%20Fixed%20AND%20fixVersion%20=%20%223.2%22%20ORDER%20BY%20key%20ASC&tempMax=1

Can't show details. Ask your admin to add this Jira URL to the allowlist.
View these issues in Jira[1315]

This release also includes the following security fix:

- DS-1603[1316] - Resolves a security issue in JSPUI

7.6.5.6 Changes in DSpace 3.1

key	summary	assignee	reporter

Can't show details. Ask your admin to add this Jira URL to the allowlist.
View these issues in Jira[1317]

key	summary	assignee	reporter

Can't show details. Ask your admin to add this Jira URL to the allowlist.
View these issues in Jira[1318]

7.6.5.7 Changes in DSpace 3.0

key	summary	assignee	reporter

1315 https://jira.duraspace.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project%20=%20DS%20AND%20issuetype%20=%20Bug%20AND%20resolution%20=%20Fixed%20AND%20fixVersion

1316 https://jira.duraspace.org/browse/DS-1603

1317 https://jira.duraspace.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project%20=%20DS%20AND%20issuetype%20in%20%28Task,%20%22Code%20Task%22,%20Documentation,%20Sub-task%29%20AND%20resolution%20=%20Fixed%20AND%20fixVersion%20=%20%223.1%22%20ORDER%20BY%20key%20ASC&tempMax=1

1318 https://jira.duraspace.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project%20=%20DS%20AND%20issuetype%20=%20Bug%20AND%20resolution%20=%20Fixed%20AND%20fixVersion

Can't show details. Ask your admin to add this Jira URL to the allowlist.
View these issues in Jira[1319]

key	summary	assignee	reporter

Can't show details. Ask your admin to add this Jira URL to the allowlist.
View these issues in Jira[1320]

key	summary	assignee	reporter

Can't show details. Ask your admin to add this Jira URL to the allowlist.
View these issues in Jira[1321]

7.6.6 Changes in 1.8.x

- Changes in DSpace 1.8.3 (see page 760)
- Changes in DSpace 1.8.2 (see page 761)
- Changes in DSpace 1.8.1 (see page 761)
- Changes in DSpace 1.8.0 (see page 762)

7.6.6.1 Changes in DSpace 1.8.3

key	summary	assignee	reporter

1319 https://jira.duraspace.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project%20=%20DS%20AND%20issuetype%20=%20%22New%20Feature%22%20AND%20resolution%20=%20Fixed%

1320 https://jira.duraspace.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project%20=%20DS%20AND%20issuetype%20in%20%28Task,%20%22Code%20Task%22,%20Documentation,%20Sub-task%29%20AND%20resolution%20=%20Fixed%20AND%20fixVersion%20=%20%223.0%22%20ORDER%20BY%20key%20ASC&tempMax=1

1321 https://jira.duraspace.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project%20=%20DS%20AND%20issuetype%20=%20Bug%20AND%20resolution%20=%20Fixed%20AND%20fixVersion

Can't show details. Ask your admin to add this Jira URL to the allowlist.
View these issues in Jira[1322]

Bug Fixes in 1.8.3

- DS-1603[1323] - Resolves a security issue in JSPUI

7.6.6.2 Changes in DSpace 1.8.2

key	summary	assignee	reporter

Can't show details. Ask your admin to add this Jira URL to the allowlist.
View these issues in Jira[1324]

key	summary	assignee	reporter

Can't show details. Ask your admin to add this Jira URL to the allowlist.
View these issues in Jira[1325]

7.6.6.3 Changes in DSpace 1.8.1

key	summary	assignee	reporter

1322 https://jira.duraspace.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project%20=%20DS%20AND%20issuetype%20in%20%28Task,%20%22Code%20Task%22,%20Documentation,%20Sub-task%29%20AND%20resolution%20=%20Fixed%20AND%20fixVersion%20=%20%221.8.3%22%20ORDER%20BY%20key%20ASC&tempMax=

1323 https://jira.duraspace.org/browse/DS-1603

1324 https://jira.duraspace.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project%20=%20DS%20AND%20issuetype%20in%20%28Task,%20%22Code%20Task%22,%20Documentation,%20Sub-task%29%20AND%20resolution%20=%20Fixed%20AND%20fixVersion%20=%20%221.8.2%22%20ORDER%20BY%20key%20ASC&tempMax=

1325 https://jira.duraspace.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project%20=%20DS%20AND%20issuetype%20=%20Bug%20AND%20resolution%20=%20Fixed%20AND%20fixVersion

Can't show details. Ask your admin to add this Jira URL to the allowlist.
View these issues in Jira[1326]

key	summary	assignee	reporter

Can't show details. Ask your admin to add this Jira URL to the allowlist.
View these issues in Jira[1327]

7.6.6.4 Changes in DSpace 1.8.0

key	summary	assignee	reporter

Can't show details. Ask your admin to add this Jira URL to the allowlist.
View these issues in Jira[1328]

key	summary	assignee	reporter

Can't show details. Ask your admin to add this Jira URL to the allowlist.
View these issues in Jira[1329]

key	summary	assignee	reporter

1326 https://jira.duraspace.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project%20=%20DS%20AND%20issuetype%20in%20%28Task,%20%22Code%20Task%22,%20Documentation,%20Sub-task%29%20AND%20resolution%20=%20Fixed%20AND%20fixVersion%20=%20%221.8.1%22%20ORDER%20BY%20key%20ASC&tempMax=

1327 https://jira.duraspace.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project%20=%20DS%20AND%20issuetype%20=%20Bug%20AND%20resolution%20=%20Fixed%20AND%20fixVersion

1328 https://jira.duraspace.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project%20=%20DS%20AND%20issuetype%20=%20%22New%20Feature%22%20AND%20resolution%20=%20Fixed%

1329 https://jira.duraspace.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project%20=%20DS%20AND%20issuetype%20in%20%28Task,%20%22Code%20Task%22,%20Documentation,%20Sub-task%29%20AND%20resolution%20=%20Fixed%20AND%20fixVersion%20=%20%221.8.0%22%20ORDER%20BY%20key%20ASC&tempMax=

Can't show details. Ask your admin to add this Jira URL to the allowlist.
View these issues in Jira[1330]

7.6.7 Changes in 1.7.x

7.6.7.1 Changes in DSpace 1.7.3

key	summary	assignee	reporter

Can't show details. Ask your admin to add this Jira URL to the allowlist.
View these issues in Jira[1331]

Bug Fixes

- DS-1603[1332] - Resolves a security issue in JSPUI

7.6.7.2 Changes in DSpace 1.7.2

key	summary	assignee	reporter

Can't show details. Ask your admin to add this Jira URL to the allowlist.
View these issues in Jira[1333]

1330 https://jira.duraspace.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project%20=%20DS%20AND%20issuetype%20=%20Bug%20AND%20resolution%20=%20Fixed%20AND%20fixVersion

1331 https://jira.duraspace.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project%20=%20DS%20AND%20issuetype%20in%20%28Task,%20%22Code%20Task%22,%20Documentation,%20Sub-task%29%20AND%20resolution%20=%20Fixed%20AND%20fixVersion%20=%20%221.7.3%22%20ORDER%20BY%20key%20ASC&tempMax=

1332 https://jira.duraspace.org/browse/DS-1603

1333 https://jira.duraspace.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project%20=%20DS%20AND%20issuetype%20=%20Bug%20AND%20resolution%20=%20Fixed%20AND%20fixVersion

7.6.7.3 Changes in DSpace 1.7.1

key	summary	assignee	reporter

Can't show details. Ask your admin to add this Jira URL to the allowlist.
View these issues in Jira[1334]

key	summary	assignee	reporter

Can't show details. Ask your admin to add this Jira URL to the allowlist.
View these issues in Jira[1335]

7.6.7.4 Changes in DSpace 1.7.0

key	summary	assignee	reporter

Can't show details. Ask your admin to add this Jira URL to the allowlist.
View these issues in Jira[1336]

key	summary	assignee	reporter

Can't show details. Ask your admin to add this Jira URL to the allowlist.
View these issues in Jira[1337]

1334 https://jira.duraspace.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project%20=%20DS%20AND%20issuetype%20in%20%28Task,%20%22Code%20Task%22,%20Documentation,%20Sub-task%29%20AND%20resolution%20=%20Fixed%20AND%20fixVersion%20=%20%221.7.1%22%20ORDER%20BY%20key%20ASC&tempMax=

1335 https://jira.duraspace.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project%20=%20DS%20AND%20issuetype%20=%20Bug%20AND%20resolution%20=%20Fixed%20AND%20fixVersion

1336 https://jira.duraspace.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project%20=%20DS%20AND%20issuetype%20=%20%22New%20Feature%22%20AND%20resolution%20=%20Fixed%

1337 https://jira.duraspace.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project%20=%20DS%20AND%20issuetype%20in%20%28Task,%20%22Code%20Task%22,%20Documentation,%20Sub-task%29%20AND%20resolution%20=%20Fixed%20AND%20fixVersion%20=%20%221.7.0%22%20ORDER%20BY%20key%20ASC&tempMax=

key	summary	assignee	reporter

Can't show details. Ask your admin to add this Jira URL to the allowlist.
View these issues in Jira[1338]

7.6.8 Changes in 1.6.x

- Changes in DSpace 1.6.2 (see page 765)
- Changes in DSpace 1.6.1 (see page 765)
- Changes in DSpace 1.6.0 (see page 766)

7.6.8.1 Changes in DSpace 1.6.2

key	summary	assignee	reporter

Can't show details. Ask your admin to add this Jira URL to the allowlist.
View these issues in Jira[1339]

key	summary	assignee	reporter

Can't show details. Ask your admin to add this Jira URL to the allowlist.
View these issues in Jira[1340]

7.6.8.2 Changes in DSpace 1.6.1

key	summary	assignee	reporter

1338 https://jira.duraspace.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project%20=%20DS%20AND%20issuetype%20=%20Bug%20AND%20resolution%20=%20Fixed%20AND%20fixVersion

1339 https://jira.duraspace.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project%20=%20DS%20AND%20issuetype%20in%20%28Task,%20%22Code%20Task%22,%20Documentation,%20Sub-task%29%20AND%20resolution%20=%20Fixed%20AND%20fixVersion%20=%20%221.6.2%22%20ORDER%20BY%20key%20ASC&tempMax=

1340 https://jira.duraspace.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project%20=%20DS%20AND%20issuetype%20=%20Bug%20AND%20resolution%20=%20Fixed%20AND%20fixVersion

Can't show details. Ask your admin to add this Jira URL to the allowlist.
View these issues in Jira[1341]

key	summary	assignee	reporter

Can't show details. Ask your admin to add this Jira URL to the allowlist.
View these issues in Jira[1342]

7.6.8.3 Changes in DSpace 1.6.0

key	summary	assignee	reporter

Can't show details. Ask your admin to add this Jira URL to the allowlist.
View these issues in Jira[1343]

key	summary	assignee	reporter

Can't show details. Ask your admin to add this Jira URL to the allowlist.
View these issues in Jira[1344]

key	summary	assignee	reporter

1341 https://jira.duraspace.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project%20=%20DS%20AND%20issuetype%20in%20%28%22New%20Feature%22,%20Task,%20Improvement,%20%22Code%20Task%22,%20Documentation,%20Sub-task%29%20AND%20resolution%20=%20Fixed%20AND%20fixVersion%20=%20%221.6.1%22%20ORDER%20BY%20key%20ASC&tempMax=

1342 https://jira.duraspace.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project%20=%20DS%20AND%20issuetype%20=%20Bug%20AND%20resolution%20=%20Fixed%20AND%20fixVersion

1343 https://jira.duraspace.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project%20=%20DS%20AND%20issuetype%20=%20%22New%20Feature%22%20AND%20resolution%20=%20Fixed%

1344 https://jira.duraspace.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project%20=%20DS%20AND%20issuetype%20in%20%28Task,%20%20%22Code%20Task%22,%20Documentation,%20Sub-task%29%20AND%20resolution%20=%20Fixed%20AND%20fixVersion%20=%20%221.6.0%22%20ORDER%20BY%20key%20ASC&tempMax=

Can't show details. Ask your admin to add this Jira URL to the allowlist.
View these issues in Jira[1345]

7.6.9 Changes in 1.5.x

7.6.9.1 Changes in DSpace 1.5.2

key	summary	assignee	reporter

Can't show details. Ask your admin to add this Jira URL to the allowlist.
View these issues in Jira[1346]

key	summary	assignee	reporter

Can't show details. Ask your admin to add this Jira URL to the allowlist.
View these issues in Jira[1347]

key	summary	assignee	reporter

Can't show details. Ask your admin to add this Jira URL to the allowlist.
View these issues in Jira[1348]

1345 https://jira.duraspace.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project%20=%20DS%20AND%20issuetype%20=%20Bug%20AND%20resolution%20=%20Fixed%20AND%20fixVersion
1346 https://jira.duraspace.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project%20=%20DS%20AND%20issuetype%20=%20%22New%20Feature%22%20AND%20resolution%20=%20Fixed%
1347 https://jira.duraspace.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project%20=%20DS%20AND%20issuetype%20in%20%28Task,%20%22Code%20Task%22,%20Documentation,%20Sub-task%29%20AND%20resolution%20=%20Fixed%20AND%20fixVersion%20=%20%221.5.2%22%20ORDER%20BY%20key%20ASC&tempMax=
1348 https://jira.duraspace.org/secure/IssueNavigator.jspa?reset=true&jqlQuery=project%20=%20DS%20AND%20issuetype%20=%20Bug%20AND%20resolution%20=%20Fixed%20AND%20fixVersion

7.6.9.2 Changes in DSpace 1.5.1

General Improvements and Bug Fixes in 1.5.1

(Scott Philips)

- (Scott Phillips) Fixed bug where users could not finish registering nor reset their password because the authentication method signatures were changed.
- Jay Paz (SF#1898241) Additional fixes to patch to enable reuse of methods.
- Added the ability to manage sessions with site wide alerts to prevent users from authenticating.
- Fixes a bug where the ability to edit an item durring workflow step 2 is not displayed.
- Jay Paz (SF#1898241) Add item Export from jspui and xmlui.
- Added easy support for google analytics statistics
- Added the ability for super admins to login as other users.
- Added an activity viewer to the Control Panel

(Mark Diggory)

- Fix for SF Bug #2082236 Subscription notification (sub-daily) no emails sent
- #2102580 William Hays: Duplicate Handle exception when replacing bitstreams
- #2102617 Sands Fish: X509Authentication fails to assign appropriate specialgroups
- (Sands Fish) Add "Select Primary Bitstream" functionality to submission workflow
- Guard against Community/Collection metadata having only whitespace characters and eliminate cases where null pointer exceptions would be thrown
- Improve DSIndexer logic in both branches to support removal of items from index when withdrawn from repository.
- (Sands Fish) Provides fix for AuthenticationUtil where users ID's are not properly compared.
- Fix NullPointerException cause by nullified Context object in LNI map item to new collection.
- Block Basic Authentication "details" from being exposed in dspace logs.
- (Bill Hays) Close InputStreamReaders explicitly to release any file handles back to OS.
- correct linking on pages when xmlui is the ROOT webapplication
- correct issue with sitemap redirection of mydspace uri.
- Add serlet-api to overlay wars to reduce compile time errors when adding classes
- Correct issues in feed generation
- XMLUI Adjust Advanced Search to use search properties from dspace.cfg.
- Correct bug in Body.toSAX where startElement is called instead of end element.
- Correct issue with libraries being excluded from wars

(Claudia Juergen)

- Fix for SF bug #2090761 Statistics wrong use of dspace.dir for log location
- Fix for SF bug #2081930 xmlui hardcoded strings in EditGroupForm.java
- Fix for SF bug #2080319 jspui hardcoded strings in browse
- Fix for SF bug #2078305 xmlui hardcoded strings used in UI in xmlui-api
- Fix for SF bug #2078324 xmlui hardcoded strings used in UI in General-Handler.xsl
- SF patch #2076066 Review in jspui submission non-dc metadata
- SF Bug #1983859 added Foreign Lucene Analyzers to poms
- SF Bug #1989916 - missing LDAP authentication key

(Stuart Lewis)

- #1947036 Patch for SF Bug1896960 SWORD authentication and LDAP + 1989874 LDAPAuthentication pluggable method broken for current users
- Added copying of registration email template to 1.4 to 1.5 upgrade instructions

- Fix for SF bug #2055941 LDAP authentication fails for new users in SWORD and Manakin

(Zuki Ebetsu / Stuart Lewis)

- #1990660 SWORD Service Document are malformed / Corrected Atom publishing MIME types

(Stuart Lewis / Claudia Juergen)

- Updated installation and configuration documents for new statistics script, and removed references to Perl

(Tim Donohue)

- Fix for SF bug #2095402 - Non-interactive Submission Steps don't work in JSPUI 1.5
- Fix for SF bug #2013921 - Movement in Submission Workflow Causes Skipped Steps
- Fix for SF bug #2015988 - Configurable Submission bug in SubmissionController
- Fix for SF bug #2034372 - Resorting Search Results in JSPUI always gives no results
- Updates to Community/Collection Item Counts (i.e. strengths) for XMLUI.
- 1.5 upgrade instructions were missing Metadata Registry updates necessary to support SWORD.

(Graham Triggs)

- Fix various problems with resources potentially not being freed, and other minor fixes suggested by FindBugs
- Replace URLEncoder with StringEscapeUtils for better fix of escaping the hidden query field
- Fix #2034372 - Resorting in JSPUI gives no results
- Fix #1714851 - set eperson.subscription.onlynew in dspace.cfg to only include items that are new to the repository
- Fix issue where the browse and search indexes will not be updated correctly if you move an Item
- Fix problem with SWORD not accepting multiple concurrent submissions
- Fix #1963060 Authors listed in reverse order
- Fix #1970852 - XMLUI: Browse by Issue Date "Type in Year" doesn't work
- Statistics viewer for XMLUI, based on existing DStat. Note that this generates the view from the analysis files (.dat), does not require HTML report generation.
- Fixed incorrect downloading of bitstream on withdrawn item
- Add JSPUI compatible log messages to XMLUI transformers
- Clean up use of ThreadLocal
- Improved cleanup of database resources when web application is unloaded
- Fix bug #1931799 - duplicate "FROM metadatavalue&quot
- Fixed Oracle bugs with ILIKE operators and LIMIT/OFFSET clauses

7.6.9.3 Changes in DSpace 1.5.0

General Improvements in 1.5.0

- Highly configurable and theme-able new user interface (Manakin).
- Apache Maven-based modular build system.
- LNI (Lightweight Network Interface) service. Allows programmatic ingest of content via WebDAV or SOAP.
- SWORD (Simple Web-service Offering Repository Deposit): repository-standard ingest service using Atom Publishing Protocol.
- Highly configurable item web submission system. All submission steps are configurable not just metadata pages.
- Browse functionality allowing customisation of the available indexes via dspace.cfg and pluggable normalisation of the sort strings. Integration with both JSP-UI and XML-UI included.
- Extensible content event notification service.
- Generation of Google and HTML sitemaps

Bug fixes and smaller patches in 1.5.0

- New options for ItemImporter to support bitstream permissions and descriptions.
- 1824710 Fix - Change in Creative Commons RDF.
- 1794700 Fix - Stat-monthly and stat-report-monthly
- 1566820 Patch - Authentication code moved to new org.dspace.authenticate package, add IP AUth
- 1670093 Patch - More stable metadata and schema registry import Option to generate community and collection "strength" as a batch job
- 1659868 Patch - Improved database level debugging
- 1620700 Patch - Add Community and Sub-Community to OAI Sets
- 1679972 Fix - OAIDCCrosswalk NPE and invalid character fix, also invalid output prevented
- 1549290 Fix - Suggest Features uses hard coded strings
- 1727034 Fix - Method MetadataField.unique() is incorrect for null values
- 1614546 Fix - Get rid of unused mets_bitstream_id column
- 1450491 Patch - i18n configurable multilingualism support
- 1764069 Patch - Replace "String" with "Integer" in PreparedStatement where needed
- 1743188 Patch - for Request #1145499 - Move Items
- 179196 Patch - Oracle SQL in Bitstream Checker
- 1751638 Patch - Set http disposition header to force download of large bitstreams
- 1799575 Patch - New EPersonConsumer event consumer
- 1566572 Patch - Item metadata in XHTML head elements
- 1589429 Patch - "Self-Named" Media Filters (i.e. MediaFilter Plugins) (updated version of this patch)
- 1888652 Patch - Statistics Rewritten In Java
- 1444364 Request - Metadata registry exporter
- 1221957 Request - Admin browser for withdrawn items
- 1740454 Fix - Concurrency
- 1552760 Fix - Submit interface looks bad in Safari
- 1642563 Patch - bin/update-handle-prefix rewritten in Java
- 1724330 Fix - Removes "null" being displayed in community-home.jsp
- 1763535 Patch - Alert DSpace administrator of new user registration
- 1759438 Patch - Multilingualism Language Switch - DSpace Header

7.6.10 Changes in 1.4.x

- Changes in DSpace 1.4.1(see page 770)
- Changes in DSpace 1.4.0(see page 772)

7.6.10.1 Changes in DSpace 1.4.1

General Improvements in 1.4.1

- Error pages now return appropriate HTTP status codes (e.g. 404 not found)
- Bad filenames in /bitstream/ URLs now result in 404 error – prevents infinite URL spaces confusing crawlers and bad "persistent" bitstream IDs circulating
- Prevent infinite URL spaces in HTMLServlet
- InstallItem no longer sets dc.format.extent, dc.format.mimetype; no longer sets default value for dc.language.iso if one is not present
- Empty values in drop-down submit fields are not added as empty metadata values
- API methods for searching epeople and groups
- Support stats from both 1.3 and 1.4

- [dspace]/bin/update-handle-prefix now runs index-all
- Remove cases of System.out from code executed in webapp
- Change "View Licence" to "View License" in Messages.properties
- dspace.cfg comments changed to indicate what default.language actually means
- HandleServlet and BitstreamServlet support If-Modified-Since requests
- Improved sanity-checking of XSL-based ingest crosswalks
- Remove thumbnail filename from alt-text
- Include item title in HTML title element
- Improvements to help prevent spammers and sploggers
- Make cleanup() commit outstanding work every 100 iterations
- Better handling where email send failed due to wrong address for new user
- Include robots.txt to limit bots navigating author, date and browse by subject pages
- Add css styles for print media
- RSS made more configurable and provide system-wide RSS feed, also moves text to Messages.properties
- Jar file updates (includes required code changes for DSIndexer and DSQuery and new jars fontbox.jar and serializer.jar)
- Various documentation additions and cleanups
- XHTML compliance improvements
- Move w3c valid xhtml boiler image into local repository
- Remove uncessary Log4j Configuration in CheckerCommand
- Include Windows CLASSPATH in dsrun.bat

Bug fixes in 1.4.1

- 1604037 - UIUtil.encodeBitstream() now correctly encodes URLs (no longer incorrectly substitutes '+' for spaces in non-query segment
- 1592984 - Date comparisons strip time in org.dspace.harvest.Harvest
- 1589902 - Duplicate field checking error on input-forms.xml
- 1596952 - Collection Wizard create Template missing schema
- 1596978 - View unfinished submissions - collection empty
- 1588625 - Incorrect text on item mapper screen
- 1597805 - DIDL Crosswalk: wrong resource management
- 1605635 - NPE in Utils.java
- 1597504 - Search result page shows shortened query string
- 1532389 - Item Templates do not work for non-dc fields
- 1066771 - Metadata edit form dropping DC qualifier
- 1548738 - Multiple Metadata Schema, schema not shown on edit item page
- 1589895 - Not possible to add unqualified Metadata Field
- 1543853 - Statistics do not work in 1.4
- 1541381 - Browse-by-date and browse-by-title not working
- 1556947 - NullPointerException when no user selected to del/edit
- 1554064 - Fix exception handling for ClassCastException in BitstreamServlet
- 1548865 - Browse errors on withdrawn item
- 1554056 - Community/collection handle URL with / redirects to homepage
- 1571490 - UTF-8 encoded characters in licence
- 1571519 - UTF-8 in statistics
- 1544807 - Browse-by-Subject/Author paging mechanism broken
- 1543966 - "Special" groups inside groups bug
- 1480496 - Cannot turn off "ignore authorization" flag!
- 1515148 - Community policies not deleting correctly
- 1556829 - Docs mention old SiteAuthenticator class
- 1606435 - Workflow text out of context
- Fix for bitstream authorization timeout

- Fix to make sure cleanup() doesn't fail with NullPointerException
- Fix for removeBitstream() failing to update primary bitstream
- Fix for Advanced Search ignoring conjunctions for arbitrary number of queries
- Fix minor bug in Harvest.java for Oracle users
- Fix missing title for news editor page
- Small Messages.properties modification (change of DSpace copyright text)
- fix PDFBox tmp file issue
- Fix HttpServletRequest encoding issues
- Fix bug in TableRow toString() method where NPE is thrown if tablename not set
- Update DIDL license and change coding style to DSpace standard

7.6.10.2 Changes in DSpace 1.4.0

General Improvements in 1.4.0

- Content verification through periodic checksum checking
- Support for branded preview image
- Add/replace Creative Commons in 'edit item' tool
- Customisable item listing columns and browse indices
- Script for updating handle prefixes (e.g. for moving from development to production)
- Configurable boolean search operator
- Controlled vocabulary patch to provide search on classification terms, and addition of terms during submission.
- Add 'visibility' element to input-forms.xml
- Browse by subject feature
- Log4J enhancement to use XML configuration
- QueryArgs class can support any number of fields in advanced search.
- Community names no longer have to be unique
- Enhanced Windows support
- Support for multiple (flat) metadata schemas
- Suggest an item page
- RSS Feeds
- Performance enhancements
- Stackable authentication methods
- Plug-in manager
- Pluggable SIP/DIP support and metadata crosswalks
- Nested groups of e-people
- Expose METS and MPEG-21 DIDL DIPs via OAI-PMH
- Configurable Lucene search analyzer (e.g. for Chinese metadata)
- Support for SMTP servers requiring authentication

Bug fixes in 1.4.0

- 1358197 - Edit Item, empty DC fields not removable
- 1363633 - Submission step 1 fails when there are no collections
- 1255264 - Resource policy eperson value was set to wrong column
- 1380494 - Error deleting an item with multiple metadata schema support
- 1443649 - Cannot configure unqualified elements for advanced search index
- 1333687 - Browse-(title|date) fails on withdrawn item
- 1066713 - Two (sub)communities cannot have one name
- 1284055 - Two Communities of same name throws error
- 1035366 - Bitstream size column should be bigint

- 1352257 - Selecting a Group for GroupToGroup while Creating Collection
- 1352226 - Navigation and Sorting in Group List (Select Groups) fails
- 1348276 - Null in collection name causes OAI ListSets to fail
- 1160898 - dspace_migrate removes Date.Issued from prev published items
- 1261191 - Malformed METS metadata exported

7.6.11 Changes in 1.3.x

- Changes in DSpace 1.3.2(see page 773)
- Changes in DSpace 1.3.1(see page 773)
- Changes in DSpace 1.3.0(see page 773)

7.6.11.1 Changes in DSpace 1.3.2

General Improvements in 1.3.2

- DSpace UI XHTML/WAI compliant
- Configure metadata fields shown on simple item display
- Supervisor/workspace help documentation

Bug fixes in 1.3.2

- Oracle compatibility fixes
- Item exporter now correctly exports metadata in UTF-8
- fixed to handle 'null' values passed in

7.6.11.2 Changes in DSpace 1.3.1

Bug fixes in 1.3.1

- 1252153 - Error on fresh install

7.6.11.3 Changes in DSpace 1.3.0

General Improvements in 1.3.0

- Initial i18n Support for JSPs - Note: the implementation of this feature required changes to almost all JSP pages
- LDAP authentication support
- Log file analysis and report generation
- Configurable item licence viewing
- Supervision order/collaborative workspace administrative tools
- Basic workspace for submissions in progress, with support for supervision
- SRB storage system option
- Updated handle server system
- Database optimisations
- Latest versions of Xerces, Xalan and OAICAT jars
- Various documentation additions and cleanups

Bug fixes in 1.3.0

- 1161459 - ItemExporter fails with Too many open files
- 1167373 - Email date field not populated
- 1193948 - New item submit problem
- 1188132 - NullPointerException when Adding EPerson
- 1188016 - Cannot Edit an Eperson
- 1219701 - Unable to open unfinished submission
- 1206836 - community strengths not reflecting sub-community
- 1238262 - Submit UI nav/progress buttons no longer show progress
- 1238276 - Double quote problem in some fields in submit UI
- 1238277 - format support level not shown in "uploaded file" page
- 1242548 - Uploading non-existing files
- 1244743 - Bad lookup key for special case of DC Title in ItemTag.java
- 1245223 - Subscription Emailer fails
- 1247508 - Error when browsing item with no content/bitstream collections
- Set the content type in the HTTP header
- Fix issue where EPerson edit would not work due to form indexing (partial fix)
- POST handling in HTMLServlet
- Missing ContentType directives added to some JSPs
- Name dependency on Collection Admin and Submitter groups fixed
- Fixed OAI-PMH XML encoding

7.6.12 Changes in 1.2.x

- Changes in DSpace 1.2.2(see page 774)
- Changes in DSpace 1.2.1(see page 775)
- Changes in DSpace 1.2.0(see page 776)

7.6.12.1 Changes in DSpace 1.2.2

General Improvements in 1.2.2

- Customisable submission forms added
- Configurable number of index terms in Lucene for full-text indexing
- Improved scalability in media filter
- Submit button on collection pages only appears if user has authorisation
- PostgreSQL 8.0 compatibility
- Search scope retention to improve browsing
- Community and collection strengths displayed
- Upgraded OAICat software

Bug fixes in 1.2.2

- Fix for Oracle too many cursors problem.
- Fix for UTF-8 encoded searches in advanced search.
- Fix for handling "\" in bitstream names.
- Fix to prevent delete of "unknown" bitstream format
- Fix for ItemImport creating new handles for replaced items

Changes in JSPs in 1.2.2

- *collection-home.jspchanged*
- *community-home.jspchanged*
- *community-list.jspchanged*
- *home.jspchanged*
- *dspace-admin/list-formats.jspchanged*
- *dspace-admin/wizard-questions.jspchanged*
- *search/results.jspchanged*
- *submit/cancel.jspchanged*
- *submit/change-file-description.jspchanged*
- *submit/choose-file.jspchanged*
- *submit/complete.jspchanged*
- *submit/creative-commons.jspchanged*
- *submit/edit-metadata.jspnew*
- *submit/get-file-format.jspchanged*
- *submit/initial-questions.jspchanged*
- *submit/progressbar.jspchanged*
- *submit/review.jspchanged*
- *submit/select-collection.jspchanged*
- *submit/show-license.jspchanged*
- *submit/show-uploaded-file.jspchanged*
- *submit/upload-error.jspchanged*
- *submit/upload-file-list.jspchanged*

7.6.12.2 Changes in DSpace 1.2.1

General Improvements in 1.2.1

- Oracle support added
- Thumbnails in item view can now be switched off/on
- Browse and search thumbnail options
- Improved item importer
    - can now import to multiple collections
    - added --test flag to simulate an import, without actually making any changes
    - added --resume flag to try to resume the import in case the import is aborted
- Configurable fields for the search index
- Script for transferring items between DSpace instances
- Sun library JARs (JavaMail, Java Activation Framework and Servlet) now included in DSpace source code bundle

Bug fixes in 1.2.1

- A logo to existing collection can now be added. Fixes SF bug #1065933
- The community logo can now be edited. Fixes SF bug #1035692
- MediaFilterManager doesn't 'touch' every item every time. Fixes SF bug #1015296
- Supported formats help page, set the format support level to "known" as default
- Fixed various database connection pool leaks

Changed JSPs in 1.2.1

- *collection-homechanged*
- *community-homechanged*
- *display-itemchanged*
- *dspace-admin/confirm-delete-collectionmoved to tools/ and changed*
- *dspace-admin/confirm-delete-communitymoved to tools/ and changed*
- *dspace-admin/edit-collectionmoved to tools/ and changed*
- *dspace-admin/edit-communitymoved to tools/ and changed*
- *dspace-admin/indexchanged*
- *dspace-admin/upload-logochanged*
- *dspace-admin/wizard-basicinfochanged*
- *dspace-admin/wizard-default-itemchanged*
- *dspace-admin/wizard-permissionschanged*
- *dspace-admin/wizard-questionschanged*
- *help/formats.htmlremoved*
- *help/formatschanged*
- *indexchanged*
- *layout/navbar-adminchanged*

7.6.12.3 Changes in DSpace 1.2.0

General Improvements in 1.2.0

- Communities can now contain sub-communities
- Items may be included in more than one collection
- Full text extraction and searching for MS Word, PDF, HTML, text documents
- Thumbnails displayed in item view for items that contain images
- Configurable MediaFilter tool creates both extracted text and thumbnails
- Bitstream IDs are now persistent - generated from item's handle and a sequence number
- Creative Commons licenses can optionally be added to items during web submission process

Administration

- If you are logged in as administrator, you see admin buttons on item, collection, and community pages
- New collection administration wizard
- Can now administer collection's submitters from collection admin tool
- Delegated administration - new 'collection editor' role - edits item metadata, manages submitters list, edits collection metadata, links to items from other collections, and can withdraw items
- Admin UI moved from /admin to /dspace-admin to avoid conflict with Tomcat /admin JSPs
- New EPerson selector popup makes Group editing much easier
- 'News' section is now editable using admin UI (no more mucking with JSPs)

Import/Export/OAI

- New tool that exports DSpace content in AIPs that use METS XML for metadata (incomplete)
- OAI - sets are now collections, identified by Handles ('safe' with /, : converted to _)
- OAI - contributor.author now mapped to oai_dc:creator[1349]

1349 http://oai_dccreator

Miscellaneous

- Build process streamlined with use of WAR files, symbolic links no longer used, friendlier to later versions of Tomcat
- MIT-specific aspects of UI removed to avoid confusion
- Item metadata now rendered to avoid interpreting as HTML (displays as entered)
- Forms now have no-cache directive to avoid trouble with browser 'back' button
- Bundles now have 'names' for more structure in item's content

JSP file changes between 1.1 and 1.2

This list generated with *cvs -Q rdiff -s -r dspace-1_1 dspace* and a sprinkling of perl.

- Changed: dspace/jsp/collection-home.jsp
- Changed: dspace/jsp/community-home.jsp
- Changed: dspace/jsp/community-list.jsp
- Changed: dspace/jsp/display-item.jsp
- Changed: dspace/jsp/index.jsp
- Changed: dspace/jsp/home.jsp
- Changed: dspace/jsp/styles.css.jsp
- Moved to dspace-admin and changed: dspace/jsp/admin/authorize-advanced.jsp
- Moved to dspace-admin and changed: dspace/jsp/admin/authorize-collection-edit.jsp
- Moved to dspace-admin and changed: dspace/jsp/admin/authorize-community-edit.jsp
- Moved to dspace-admin and changed: dspace/jsp/admin/authorize-item-edit.jsp
- Moved to dspace-admin and changed: dspace/jsp/admin/authorize-main.jsp
- Moved to dspace-admin and changed: dspace/jsp/admin/authorize-policy-edit.jsp
- Moved to dspace-admin: dspace/jsp/admin/collection-select.jsp
- Moved to dspace-admin: dspace/jsp/admin/community-select.jsp
- Moved to dspace-admin: dspace/jsp/admin/confirm-delete-collection.jsp
- Moved to dspace-admin: dspace/jsp/admin/confirm-delete-community.jsp
- Moved to dspace-admin: dspace/jsp/admin/confirm-delete-dctype.jsp
- Moved to dspace-admin: dspace/jsp/admin/confirm-delete-eperson.jsp
- Moved to dspace-admin: dspace/jsp/admin/confirm-delete-format.jsp
- Moved to dspace/jsp/tools: dspace/jsp/admin/confirm-delete-item.jsp
- Moved to dspace/jsp/tools: dspace/jsp/admin/confirm-withdraw-item.jsp
- Moved to dspace-admin and changed: dspace/jsp/admin/edit-collection.jsp
- Moved to dspace-admin and changed: dspace/jsp/admin/edit-community.jsp
- Moved to dspace/jsp/tools and changed: dspace/jsp/admin/edit-item-form.jsp
- Moved to dspace-admin and changed: dspace/jsp/admin/eperson-browse.jsp
- Moved to dspace-admin: dspace/jsp/admin/eperson-confirm-delete.jsp
- Moved to dspace-admin and changed: dspace/jsp/admin/eperson-edit.jsp
- Moved to dspace-admin and changed: dspace/jsp/admin/eperson-main.jsp
- Moved to dspace/jsp/tools and changed: dspace/jsp/admin/get-item-id.jsp
- Moved to dspace/jsp/tools and changed: dspace/jsp/admin/group-edit.jsp
- Moved to dspace-admin and changed: dspace/jsp/admin/group-eperson-select.jsp
- Moved to dspace/jsp/tools and changed: dspace/jsp/admin/group-list.jsp
- Moved to dspace-admin: dspace/jsp/admin/index.jsp
- Moved to dspace-admin and changed: dspace/jsp/admin/item-select.jsp
- Moved to dspace-admin and changed: dspace/jsp/admin/list-communities.jsp
- Moved to dspace-admin and changed: dspace/jsp/admin/list-dc-types.jsp
- Removed: dspace/jsp/admin/list-epeople.jsp
- Moved to dspace-admin and changed: dspace/jsp/admin/list-formats.jsp
- Moved to dspace/jsp/tools: dspace/jsp/admin/upload-bitstream.jsp

- Moved to dspace-admin and changed: dspace/jsp/admin/upload-logo.jsp
- Moved to dspace-admin: dspace/jsp/admin/workflow-abort-confirm.jsp
- Moved to dspace-admin and changed: dspace/jsp/admin/workflow-list.jsp
- Changed: dspace/jsp/browse/authors.jsp
- Changed: dspace/jsp/browse/items-by-author.jsp
- Changed: dspace/jsp/browse/items-by-date.jsp
- Changed: dspace/jsp/browse/no-results.jsp
- New: dspace-admin/eperson-deletion-error.jsp
- New: dspace/jsp/dspace-admin/news-edit.jsp
- New: dspace/jsp/dspace-admin/news-main.jsp
- New: dspace/jsp/dspace-admin/wizard-basicinfo.jsp
- New: dspace/jsp/dspace-admin/wizard-default-item.jsp
- New: dspace/jsp/dspace-admin/wizard-permissions.jsp
- New: dspace/jsp/dspace-admin/wizard-questions.jsp
- Changed: dspace/jsp/components/contact-info.jsp
- Changed: dspace/jsp/error/internal.jsp
- New: dspace/jsp/help/formats.jsp
- Changed: dspace/jsp/layout/footer-default.jsp
- Changed: dspace/jsp/layout/header-default.jsp
- Changed: dspace/jsp/layout/navbar-admin.jsp
- Changed: dspace/jsp/layout/navbar-default.jsp
- Changed: dspace/jsp/login/password.jsp
- Changed: dspace/jsp/mydspace/main.jsp
- Changed: dspace/jsp/mydspace/perform-task.jsp
- Changed: dspace/jsp/mydspace/preview-task.jsp
- Changed: dspace/jsp/mydspace/reject-reason.jsp
- Changed: dspace/jsp/mydspace/remove-item.jsp
- Changed: dspace/jsp/register/edit-profile.jsp
- Changed: dspace/jsp/register/inactive-account.jsp
- Changed: dspace/jsp/register/new-password.jsp
- Changed: dspace/jsp/register/registration-form.jsp
- Changed: dspace/jsp/search/advanced.jsp
- Changed: dspace/jsp/search/results.jsp
- Changed: dspace/jsp/submit/cancel.jsp
- New: dspace/jsp/submit/cc-license.jsp
- Changed: dspace/jsp/submit/choose-file.jsp
- New: dspace/jsp/submit/creative-commons.css
- New: dspace/jsp/submit/creative-commons.jsp
- Changed: dspace/jsp/submit/edit-metadata-1.jsp
- Changed: dspace/jsp/submit/edit-metadata-2.jsp
- Changed: dspace/jsp/submit/get-file-format.jsp
- Changed: dspace/jsp/submit/initial-questions.jsp
- Changed: dspace/jsp/submit/progressbar.jsp
- Changed: dspace/jsp/submit/review.jsp
- Changed: dspace/jsp/submit/select-collection.jsp
- Changed: dspace/jsp/submit/show-license.jsp
- Changed: dspace/jsp/submit/show-uploaded-file.jsp
- Changed: dspace/jsp/submit/upload-error.jsp
- Changed: dspace/jsp/submit/upload-file-list.jsp
- Changed: dspace/jsp/submit/verify-prune.jsp
- New: dspace/jsp/tools/edit-item-form.jsp
- New: dspace/jsp/tools/eperson-list.jsp
- New: dspace/jsp/tools/itemmap-browse.jsp

- New: dspace/jsp/tools/itemmap-info.jsp
- New: dspace/jsp/tools/itemmap-main.jsp

7.6.13 Changes in 1.1.x

7.6.13.1 Changes in DSpace 1.1.1

Bug fixes in 1.1.1

- non-administrators can now submit again
- installations now preserve file creation dates, eliminating confusion with upgrades
- authorization editing pages no longer create null entries in database, and no longer handles them poorly (no longer gives blank page instead of displaying policies.)
- registration page Invalid token error page now displayed when an invalid token is received (as opposed to internal server error.) Fixes SF bug #739999
- eperson admin 'recent submission' links fixed for DSpaces deployed somewhere other than at / (e.g. /dspace).
- help pages Link to help pages now includes servlet context (e.g. '/dspace'). Fixes SF bug #738399.

Improvements in 1.1.1

- *bin/dspace-info.pl* now checks jsp and asset store files for zero-length files
- *make-release-package* now works with SourceForge CVS
- eperson editor now doesn't display the spurious text 'null'
- item exporter now uses Jakarta's cli command line arg parser (much cleaner)
- item importer improvements:
    - now uses Jakarta's cli command line arg parser (much cleaner)
    - imported items can now be routed through a workflow
    - more validation and error messages before import
    - can now use email addresses and handles instead of just database IDs
    - can import an item to a collection with the workflow suppressed

7.6.13.2 Changes in DSpace 1.1

- Fixed various OAI-related bugs; DSpace's OAI support should now be correct. Note that harvesting is now based on the new Item 'last modified' date (as opposed to the Dublin Core *date.available* date.)
- Fixed Handle support--DSpace now responds to naming authority requests correctly.
- Multiple bitstream stores can now be specified; this allows DSpace storage to span several disks, and so there is no longer a hard limit on storage.
- Search improvements:
    - New fielded searching UI
    - Search results are now paged
    - Abstracts are indexed
    - Better use of Lucene API; should stop the number of open file handles getting large
- Submission UI improvements:
    - now insists on a title being specified

    - fixed navigation on file upload page
    - citation & identifier fields for previously published submissions now fixed
- Many Unicode fixes to the database and Web user interface
- Collections can now be deleted
- Bitstream descriptions (if available) displayed on item display page
- Modified a couple of servlets to handle invalid parameters better (i.e. to report a suitable error message instead of an internal server error)
- Item templates now work
- Fixed registration token expiration problem (they no longer expire.)