

Contents

1 Overview

2 Workflow

- ◆ 2.1 Handshaking from the Dryad Perspective
 - ◇ 2.1.1 User Submits to Dryad First
 - ◇ 2.1.2 User Submits to TreeBASE First, Links from Dryad Record
 - ◇ 2.1.3 User submits to TreeBASE, Dryad harvests only
- ◆ 2.2 Handshaking from the TreeBASE Perspective
- ◆ 2.3 Process for completing a submission within TreeBASE
 - ◇ 2.3.1 Minimum Requirements
 - ◇ 2.3.2 Detailed Process

3 Debugging

4 Running Via Script

5 Implementation

Overview

This page describes the Dryad BagIt Handshaking module. This module enables Dryad to share data packages with other repositories using BagIt as a transfer protocol. Currently, it is used to share NEXUS data between Dryad and TreeBASE.

Workflow

Handshaking from the Dryad Perspective

There are three possible ways that Dryad may integrate with TreeBASE.

User Submits to Dryad First

1. User submits Nexus to Dryad and selects to send the file to TreeBASE in the final stage of submission.
2. Dryad pushes object to TreeBASE. (This is before the object is curated in Dryad)
 1. Metadata (converted to Dryad Application Profile) and all uploaded nexus files are packed into a BagIt package and pushed to TreeBASE
 2. TreeBASE has a PUT RESTful service for receiving data
 3. TreeBASE only accepts the PUT if the sender's IP is within the Dryad range
 4. TreeBASE responds by returning an URL
3. Dryad emails the user to inform that file(s) have been uploaded to TreeBASE and includes a URL to start submission at TreeBASE
4. User goes to TreeBASE and completes record.
 1. Upon the user logging in, TreeBASE is triggered to unpack the !BagIt package and create a submission based on the contents
5. Dryad harvests TreeBASE content and picks up any additional relevant metadata or updated files.

BagIt Handshaking

User Submits to TreeBASE First, Links from Dryad Record

1. User submits/edits package in Dryad and includes a TB ID
2. Dryad adds TB ID to the record and creates a link to the HTML view of the TB record
3. Dryad curator checks link to confirm that TB ID looks valid (we decided to leave ID in record if the link isn't active yet but looks like a valid TB ID).
4. General link checking software will be run to catch dead links that have been dead for awhile.

User submits to TreeBASE, Dryad harvests only

1. Treat as any other harvested content (second class)

Handshaking from the TreeBASE Perspective

1. Dryad offers an option to upload files to TreeBASE for Dryad submissions that have phylogenetic data
2. Choosing this causes Dryad to send a BagIt package to TreeBASE via rest service:
<http://www.treebase.org/treebase-web/handshaking/dryadImport> (there is a different URL for testing -- <http://treebasedb-dev.nescent.org/treebase-web>)
3. The REST service stores the BagIt package and returns a unique URL as a string, resembling:
"~/treebase-web/login.jsp?importKey=123456789"
4. Dryad emails this URL to the user and asks the user to follow it so as to finish the submission on the TreeBASE website
5. Whoever follows the URL will have the ownership of the data. The user is confronted with a login page, including the option of creating a new account. If a new account is required, so long as the user remains within the same session (i.e. does not start a new browser), the !BagIt is unpacked and the contents become parked under a new submission that is created in the user's account.
6. The user can open the new submission and continue filling in information. The new submission should have citation information already filled in and any NEXUS files in the !BagIt will have been parsed and stored.
7. TreeBASE does not make content available until the associated article is published.

Process for completing a submission within TreeBASE

Minimum Requirements

nexus file

- ◆ at least one tree OR at least one matrix
- ◆ if there is a tree and a matrix, the taxon labels must match up.
- ◆ must be "understood" by Mesquite

citation

analysis info linking matrices and trees

Detailed Process

1. create account
2. login
3. create new submission
4. type title
 1. the submission gets a PURL at this point

BagIt Handshaking

2. the PURL can have a code added for reviewer access
5. fill in citation
 1. minimum: year, title, journal name (or book/section title)
 2. journal names auto-suggest as you type
6. add authors
 1. minimum: at least one author (with first name and last name)
 2. must always search for an existing author first, even if you know they're not in the system
 3. allows reordering or deleting authors while you're in the process
7. upload file(s)
 1. minimum: must be nexus, as described above
8. (optional) add notes
 1. this is a textarea, with a reasonable character limit (not enough for a readme file)
9. (optional) edit details for matrices
10. (optional) edit row segment template
 1. minimum: row ID, start index, end index
11. (optional) provide more details for trees
12. (optional) taxa
 1. match all named taxa against ubio or ncbi
 2. although the cleanup is optional, the TB editor may reject it if it's not cleaned up
13. analysis
 1. minimum: create an analysis with at least one step. Typically, this will be a matrix that is processed to create one or more trees.
 2. minimum: otu labels must match in the analysis steps
14. when initial submission complete, user clicks "change to ready state"
 1. this triggers the curator to look at it
 2. user can leave items as "in progress" as long as they want -- this is a "poor man's embargo" system

Debugging

If changes in the metadata break the metadata crosswalk, the first place to look should be in the DRYAD-V3 metadata crosswalk (in the crosswalks folder in the config directory). Since it can be cumbersome to test this piece, try adding a root element to the XML that is generated as a result of the crosswalk. Do this by wrapping the contents of the root XSLT template in a root element. This should be in the XSLT already and just need to be uncommented.

The next step to debugging a metadata crosswalk failure is to use the XMLVERBATIM metadata crosswalk that is also in the crosswalks folder inside of the config directory. This XSLT will just spit out the XML that is coming out of the DSpace Item to XML transformation. Looking at this should enable you to determine where the Dryad stylesheet varies from the output XML.

Running Via Script

There is a script that exports Dryad data packages and requested data files into the BagIt format and sends the BagIt package to the TreeBASE web service. The script is implemented as DSpace package disseminator and can be run through the standard DSpace disseminator script. For example:

BagIt Handshaking

```
/opt/dryad/bin/dspace packager -d -i 10255/dryad.630 -e submitter@email.org  
-o xwalk=DRYAD-V3 -o repo=TREEBASE -o  
'files=10255/dryad.631;10255/dryad.632;10255/dryad.633;10255/dryad.634' -t  
BAGIT -
```

/opt/dryad/bin/dspace = the standard DSpace interface to calling scripts from the command line; this must be run as the same user as the Tomcat serving DSpace is run as.

packager -d = the specific DSpace packager script; the -d tells it to run the packager to disseminate (there isn't a bagit ingester yet, but -i would be used if there were).

-i 10255/dryad.630 = the Dryad data package's pseudo-handle (that DSpace uses as an internal identifier); this must be the pseudo-handle of a data package, not a data file.

-e submitter@email.org = the email address of the person who is making the submission to Dryad; we have this because someone must be logged in in order to submit to Dryad.

-o xwalk=DRYAD-V3 = this is a hard-coded value at the moment (will always be the same); it says to export the metadata stored in DSpace in the Dryad Application Metadata Profile format (version 3 is the latest version).

-o repo=TREEBASE = this is also hard-coded for TreeBASE; repo just indicates the remote repository to which the bagit file should be uploaded to (in the future, Dryad may send !BagIt files to other repositories as well).

-o files=10255/dryad.631;10255/dryad.632;10255/dryad.633;10255/dryad.634 = the list of data files, associated with the data package, that the submitter wants exported to TreeBASE; it may not be all the data files, but all the files of a particular type (for instance, in the case of TreeBASE, Nexus files). For calling the script from the command line, this argument will probably have to be enclosed in single quotes (see the example above).

-t BAGIT = the package disseminator that should be used to export the files; in the case of TreeBASE, this is always hard-coded to BAGIT because that is the format that the TreeBASE importer supports.

- = This indicates that output should be sent to the command line; currently, the script follows the UNIX philosophy that 'no response' is an indicator of success (optionally, the code can be uncommented to allow the output of 'success' on a successful export). If there is a problem, an exception will be thrown and an email with details will be sent to the address configured as the DSpace admin.

Implementation

The BagIt disseminator is implemented as a DSpace plugin.

It is configured in dspace.cfg

Java code is in modules/bagit/dspace-bagit-api/src/main/java/org/dspace/content/packager

XSL for transforming the metadata is in config/crosswalks/dryad-v3.xsl

Created March 14 2011 by Kevin S. Clarke

Contents

- [1 Overview](#)
- [2 Instructions](#)
- [3 Technical Documentation](#)
- [4 Design History](#)

Overview

Dryad assigns Digital Object Identifiers (DOIs) to each Dryad Data Package and Data File. These persistent identifiers provide an easy way to cite and access data deposited in the Dryad Data Repository. Dryad registers its DOIs and their associated metadata through the DataCite service and they are resolvable through the DOI.org system.

Instructions

On submission to Dryad, authors will receive an email that informs the submitter that a DOI will be assigned on the data package's acceptance into Dryad.

[Template that is used for this email](#)

Once the Dryad curator has reviewed and approved the submission, a second email will be sent. This email includes a DOI that the author can use to cite the data package.

[Template that is used for this email](#)

Dryad's page on [recommended data citation practices](#) details the use of the Dryad assigned DOIs.

Technical Documentation

[DOI Service](#)

[DOI Usage](#) -- specification that defines how DOIs are used in Dryad

[EZID service](#) -- service from the California Digital Library that manages DOI registration with [DataCite](#)

Design History

Our implementation was influenced by the California Digital Library (CDL)'s [NOID](#) project.

Dryad's DOI Services were first developed to work with the [German National Library of Science and Technology](#) (TIB)'s DataCite registration service and then later adopted to use the California Digital Library's [EZID](#) DOI registration service.

Ongoing development will take place in conjunction with DSpace's nascent Identifier Services module.

Contents

1 Wiki Standards

- ◆ 1.1 General requirements
- ◆ 1.2 Requirements for technical features

2 Places and tools

- ◆ 2.1 Storage of working drafts and documents
- ◆ 2.2 Presentations
- ◆ 2.3 Mockups
- ◆ 2.4 Diagrams

3 Questions

Wiki Standards

Standards for documentation that appears on this wiki.

General requirements

Every page on this wiki should:

1. Include introductory text describing the basic purpose of the page
2. If the document is not current, include a statement about the status. This statement should include an overview of the revision history. Use the keyword "outdated" in the status description.
3. Include appropriate headings. Typically, any section longer than one screenfull should have its own heading. The major headings on a page should use the 2-equal-sign markup, and subheadings should use 3- or 4-equal-sign markup.
4. Link to other relevant pages
5. Be placed in at least one category

When marking a page as outdated:

1. Edit/add the status statement at the beginning of the page. Include a summary of the revision history.
2. Move the document so its name begins with "Old:".
 1. NOTE: We're not using a namespace for "Old", because MediaWiki won't let you move pages into a namespace.
 2. NOTE: pages in the WG namespace should **not** be moved to Old, since that would make them public. Just leave them as WG.
3. Look for any pages that link to this page, and inspect the links. Some links may need to be updated to point to a newer page.

Requirements for technical features

For each technical feature, there should be a wiki page with the following sections:

1. **Overview** -- in plain language, bullet points of what the feature means for a typical user
2. **Instructions** -- step-by-step instructions, with screenshots highlighting the use of the feature. This may include a workflow diagram.

BagIt Handshaking

3. **Technical Documentation** -- brief list of links to technical documentation pages on the google code wiki or elsewhere
4. **Design history** -- link to any wiki pages or other resources used while designing the feature

The wiki page for the feature should:

1. be linked from any appropriate places on the "main" datadryad.org site (e.g. "click here for help")
2. be linked from the repository development plan (where the feature was first released)
3. be categorized under the Help category and the Features category, as well as any other appropriate categories

The technical documentation pages should be located in the Google Code wiki (they are part of the Dryad *software*, not the Dryad *project*). Each page will contain the following sections:

1. **Overview** -- repeat the text from the other wiki page.
2. **Functionality** or **Usage** -- Detailed description of the feature's scope and how the feature is invoked. (Don't repeat too much of the documentation from the Project wiki).
3. **Workflow** -- a workflow diagram, indicating what pieces of code are active as the feature is used, along with a set of numbered steps describing the workflow in text.
4. **Configuration** -- description of configuration options available for this feature
5. **Relation to DSpace** -- Indicate how this feature overrides or changes the default features in DSpace. Link to any relevant pages in the DSpace wiki.

Note: Some features implemented by @mire already have documentation in PDF form. Initially, we will link to these PDFs. Later, we may break them up into wiki pages.

Places and tools

Places where project documentation is created, developed and stored. The Dryad project requires creation of many documents. This section attempts to distill the software and locations in use, and their best uses.

(Yes, this list is a mix of file formats, applications, and locations. It's messy.)

Dryad web site

- ◆ finalized documents that are needed by day-to-day users of Dryad

Dryad Development Wiki (see Standards above)

- ◆ Anything that needs to be communicated publicly
- ◆ Final versions of PR materials are on the Publicity Material page
- ◆ Most documentation belongs here, but not documents that need to be kept confidential

GoogleDocs

- ◆ For files on which multiple people are working, especially spreadsheets (because this is hard to do on the wiki) and anything that requires concurrent editing
- ◆ Most useful while a document is initially being created. Some documents remain here, but others are exported to more permanent formats
- ◆ For information that isn't public, but keep in mind that this isn't truly secure
- ◆ It can also be used to privately share one or a handful of documents that aren't used for concurrent editing (e.g. PDFs)

Microsoft Word

BagIt Handshaking

- ◆ The late stages of many formal documents (publications, grant proposals, etc.)

PDF

- ◆ The final stages of PR and other documents that need to be widely read. (Many documents are transformed to PDF before placement on this wiki.)

WebDav

- ◆ For the most private information
- ◆ Only put things here if they need the security, because it is less convenient to use
- ◆ It is appropriate if there is a large number of files and the person who needs them will be accessing the directory on a regular basis.

Dropbox

- ◆ Mostly an option for backing up an individual's files
- ◆ Not currently being used much for sharing and not everyone has an account, also not useful for concurrent editing

Google Code Repository

- ◆ Documentation associated with the Dryad *software*, as opposed to the Dryad *project*.

WordPress'

- ◆ Stores completed blog entries (i.e., short-to-medium length time-sensitive news and PR pieces), as well as drafts of pending blog entries.

Storage of working drafts and documents

Dropbox

- ◆ best for large files requiring collaboration

Google Docs

- ◆ only works for the document types that Google supports

Presentations

PowerPoint

Mockups

Balsalmiq

Diagrams

(still being discussed)

Creately -- Online. Excels at creating/managing connections between objects

PowerPoint -- Versatile. Doesn't support Linux.

Dia -- Free

OpenOffice Draw

Rejected:

OmniGraffle -- Mac only

Google Docs -- really doesn't do well for diagrams

Questions

QUESTION: Do we want to keep curation manuals and software manuals on the wiki?

If so, any opinions about their organization and how to make them more prominent than all the other curation-related or software-related information on the wiki?

ANSWERS SO FAR: Software manuals will be generated from wiki content, using a list of pages that are included in each manual.

QUESTION: What are preferences for working on shared files?

GoogleDocs seems to be getting the most use - is that what everyone prefers to use? Do people like and use Dropbox more than it seems? Are there other sharing practices that should be added here?

ANSWERS SO FAR: Dropbox isn't good for working on shared documents concurrently, but it does obviate the need to send around a copy of the latest version.

QUESTION: Do we need to have the blog content backed up somewhere?

Handshaking is the process of coordinating submission between Dryad and specialized repositories in order to (a) lower user burden by streamlining the submission workflow and (b) allow Dryad and specialized repositories to exchange identifiers and other metadata in order to enable cross-referencing of the different data products associated with a given publication.

Target repositories

The initial NSF grant names two handshaking partners TreeBASE and GenBank, which Dryad's partner journals have previously identified as required points of deposition for phylogenetic tree data and DNA sequences, respectively. If the handshaking experiments with these repositories are successful in achieving the goals above, Dryad will work to implement handshaking mechanisms with additional specialized repositories as needed to satisfy the data deposition requirements of its partner journals.

Overview of process

1. User starts manuscript submission at journal
2. User is given a link for data deposition at Dryad
3. At Dryad, user is asked whether they have data that should be deposited at another repository
4. If yes, then depending on the repository:
 - ◆ Dryad captures the initial submission files and passes those to the specialized repository. After completing the Dryad submission, the user then follows a link to complete the submission at the external repository. OR
 - ◆ Dryad either opens a session at the external repository and the user goes through the full deposit process at the external site. The Dryad submission can be completed at any point during or after this process.
 - ◆ In both cases, publication metadata and data identifiers are passed between the repositories automatically.

Status of work

Pages describing the design and implementation status of handshaking for each repository are listed below:

GenBank: [GenBank Submission Integration](#)

TreeBASE: [TreeBASE Submission Integration](#), [Use Cases](#), [TreeBASE OAI-PMH implementation](#), [notes \(private\)](#)

♦ Obsolete: [TreeBASE SWORD specs](#)

Contents

[1 Overview](#)

[2 Instructions](#)

[3 Technical](#)

[Documentation](#)

[4 Design History](#)

Overview

Dryad harvests the contents of partner repositories in order to provide a one stop entry point to cross-searching relevant materials. Current partner repositories harvested by Dryad include [TreeBASE](#) and [KNB](#). Materials harvested into Dryad are also searched when a search of the Dryad collection is performed. Results from the partner repositories are displayed in search tabs along with the Dryad search results.

Instructions

The integration of partner repository materials is visible when a search of Dryad is performed. The search results page will return a list of Dryad results and, in separate tabs, results from the same search performed against the harvested partner repository resources. Before clicking the partner repository tab, a searcher will be able to see how many results are available from the hit count displayed on the search results tab.

Search Results		
Dryad (8)	TreeBASE (5)	KNB (14)
Kisel Y, Barraclough TG (2010) Data from: Speciation has a spatial scale that depends on levels of gene flow. <i>The American Naturalist</i> doi:10.5061/dryad.887		
Mitchell A, Mitter C, Regier JC (2000) Data from: More taxa or more characters revisited: combining data from nuclear protein-encoding genes for phylogenetic analyses of Noctuoidea (Insecta: Lepidoptera). <i>Systematic Biology</i> doi:10.5061/dryad.558		
Kuo MM, Sperling FAH, Caterino MS, Reed RD (2001) Data from: A partitioned likelihood analysis of swallowtail butterfly phylogeny (Lepidoptera: Papilionidae). <i>Systematic Biology</i> doi:10.5061/dryad.615		
Walters JR, Hardcastle TJ (2011) Data from: Getting a full dose? Reconsidering sex chromosome dosage compensation in the silkworm, <i>Bombyx mori</i> . <i>Genome Biology and Evolution</i> doi:10.5061/dryad.8716		
Altermatt F, Pearse IS (2011) Data from: Similarity and specialization of the larval versus adult diet of European butterflies and moths. <i>The American Naturalist</i> doi:10.5061/dryad.cb6pk		
Ohshima I, Yoshizawa K (2010) Data from: Differential introgression causes genealogical discordance in host races of <i>Acrocercops transecta</i> (Insecta: Lepidoptera). <i>Molecular Ecology</i> doi:10.5061/dryad.1390		
Groot A, Classen A, Staudacher H, Schal C, Heckel D (2010) Data from: Phenotypic plasticity in sexual communication signal of a noctuid moth. <i>Journal of Evolutionary Biology</i> doi:10.5061/dryad.1956		
Santos H, Burban C, Rousset J, Rossi J, Branco M, Kerdelhué C (2010) Data from: Incipient allochronic speciation in the pine processionary moth <i>Thaumetopoea pityocampa</i> (Lepidoptera: Notodontidae). <i>Journal of Evolutionary Biology</i> doi:10.5061/dryad.2001		

Technical Documentation

More detail on the OAI-PMH harvesting of KNB/TreeBASE content can be found on the [Harvesting Technology](#) page

Technical details on the way Dryad handles the display of the partner repository content in the search results can be found on the [Tabbed Searching Technology](#) page.

Design History

Mockups from the design phase of the Tabbed Searching implementation [are available](#). The design of Dryad's harvesting capabilities builds on those of [DSpace's](#).

Contents

1 Installing the Dryad Data Repository Software

- ◆ 1.1 Installing the Prerequisite Software
 - ◇ 1.1.1 Installing on Ubuntu Linux (9.10)
- ◆ 1.2 Preparing the Local System
 - ◇ 1.2.1 Creating a dryad user (Ubuntu Linux)
 - ◇ 1.2.2 Setting up symlinks and scripts
- ◆ 1.3 Downloading Dryad's Source Code
 - ◇ 1.3.1 Checking Out the Dryad Source Code on an Ubuntu Linux Machine
 - ◇ 1.3.2 Compiling Dryad on an Ubuntu Linux Machine
 - ◇ 1.3.3 Compiling Dryad on an OSX Machine
- ◆ 1.4 Initializing Dryad's Postgresql Database

◇ 1.4.1 Initializing the Database on an Ubuntu Linux Machine

- ◆ 1.5 Updating Dryad Prior to (Re)Deploying
- ◆ 1.6 Deploying Dryad to Tomcat
- ◆ 1.7 Proxying Tomcat with Apache
- ◆ 1.8 Running Jenkins to Manage Deployment
- ◆ 1.9 Other installation issues
- ◆ 1.10 Scripts to Maintain Dryad
- ◆ 1.11 Useful resources
- ◆ 1.12 Final Notes and Potential Pitfalls

Installing the Dryad Data Repository Software

Dryad is a repository for data sets underlying scientific publications, with an initial focus on evolution, ecology, and related fields. It is built on top of DSpace, an open source digital repository software package. There are no official releases of Dryad yet, but you can check its source code out of the project's Google Code repository and build it yourself. This isn't difficult, but does require that you have a few prerequisites installed.

Installing the Prerequisite Software

Required software:

- subversion client
- Java
- Maven (version 2.2.1 is recommended. maven 3 will work, but may require different command arguments)
- Ant
- PostgreSQL
- Perl
- Apache Tomcat

Highly recommended:

- Apache Web Server (to proxy Tomcat)
- Hudson/Jenkins (running on jetty, so it is able to restart Tomcat)
- Emacs

Since Dryad and DSpace are Java-based projects, they will run on a variety of operating systems. For the purposes of this guide, we will step through installing Dryad on an Ubuntu Linux system.

Installing on Ubuntu Linux (9.10)

The core prerequisites for the Dryad project can all be found in the Ubuntu software package management system. Aptitude may be used to install them:

```
sudo aptitude install subversion
sudo aptitude install sun-java6-jdk
sudo aptitude install maven2
sudo aptitude install ant
sudo aptitude install postgresql
```

BagIt Handshaking

```
sudo aptitude install perl
sudo aptitude install tomcat6
sudo aptitude install apache2
```

One thing to check before proceeding is that the newly installed Sun JDK/JRE is the default JDK/JRE for your Ubuntu system. We need to do this because Ubuntu comes with OpenJDK installed as the default JDK/JRE. Unfortunately, though, OpenJDK does not work with our version of Cocoon, the framework on which parts of the DSpace user interface are built. So, we need to check the system's default JDK/JRE and set it to be Sun's (if it isn't already). To do this type:

```
sudo update-alternatives --config java
sudo update-alternatives --config javac
```

If either of these isn't set to use the Sun version of java or javac, select the number associated with the Sun version so that it will be used as the default.

Preparing the Local System

Typically, when running on a server, Dryad runs under the 'dryad' account. This is not required, but subsequent instructions on this page assume the 'dryad' user is present. Dryad will run out of a normal user account -- just make the appropriate adjustments in the commands below.

Creating a dryad user (Ubuntu Linux)

Creating a 'dryad' User on an Ubuntu Linux System First, we want to create a dryad user on the machine:

```
sudo useradd -m dryad
sudo passwd dryad
```

All following commands on this page should be run as the dryad user.

Setting up symlinks and scripts

Dryad installations run by the core Dryad team always use a standard set of directories and commands. Create symlinks and scripts so these standard directories/commands point to the actual installed locations on your machine. See [Standard Server Configuration](#).

Downloading Dryad's Source Code

Checking the Dryad source code out of the Google Code repository will download the project's source code to your machine. Once this is done, as we will see, Dryad can be compiled and deployed to a Web server running on the local machine.

Checking Out the Dryad Source Code on an Ubuntu Linux Machine

If you are not a member of the Dryad project, you can go to the terminal and check out the source code by 1) changing into the dryad user's home directory (/home/dryad) and 2) typing:

```
svn checkout http://dryad.googlecode.com/svn/trunk/ dryad
```

BagIt Handshaking

If you are a member of the Dryad project (i.e., you have an account on Google Code with permission to edit Dryad files), you can type:

```
svn checkout https://dryad.googlecode.com/svn/trunk/ dryad --username [username]
```

Compiling Dryad on an Ubuntu Linux Machine

Dryad uses Maven profiles to configure local instances of the Dryad application. Where before, one would edit the `${dspace.dir}/config/dspace.cfg` file, now a Maven profile should be created and the configuration information put in there. The `dspace.cfg`, as it is checked out of Subversion, is set up to read its configuration values from this Maven profile configuration.

Below is a sample Maven profile (the `${MAVEN_HOME}/conf/settings.xml` file) that can be modified for custom use:

```
<?xml version="1.0" encoding="UTF-8"?>
<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0 http://maven.apache.org/xsd/settings-1.0.0">
  <profiles>
    <profile>
      <id>env-dev</id>
      <properties>
        <!-- This is where you want Dryad installed after the build -->
        <default.dspace.dir>/opt/dryad</default.dspace.dir>
        <default.dspace.hostname>localhost</default.dspace.hostname>
        <default.dspace.port>9999</default.dspace.port>
        <!--
          We suggest dryad_repo as the repo name and dryad_app as the db user
        -->
        <default.db.url>jdbc:postgresql://localhost:5432/dryad_repo</default.db.url>
        <default.db.username>dryad_app</default.db.username>
        <default.db.password>mydbpassword</default.db.password>
        <!-- The DataCite username/password that has permission to register DOIs -->
        <default.doi.username>myusername</default.doi.username>
        <default.doi.password>mypassword</default.doi.password>
        <!-- DOI prefix; 10.5061 is the official Dryad prefix -->
        <default.doi.prefix>10.5061</default.doi.prefix>
        <!-- Whether URLs and DOIs should resolve locally; true for dev instance -->
        <default.dryad.localize>true</default.dryad.localize>
        <!-- Whether DOIs should be registered; false for a dev instance -->
        <default.doi.datacite.connected>false</default.doi.datacite.connected>
        <!-- Mail setup; for development, it's easiest to use a GMail account -->
        <default.mail.server>smtp.gmail.com</default.mail.server>
        <default.mail.server.username>mygmailaccount@gmail.com</default.mail.server.username>
        <default.mail.server.password>mygmailpassword</default.mail.server.password>
        <!-- A configuration for using GMail as the mail server -->
        <default.mail.extraproperties>mail.smtp.socketFactory.port=465,
          mail.smtp.socketFactory.class=javax.net.ssl.SSLSocketFactory,
          mail.smtp.socketFactory.fallback=false</default.mail.extraproperties>
        <!-- A configuration for using localhost -->
        <default.mail.extraproperties>mail.smtp.localhost=SUBDOMAIN.HOSTNAME.TLD</default.mail.extraproperties>
      </properties>
    </profile>
  </profiles>
  <!-- In dev, these are usually the same, but may be different in production -->
  <default.mail.admin>mygmailaccount@gmail.com</default.mail.admin>
  <default.mail.help>mygmailaccount@gmail.com</default.mail.help>
</settings>
```

BagIt Handshaking

```
<!-- The email of the DSpace account the harvester should be run as -->
<default.harvester.eperson>mygmailaccount@gmail.com</default.harvester.eperson>
<!-- The location of the Solr server that indexes Dryad content -->
<default.solr.search.server>http://localhost:9999/solr/search/</default.solr.search.server>
</properties>
</profile>
</profiles>
</settings>
```

The next step is to change into the directory that was created by the Subversion check-out process. We want, then, to change directories into the dspace directory of the Dryad subproject (e.g., the dryad/dryad directory):

```
cd dryad/dryad/dspace
```

If you've used a directory for the dspace.dir value that doesn't already exist, you'll need to create the new directory; this can be done with the mkdir command. For instance:

```
sudo mkdir /opt/dryad
```

You will also need to change the ownership of this directory to the dryad user:

```
sudo chown dryad /opt/dryad
```

Once this is done, you are ready to build Dryad. For its build management system, Dryad uses Maven 2. This means to compile the Dryad package, you need run a simple command from the dspace directory (e.g., dryad/dryad/dspace). Since you should still be in this directory, type:

```
mvn clean package -P env-dev
```

(Replace "env-dev" with the name of the maven profile you set above.) When this is done, Dryad has been successfully compiled. The next step is to set up the PostgreSQL database and deploy the Dryad application to a location from which it can be served by Tomcat.

Compiling Dryad on an OSX Machine

Same as above, but the default Maven location is: /usr/share/maven

Initializing Dryad's PostgreSQL Database

While DSpace, and so Dryad, uses Maven 2 for compiling, it uses Ant for deploying the application to a Web server. The steps for doing this are very simple but there are a few additional configurations that need to take place first. Again, we'll walk through the steps for a Linux system, but they should be generalizable to a Windows or Mac OS X system as well.

Initializing the Database on an Ubuntu Linux Machine

First, we need to setup the PostgreSQL database so that it is ready to accept the DSpace/Dryad setup. First, we create a dryad user in the database (you will need to know the password of the postgres user on the machine):

Compiling Dryad on an Ubuntu Linux Machine

BagIt Handshaking

```
sudo -u postgres createuser -U postgres -d -A -P dryad_app
```

Next, we create the database into which Dryad will install itself:

```
sudo -u dryad createdb -U dryad_app -E UNICODE dryad_repo
```

If you have trouble connecting the database user after you have successfully created the `dryad_app` user, it might be that your postgresql database is set up for ident authentication instead of password authentication. To change this, edit your `pg_hba.conf` file (where this is will depend on your system), changing ident to md5 (see the postgresql documentation for more details).

Also, if you don't know the postgres admin password, but have sudo privileges on the machine, you can reset the database password with a couple of steps (if you don't need to reset the database's password, you can skip this step).

First, shutdown the database:

```
sudo /etc/init.d/postgresql-8.4 stop
```

Next, edit the `pg_hba.conf` file using nano:

```
sudo nano /etc/postgresql/8.4/main/pg_hba.conf
```

Change the "local all postgres md5" line to "local all postgres trust" and restart the server:

```
sudo /etc/init.d/postgresql-8.4 start
```

You can then change the postgres password by typing:

```
psql -U postgres template1 -c "alter user postgres with password '[newpassword]';"
```

Once this is done, you need to shutdown the server:

```
sudo /etc/init.d/postgresql-8.4 stop
```

You can then change back the `/etc/postgresql/8.4/main/pg_hba.conf` file to use "md5" and restart the server:

```
sudo /etc/init.d/postgresql-8.4 start
```

Once the database is ready, we can use the Ant script included in the DSpace/Dryad distribution to finish initializing everything. The Ant script uses the configuration that we modified in the `config/dspace.cfg` file (so will install Dryad to the location set in the `dspace.dir` variable). If you encounter any problems, check the settings in the `dspace.cfg` file to make sure that they are what you think they should be.

To run the Ant script, we need to change into the build directory (which was created as a result of running the Maven build script). To do this, type:

```
cd target/dspace-*.dir
```

Next, run Ant to initialize the database:

BagIt Handshaking

```
ant fresh_install
```

If something should fail in the initialization process, you may want to go back to the compile step and repeat the process. To do this, you'll first need to erase what has been previously built. Maven provides a task (or command) for this.

First, move up two directories into the main dspace directory:

```
cd ../..
```

Then, run the Maven clean command which will remove the target directory with all the compiled code:

```
mvn clean
```

You will also, most likely, want to start with a clean database install as well:

```
dropdb -U dryad_app dryad_repo
```

You can then repeat the database creation and compilation steps, addressing whatever issues presented themselves in the first pass.

Lastly, you want to create an administrator for your Dryad instance; when you run the following script, you will be walked through this process (this should be run from the main dryad/dryad/dspace directory):

```
bin/create-administrator
```

Updating Dryad Prior to (Re)Deploying

After the initialization of the Dryad database, you want to run an update that will pull any changes you have made to the Dryad codebase into the instance that is to be deployed using Tomcat. There are a variety of parameters that can be used in the update that will indicate whether you want to update just the webapp, the webapp and the Java code, or the webapp, Java code, and configuration files. The first update command below will update the configuration and pick up on any changes that might have been made to the codebase.

From the target/dspace-*.dir directory, type:

```
ant -Doverwrite=true update
```

To just update the webapps (if there haven't been any changes to the Java code) type:

```
ant update_webapps
```

Dryad can also be built referencing a particular dspace.cfg file if needed:

```
ant -Dconfig=/home/dryad/dspace.cfg update
```

If you find changes that you are expecting to see are not showing up, you can always do a clean install of the codebase using "sudo -u dryad mvn clean package" prior to running the Ant task.

If you are updating a development installation of dryad, you probably want to invoke maven as

BagIt Handshaking

```
mvn clean package -P env-dev -U
```

If maven complains about not recognizing the env-dev profile (message will appear at the end of the maven output) you may want to check the profiles defined in your maven settings ({USER_HOME}/.m2/conf/settings.xml). If you are using maven 3, it may look for user settings elsewhere, use -s to override this. The -U option simply forces maven to check remote repositories for updates.

Deploying Dryad to Tomcat

When running Dryad, the server needs to interact with the Web application in a way that gives it access to directories and databases. This can be accomplished by changing ownership of the deployed directory (in our case, /opt/dryad) to the user running Tomcat (often a tomcat or www-data user) or by having the Tomcat instance run as the dryad user. Which path you choose may be determined by what other webapps you intend to run in Tomcat (and what their needs are) and whether you're using a continuous integration server like Hudson to manage your build processes.

To change the user that Tomcat runs as, edit the /etc/init.d/tomcat6 file using nano:

```
sudo nano /etc/init.d/tomcat6
```

When you edit, find the line that says "TOMCAT6_USER=tomcat6" and change it to "TOMCAT6_USER=dryad". You will then also need to change the permissions on the directories to which that Tomcat needs to write.

Tomcat writes to the cache file:

```
sudo chown -R dryad /var/cache/tomcat6
```

It writes to the lib directory

```
sudo chown -R dryad /var/lib/tomcat6
```

It also writes to the log directory

```
sudo chown -R dryad /var/log/tomcat6
```

You may need to adjust Tomcat's policy files so the Web application has permission to write to the file system. Newer versions of Tomcat on Ubuntu will require this. Other Linux distributions might not, but it doesn't hurt either. First, use nano to edit the "/etc/tomcat6/policy.d/50local.policy" file.

```
sudo nano /etc/tomcat6/policy.d/50local.policy
```

Add the below to the bottom of the file:

```
grant codeBase "file:///opt/dryad/webapps/-" {  
  permission java.security.AllPermission;  
};
```

Next, you need to add the Dryad webapps to the Tomcat /etc/tomcat6/server.xml file (where webapps are statically configured). Replace the Host element in the file with the following:

BagIt Handshaking

```
<Host name="localhost" appBase="/opt/dryad/webapps" unpackWARs="true" autoDeploy="true" xmlValidation="true"
<Context docBase="xmlui" path="" reloadable="true" cachingAllowed="false" allowLinking="true"/>
<Context docBase="solr" path="/solr" reloadable="true" cachingAllowed="false" allowLinking="true">
  <Environment name="solr/home" type="java.lang.String" value="/opt/dryad/solr/" override="true" />
</Context>
<Context docBase="oai" path="/oai" reloadable="true" cachingAllowed="false" allowLinking="true"/>
<Context docBase="doi" path="/doi" reloadable="true" cachingAllowed="false" allowLinking="true"/>
</Host>
```

You will also want to change the port Tomcat runs at (to correspond to the port that we've configured/used in this HowTo); to do this replace:

```
<Connector port="8080" protocol="HTTP/1.1" connectionTimeout="20000" URIEncoding="UTF-8" redirectPort="8443" />
```

with:

```
<Connector port="9999" protocol="HTTP/1.1" connectionTimeout="20000" URIEncoding="UTF-8" redirectPort="8443" />
```

It's worth noting that the port at which Dryad is run determines which Dryad logo is displayed at the top of the page. (The logo-switching logic is in the theme's Dryad.xsl) Below is a list of ports and corresponding Dryad logos:

```
Port 9999 -- development logo
Port 8080 -- production logo
Port 8888 -- staging logo
Port 7777 -- demo logo
Port 6666 -- MRC logo
```

You should now be able to restart Tomcat and have it serve Dryad at the port and URL that you set in the dspace.cfg file.

```
sudo /etc/init.d/tomcat6 restart
```

Proxying Tomcat with Apache

For a user-facing Dryad instance, you will want to configure Apache to proxy Tomcat so that all requests come through port 80 and so that the server is listening to a particular domain. It's possible to do this in Tomcat, of course, but many people prefer to use Apache as the public face for their Web applications.

To proxy the Dryad webapp, just add the following to Apache's configuration file for the domain from which you want to serve Dryad.

```
ProxyPass / http://localhost:9999/
ProxyPassReverse / http://localhost:9999/
```

For your own development purposes, it is not required that you proxy Tomcat with Apache. Working with Tomcat running at another port (like 9999) works just fine.

Notes:

1. The production instance of Dryad uses many more settings for proxies and redirects. It is best to copy a

BagIt Handshaking

configuration file from an existing server.

2. The default "htdocs" directory for apache should contain copies of the Dryad maintenance page, logo, and favicon.

Running Jenkins to Manage Deployment

Jenkins is a great tool to manage the build/deploy cycle of the code. It does take a bit of setup. Issues to remember:

Jenkins must run on a servlet container other than the Dryad Tomcat, because it needs to restart the Dryad Tomcat.

We normally use Jetty for the alternate servlet container.

Jetty must run on different ports than Tomcat, so they do not interfere with each other.

Other installation issues

It is useful to ensure that all Dryad services start when a machine is rebooted. The services that should start include:

Postgres
Tomcat
Apache
Jetty
Handle server

Scripts to Maintain Dryad

There are a series of DSpace and Dryad scripts that you might want to set in your machine's cron jobs. These keep Dryad running as expected. Below is an example crontab -l listing:

```
#Send out subscription e-mails at 01:00 every day
0 1 * * * /opt/dryad/bin/dspace sub-daily

#Generate sitemaps for Dryad at 01:30 every day (also necessary for DOI generation)
30 1 * * * /opt/dryad/bin/dspace generate-sitemaps

#Run the media filter at 02:00 every day
0 2 * * * /opt/dryad/bin/dspace filter-media

#Run the embargo bit updater at 02:30 every day
30 2 * * * /opt/dryad/bin/dspace embargo-lifter

#Run the checksum checker at 03:00
0 3 * * * /opt/dryad/bin/dspace checker -lpu

#Run the register-dois script at 03:30 to catch new submissions without DOIs
30 3 * * * /opt/dryad/bin/dspace register-dois

#Mail the checker's results to the sysadmin at 04:00
0 4 * * * /opt/dryad/bin/dspace checker-emailer
```

BagIt Handshaking

```
#Run the update-dois script at 4:30 to update DSpace with newly registered DOIs
30 4 * * * /opt/dryad/bin/dspace update-dois
```

```
#Run stat analyses
0 5 * * * /opt/dryad/bin/dspace stat-general
0 5 * * * /opt/dryad/bin/dspace stat-monthly
0 6 * * * /opt/dryad/bin/dspace stat-report-general
0 6 * * * /opt/dryad/bin/dspace stat-report-monthly
```

Useful resources

[Dspace database schema](#)

Final Notes and Potential Pitfalls

The Handle server should only be run on the production machine (not on development or staging machines).

To update a handle service you need to re-run the SimpleSetup which will generate a new sitebndl file. That file will need to be sent to CNRI(hdladmin@cnri.reston.va.us) and will be used to update the prefix(es) in the Global Registry.

For the error "DSpace has failed to initialize, during stage 3. Error while attempting to read the XML UI configuration file", the solution is to check all paths for config files. Make sure Tomcat has the rights to read them. Note that the XMLUI webapp has a WEB-INF/web.xml that contains a path to the dspace.cfg file.

If you're trying to start the handle server and get an error about filename too long, this means that the start script is using a dspace with an echo statement in it. Remove the echo.

For the error "java.net.MalformedURLException: unknown protocol: resource", modify the log4j.properties, so the rootLogger is not set to debug level. See DSpace bug 239.

If you get an error related to the bi_* tables on running bin/index-init or bin/index-update, it might be that the dryad_app user doesn't have permission to create tables in the db (which s/he needs); change that then rerun the index-init to fix and create the tables.

If you get an error "java.lang.RuntimeException: Unable to acquire dispatcher named default", you may be calling one of the old DSpace shell scripts. Use the bin/dspace command instead.

If you get an error about a metadata field, ensure that all required metadata fields exist. "dsrun org.dspace.administer.MetadataImporter? -f config/registries/FILENAME" (replace FILENAME with one of the files in the registries directory)

Email contact for technical support: help@datadryad.org

See Also

[Installation Checklist for Production Upgrade](#)
[Updating Data from Existing Instance](#)

How to safely upgrade the production machine to a new version of the Dryad code. This is an expanded, and more thorough, version of [Updating the Codebase](#).

Before taking tomcat down:

Ensure you have tested what will happen to submissions that are in the workflow system when the

BagIt Handshaking

release is performed.

If there are significant changes to the submission system, create two test submissions -- one that is pushed through to the curator, and one that stays in the user's workspace. These will be used for testing once the upgraded system comes back up.

Update version/date information in modules/xmlui/src/main/webapp/themes/Dryad/meta/version.xml

Tag new release in SVN

Create a backup of the database and metadata

```
pg_dump --inserts dryad_repo >dryadDBexport-2011-9-2.sql
pg_dump dryad_repo >dryadDBexport-2011-9-2-noins.sql
bin/dspace metadata-export -f dryad-backup/packageExport-2011-9-2.csv -i 10255/3
bin/dspace metadata-export -f dryad-backup/fileExport-2011-9-2.csv -i 10255/2
```

Update Maven profile configurations if necessary

Checkout newly tagged release from SVN

Final test of basic functionality and all new features on the staging server.

- ◆ Be sure to check links between packages and files and that DOIs resolve (to do this, set localize config in dspace.cfg to false)

Use the admin interface to notify users that the system will be going down shortly.

Edit and post the "down for maintenance" page.

```
emacs /var/www/dryad/htdocs/index.html
emacs /opt/coolstack/apache2/conf/httpd.dryad.conf
(apache restart)
```

Run the Hudson build process. When it finishes:

You may need to clear Tomcat's maintenance mode

```
sudo svcs tomcat
sudo svcadm clear tomcat
```

Install the correct robots.txt file by copying from robots.txt.production (this step should be moved into the maven configuration)

Update any configuration, metadata registry, database settings for new features.

```
bin/dsrun org.dspace.administer.MetadataImporter -f config/registries/internal-types.xml -u
```

If any search changes were applied, reindex the discovery index

```
bin/dspace update-discovery-index -f
```

Run through the "smoke tests" on the [Testing](#) page.

Test what happened to the two test submissions that were in the submission workflow during the upgrade.

Remove the "down for maintenance" page.

Re-process any metadata email that was received during the downtime.

If the system has been moved to a new machine

1. Ensure that all of the procedures on Updating Data from Existing Instance have been followed, including:
 - ◆ Re-build the search index
 - ◆ Copy over the statistics index
 - ◆ Copy over crontabs
 - ◆ Copy over the bagit binaries (and set permissions for the dryad user)
2. Ensure that all backup machines are now backing up the correct (new) machine.
3. Update the Handle server on the new machine.

After upgrade is complete

Perform basic testing on the production server.

Edit release notes on wiki.

Tell the team the release is out.

Write a blog entry about the release.

Do the same steps on the staging-mirror system (possibly used for failover)

Contents

[1 Overview](#)

[2 Instructions](#)

[3 Technical](#)

[Documentation](#)

[4 Design History](#)

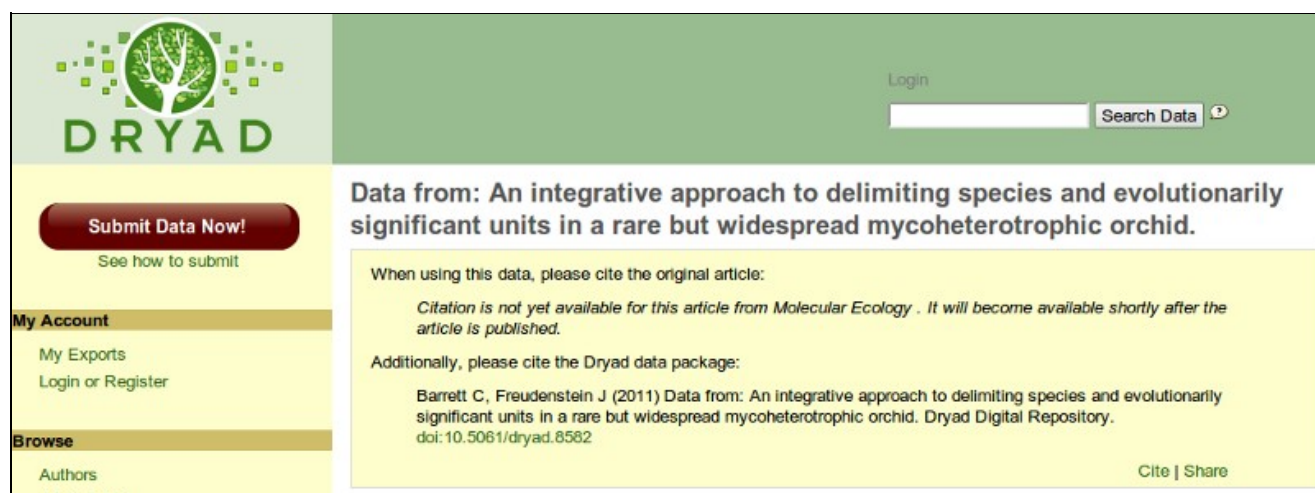
Overview

Dryad enables users to download citations for data packages in the RIS and BibTex formats for importing into their desktop citation management software. Dryad also enables the bookmarking (or "liking") of data packages on a variety of social networking sites (e.g., Facebook, Delicious, CiteULike, Digg, etc.) Both these options take advantage of Dryad assigned DOIs to provide a consistent mechanism for referencing Dryad Data Packages.

Instructions

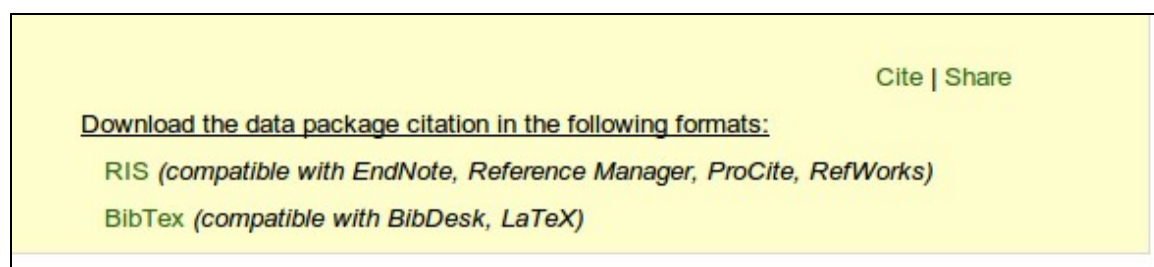
The option to download a citation for a Dryad data package is available underneath the citation display found at the top of every page that displays a Dryad data package. Underneath the citation, on the right side of the page, will be the option to cite or share the Dryad data package information.

BagIt Handshaking



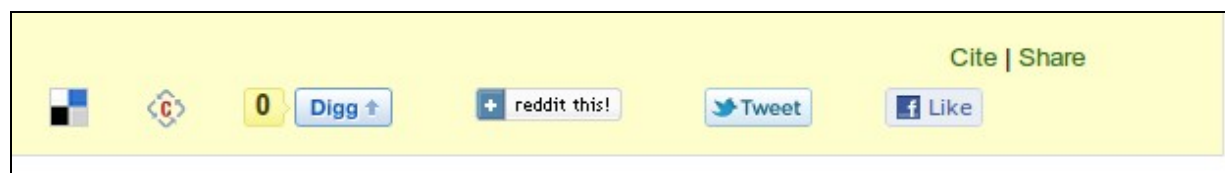
The screenshot shows the Dryad website interface. On the left, there is a navigation menu with links: "Submit Data Now!" (with a sub-link "See how to submit"), "My Account" (with sub-links "My Exports" and "Login or Register"), and "Browse" (with sub-links "Authors" and "Journal Editors"). The main content area features a green header with the Dryad logo and a search bar. Below the header, the title of the data package is displayed: "Data from: An integrative approach to delimiting species and evolutionarily significant units in a rare but widespread mycoheterotrophic orchid." A yellow box contains citation instructions: "When using this data, please cite the original article: Citation is not yet available for this article from Molecular Ecology. It will become available shortly after the article is published. Additionally, please cite the Dryad data package: Barrett C, Freudenstein J (2011) Data from: An integrative approach to delimiting species and evolutionarily significant units in a rare but widespread mycoheterotrophic orchid. Dryad Digital Repository. doi:10.5061/dryad.8582". A "Cite | Share" link is visible in the bottom right corner of the yellow box.

Clicking on the 'cite' option will display information about how the Dryad data package citation can be downloaded to the desktop. Clicking on 'RIS' or 'BibTex' will download the citation in that format.



This screenshot shows a yellow box with the "Cite | Share" link in the top right corner. The text inside the box reads: "Download the data package citation in the following formats:" followed by two options: "RIS (compatible with EndNote, Reference Manager, ProCite, RefWorks)" and "BibTex (compatible with BibDesk, LaTeX)".

Alternatively, Dryad data packages can be bookmarked on a variety of social networking sites (Delicious, CiteULike, Facebook, etc.) To do this, click the 'share' link and then select from the displayed social networking sites.



This screenshot shows a yellow box with the "Cite | Share" link in the top right corner. Below the link, there are several social sharing icons: a small square icon, a CiteULike icon, a Digg icon, a reddit icon, a Twitter icon, and a Facebook Like icon.

Technical Documentation

More information about the technical details behind downloading a data citation and sharing a link to a Dryad Data Package can be found on the [Citation Sharing Technology](#) page.

Design History

Dryad's selection of citation download formats was based on looking at how others in the same space are providing the functionality, in particular the way PLoS ONE [handles it](#). We tested several citation management software packages (EndNote, etc.) to confirm that they could import the format of RIS that we export.

Dryad also used the linking mechanisms suggested by the social networking sites that we support. If a site supplied a !JavaScript to make the link we used that. If a site used an iframe to support the bookmarking activity, we used that.

We create scripts and symlinks wherever possible to keep all machines using the directories/commands listed here, regardless of differences in the underlying OS. This makes system maintenance much easier, and allows our scripts to work correctly on any standard Dryad system.

All content is owned by the dryad user.

DSpace/Tomcat runs as the dryad user.

All major directories are symlinked from the Dryad home directory, for easy access

The handle server should only run on production, because we don't want multiple copies registered with the root handle server.

Handle prefix: 10255

DOI prefix: 10.5061

Dryad/DSpace:

code is built to /opt/dryad (or an instance-specific directory, like dryad-demo)

dspace logs: /opt/dryad/log

Tomcat:

To restart Tomcat: (Dryad home)/bin/tomcat-restart.sh

Tomcat configuration: (Dryad home)/tomcat/conf

Tomcat logs: (Dryad home)/tomcat/logs

Tomcat webapps: (Dryad home)/tomcat/webapps

Apache:

To restart Apache: (Dryad home)/bin/apache-restart.sh

To put the maintenance page in place: (Dryad home)/bin/apache-maint-on.sh

To remove the maintenance page: (Dryad home)/bin/apache-maint-off.sh

Apache configuration: (Dryad home)/apache/conf

Apache logs: (Dryad home)/apache/logs

Apache static pages: (Dryad home)/apache/htdocs

Postgres:

Run the postgres client with the "standard" dryad user and database: postgres-client.sh

Hudson/Jenkins:

BagIt Handshaking

installed at /jenkins on the primary machine name (e.g., <https://datadryad.org/jenkins>)

Maven profile:

(Dryad home)/maven/conf/settings.xml

Contents

- [1 Overview](#)
- [2 Instructions](#)
- [3 Technical Documentation](#)
- [4 Design History](#)

Overview

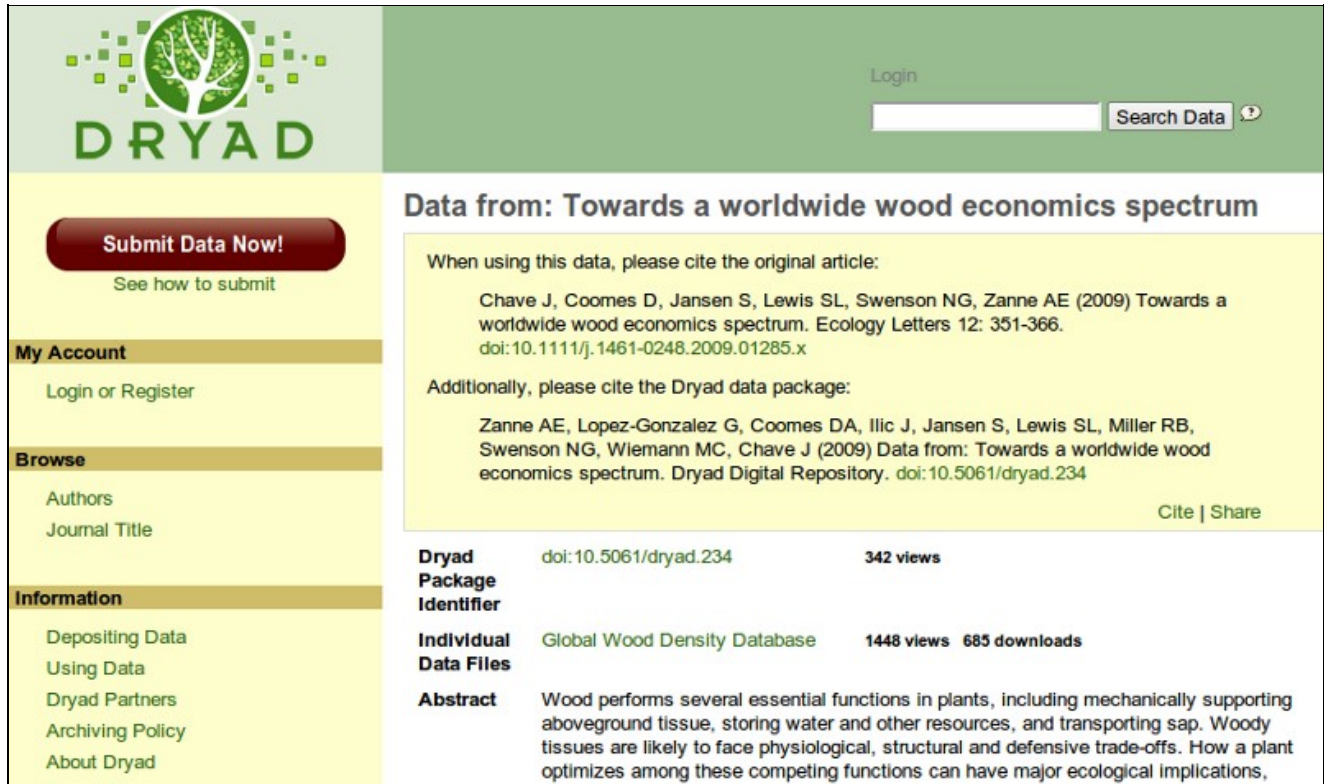
Dryad provides general statistical information about the number of data files and data packages in the repository as well as information about how many times each data package and data file has been downloaded and viewed. These statistics provide an indication about the amount of use that data packages and file receive and give an overall snapshot of Dryad's growth.

Instructions

To see the overall Dryad statistics, just visit the Dryad home page. On this page, near the top, is a sentence that shows the current date and how many data packages and data files are in Dryad at that point in time. This sentence also gives the number of distinct journals represented in Dryad. For an example, see the screenshot below.



There are also statistics on the data package and data file level within Dryad. They look the same except on the Dryad data package only the number of *views* are displayed while on the data file *views* and *downloads* for that particular data file are displayed.



The screenshot shows the Dryad website interface. On the left is a navigation menu with sections: 'My Account' (Login or Register), 'Browse' (Authors, Journal Title), and 'Information' (Depositing Data, Using Data, Dryad Partners, Archiving Policy, About Dryad). The main content area features a header with the Dryad logo and a 'Submit Data Now!' button. Below this, the title 'Data from: Towards a worldwide wood economics spectrum' is displayed. A yellow box contains citation instructions and the original article details: Chave J, Coomes D, Jansen S, Lewis SL, Swenson NG, Zanne AE (2009) Towards a worldwide wood economics spectrum. Ecology Letters 12: 351-366. doi:10.1111/j.1461-0248.2009.01285.x. It also provides the Dryad data package citation: Zanne AE, Lopez-Gonzalez G, Coomes DA, Illic J, Jansen S, Lewis SL, Miller RB, Swenson NG, Wiemann MC, Chave J (2009) Data from: Towards a worldwide wood economics spectrum. Dryad Digital Repository. doi:10.5061/dryad.234. Below this, a table shows statistics for the Dryad Package Identifier (doi:10.5061/dryad.234) with 342 views, and Individual Data Files (Global Wood Density Database) with 1448 views and 685 downloads. An abstract is also provided.

Submit Data Now!
See how to submit

My Account
Login or Register

Browse
Authors
Journal Title

Information
Depositing Data
Using Data
Dryad Partners
Archiving Policy
About Dryad

Data from: Towards a worldwide wood economics spectrum

When using this data, please cite the original article:
Chave J, Coomes D, Jansen S, Lewis SL, Swenson NG, Zanne AE (2009) Towards a worldwide wood economics spectrum. Ecology Letters 12: 351-366. doi:10.1111/j.1461-0248.2009.01285.x

Additionally, please cite the Dryad data package:
Zanne AE, Lopez-Gonzalez G, Coomes DA, Illic J, Jansen S, Lewis SL, Miller RB, Swenson NG, Wiemann MC, Chave J (2009) Data from: Towards a worldwide wood economics spectrum. Dryad Digital Repository. doi:10.5061/dryad.234

Cite | Share

Dryad Package Identifier	doi:10.5061/dryad.234	342 views
Individual Data Files	Global Wood Density Database	1448 views 685 downloads

Abstract
Wood performs several essential functions in plants, including mechanically supporting aboveground tissue, storing water and other resources, and transporting sap. Woody tissues are likely to face physiological, structural and defensive trade-offs. How a plant optimizes among these competing functions can have major ecological implications,

Technical Documentation

The Dryad statistics build on the statistics package provided by DSpace. More information about the implementation can be found on the [Statistics Technology](#) page.

Design History

Dryad's statistics build on DSpace's [Statistics module](#). We also use [Google Analytics](#) so looked at that in the process of adding the Dryad statistics functionality.

Contents

- [1 Overview](#)
- [2 Instructions](#)
- [3 Technical Documentation](#)
- [4 Design history](#)

Overview

Dryad's submission system allows users to quickly and easily submit data packages. Key features include:

1. Integration with journals -- allowing authors to automatically fill the Dryad forms with information from the journal's description of an article, and automatically notifying journal editors when a Dryad submission is completed.
2. Minimal description requirements -- Dryad minimizes the amount of typing/clicking required by submitters. Descriptions are automatically propagated from the article description to descriptions for individual data files. Reasonable defaults are set for the various choices available.
3. Integration with partner repositories -- allowing submitters to use Dryad information as a starting point for submitting content to another repository.

See also the journal-oriented [Submission Integration](#) page.

Instructions

An overview of the Dryad submission process can be seen in the [Dryad submission video](#).

For a journal-oriented view of the process, see [Overview of the submission workflow](#).

Technical Documentation

Technical documents are available on several topics:

[Submission System Technology](#)
[Journal Metadata Processing](#)
[Integration with TreeBASE](#)

Design history

Mockups of submission features:

[Submission System Mockups](#)
[Login-Register Mockup](#)
[TreeBASE Submission Mockups](#)
[Repository Handshaking Mockup](#)

Back-end submission features:

[Submission System Updates Phase 3](#)

Contents

[1 Overview](#)
[2 Instructions](#)

- ◆ [2.1 Initial submission to Dryad](#)
- ◆ [2.2 Initial submission to TreeBASE](#)
- [3 Technical Documentation](#)
- [4 Design History](#)

Overview

Authors who submit content to Dryad have the option to forward their Dryad submission to TreeBASE. This saves the time of the author by automating the submission of data to multiple repositories, and create an explicit link between the entries in the two repositories for easier data reuse.

Alternately, authors who initially submit content to TreeBASE may create a link from a Dryad data package to the relevant item in TreeBASE.

Instructions

When a submitter decides to submit data to Dryad, and logs into the Dryad submission process, there are three simple stages to the submission process:

1. Describe the publication
2. Upload and describe the data files
3. Approve data for publication

The TreeBASE submission integration options occur at steps two and three, depending on whether the data is first being submitted to Dryad or has already been submitted to TreeBASE.

Initial submission to Dryad

At the second stage of the submission process, a submitter will see the option to "choose file" from their local machine. This will upload the data into Dryad. When this has happened, the submission form's page will change to indicate the file size of the uploaded data file.

BagIt Handshaking

Publication
Data from: Sample Data Package, Clarke, Kevin S.

Data file *
Please upload your data file or provide the identifier of a file located in another repository.

File	Size	Format	Action
sample.nexus	195Kb	Nexus	<input type="button" value="Remove"/>

Data file description
Title*:

Description:

If a file has been uploaded through the "choose file" interface, at the last stage in the submission process, the author will be given the option to upload the file that has been uploaded to Dryad to TreeBASE as well. Checking the checkbox and selecting TreeBASE from the repository dropdown will initiate the file's upload to TreeBASE.

1. Describe your publication
2. Upload and describe your data files
- 3. Approve data for publication**

Press a button to edit the description of your publication, add or modify all attached data files, or finalize the submission.

Describe publication:

Data from: Sample Data Package

Add and describe data files:

sample

Upload data files to partner repositories (optional):

☒ sample.nexus

Finalize submission:

When you have added all data files for this publication, press the button below to finish the submission. A Dryad curator will review the submission to ensure the descriptions are complete and the files are not corrupted. After this brief review, your data will be archived in Dryad.

The author will receive an email when the Dryad submission has been received and another email when the TreeBASE submission has completed. In the second email, there will be a URL that can be visited in order to complete the submission within the TreeBASE system. This will require entering additional information that is not required for submission to Dryad.

Initial submission to TreeBASE

For the case, where data has already been submitted to TreeBASE, there is the option to link that data to Dryad. At the second stage of the submission process, a submitter will see the opportunity to enter an identification number for the data and the name of the repository in which the data has been submitted.

BagIt Handshaking

Publication

Data from: Sample Data Package, Clarke, Kevin S.

Data file *

Please upload your data file or provide the identifier of a file located in another repository.

☒ No file chosen

☐

Data file description

Title*:

Description:

So, if the data has already been uploaded to TreeBASE, instead of uploading the data again, a submitter may just enter the TreeBASE identifier and select TreeBASE as the remote repository from the dropdown menu in the submission form. This will create a link between the Dryad data record and the data stored in the remote repository.

BagIt Handshaking

Publication
Data from: Sample Data Package, Clarke, Kevin S.

Data file *
Please upload your data file or provide the identifier of a file located in another repository.
☐ No file chosen
☒ ▼

Data file description
Title*:

Description:

Technical Documentation

More detail on the TreeBASE/BagIt handshaking can be found on the Dryad [BagIt Handshaking](#) page.

This integration is based on the following technologies:

BagIt -- A lightweight format for packaging digital content and ensuring that it is transferred intact.
OAI-PMH -- A protocol developed by the digital library community to allow harvesting of metadata from remote repositories.

We are evaluating the SWORD protocol to manage the transfer of BagIt packages, but we have not yet determined whether SWORD will be lightweight enough to justify its use.

Design History

For information on design decisions, look at the [BagIt](#) and [OAI-PMH](#) pages listed in the Technical documentation and at the [TreeBASE OAI Provider](#) page on this wiki. We also looked at [ADMIRAL: A data management infrastructure for research across the life sciences](#)

Troubleshooting issues with a Dryad installation.

Contents

- 1 Apache/redirection issues
- 2 Selinux issues
- 3 Errors loading certain pages
- 4 DOI Issues
- 5 Logging issues

Apache/redirection issues

Ensure that the Apache proxy is configured for the correct machine, and that the machine name is correct in the dspace.cfg setting for dspace hostname.

install setroubleshootd...this should send you emails with blocked actions by selinux and how to fix them:

```
yum install setroubleshoot
echo myemail@address.com >> /var/lib/setroubleshoot/email_alert_recipients
```

You can add this to your apache config to log what the rewriterules are doing:

```
RewriteLog "rewrite.log"
RewriteLogLevel 1
```

Selinux issues

If selinux is denying access to something, use the audit2allow command to establish new policies.

Errors loading certain pages

There are many possible causes. Things to check:

- Tomcat version
- Java version AND Java type -- Dryad is known to work with Sun/Oracle Java 1.6. Other versions of Java have been known to cause strange errors in the xmlui.
- Permissions. All files in the install directory should be owned by the dryad user, and Tomcat should run as this user.

DOI Issues

If DOIs are not resolving correctly:

1. Run the DOI synchronization script on the server DOI database:

```
./dspace dsrun org.dspace.doi.DOIDbSync
-s: to synchronize + report
```

BagIt Handshaking

-r: to produce the report

1. Check the definition of the DOIs in the EZID server

Logging issues

The dspace.cfg file specifies which log configuration file is used. Normally, we use the log4j.xml file to configure logging. Note that the beginning section just sets up many appenders, while the lower section actually associates loggers with the appenders.

You may have an instance of Dryad already installed that you would like to be able to copy the data from. There are three sources of data that you'll need to copy: the local DOI database, the DSpace assetstore, and the DSpace PostgreSQL database.

To copy the local DOI database, just copy the file ``/opt/dryad/doi-minter/doi.db`` from one machine to the other.

To copy the PostgreSQL database, do a standard Backup and Restore (it helps if your db and db user are the same in the both instances). Dumping as SQL text is generally the most reliable, especially if the local PostgreSQL installation is a different version from the source of the dump. More detailed instructions:

```
# (on source machine) dump the postgres database to a file
# the --inserts flag makes the restore process much slower, but may be necessary if the
# import command doesn't work with the data exported without it
pg_dump dryad_repo >dryadDBexport.sql
# (on target machine) stop tomcat so nothing interferes with the reload
sudo /etc/init.d/tomcat6 stop
# (target machine) remove old database from target server (if it exists) and rebuild
# may need to restart postgres to disable any connections to it
sudo dropdb -U postgres dryad_repo
sudo -u dryad createdb -U dryad_app -E UNICODE dryad_repo
# (on target machine) load the database into target server
# this step may take an hour or more; do it as a batch process to keep it running
# even if you lose your connection
# you can check on progress by starting psql and seeing which tables have content
at now
at> psql -U postgres dryad_repo <dryadDBexport.sql >import.log
at> (press Ctrl-D)
```

To copy the assetstore, do an rsync from `/opt/dryad/assetstore` on one machine to the same location on the other.

```
# (source machine, assuming dryaddev.lib.ncsu.edu is the target machine)
rsync -azv assetstore dryad@dryaddev.lib.ncsu.edu:/opt/dryad-demo/
# may need to include switches --delete --rsh=ssh --rsync-path=/usr/local/bin/rsync
```

After installing or updating a database, the solr index will need to be (re)built.

```
# ensure Tomcat is running
sudo /etc/init.d/tomcat6 start
# rebuild the solr index
sudo -u dryad ./dspace update-discovery-index
```

BagIt Handshaking

The statistics need to be synced from the source machine

```
# (running on target machine, in /opt/dryad/solr)
rsync -azv --delete --rsh=ssh --rsync-path=/usr/local/bin/rsync dryad@dryad.lib.ncsu.edu:/opt/dryad/s
```

How to update the code that runs a particular Dryad instance.

The Dryad source code is implemented as an overlay to DSpace.

Building a new release involves checking out a particular tagged version from the [Dryad Subversion repository](#). Before a tagged version can be checked out, though, it must first be created by making an svn copy of the trunk directory, as in:

```
svn copy https://dryad.googlecode.com/svn/trunk/dryad https://dryad.googlecode.com/svn/tags/2011-7-29
```

Dryad uses dates to indicate tagged revisions. If multiple tags are created in a single day, a timestamp should be added to the end of the date, such as: 2011-03-25T10.29

Once the tag is applied, go to the appropriate Hudson instance to initiate the build. You will have to have a Hudson account in order to start the build. Once in Hudson, go to the pre-configured build entry (click on the name of the build), and click "Configure" in the left-hand menu. Place the tag for the version you have just tagged in the "Repository URL" box and scroll down to the bottom to hit save. Once you have saved the tag you want to build, click "Build Now" from the options on the upper left side of the page.

Hudson should tell you that the build has succeeded. If it fails, it may just be that Tomcat failed to start. Consult the build log, titled "Console Output", also on the upper left side of the Hudson screen for more details about the build (logs, warnings, exceptions, etc.)

Note: we use branches in Subversion, rather than tags, for the version from which we build to production. In Subversion, a branch is the same thing as a tag so you can adjust the instructions above to use the branch path rather than tag path if you would like to do this. The only reason for doing this is that Eclipse treats branches differently from tags. If you're not using Eclipse, this shouldn't matter. Either works fine.

Created March 14 2011 by Kevin S. Clarke

Contents

- [1 Overview](#)
- [2 Instructions](#)
- [3 Technical Documentation](#)
- [4 Design History](#)

Overview

Dryad assigns [Digital Object Identifiers](#) (DOIs) to each Dryad Data Package and Data File. These persistent identifiers provide an easy way to cite and access data deposited in the Dryad Data Repository. Dryad registers its DOIs and their [associated metadata](#) through the [DataCite](#) service and they are resolvable through the [DOI.org](#) system.

Instructions

On submission to Dryad, authors will receive an email that informs the submitter that a DOI will be assigned on the data package's acceptance into Dryad.

[Template that is used for this email](#)

Once the Dryad curator has reviewed and approved the submission, a second email will be sent. This email includes a DOI that the author can use to cite the data package.

[Template that is used for this email](#)

Dryad's page on [recommended data citation practices](#) details the use of the Dryad assigned DOIs.

Technical Documentation

[DOI Service](#)

[DOI Usage](#) -- specification that defines how DOIs are used in Dryad

[EZID service](#) -- service from the California Digital Library that manages DOI registration with [DataCite](#)

Design History

Our implementation was influenced by the California Digital Library (CDL)'s [NOID](#) project.

Dryad's DOI Services were first developed to work with the [German National Library of Science and Technology](#) (TIB)'s DataCite registration service and then later adopted to use the California Digital Library's [EZID](#) DOI registration service.

Ongoing development will take place in conjunction with DSpace's nascent Identifier Services module.

Contents

[1 Overview](#)

[2 Instructions](#)

[3 Technical](#)

[Documentation](#)

[4 Design History](#)

Overview

Dryad is a member node in the [DataONE](#) network. DataONE is an NSF-funded DataNet, a distributed organization that aims to provide persistent, robust, and secure access to well-described and easily discovered data from the genome to the ecosystem, including Earth observational data from atmospheric, ecological, hydrological, and oceanographic sources.

Instructions

The DataONE MN interface is a programming interface so won't be used unless a scripting language is used to interact with the interface. A simple listing can be seen, though, by putting the following into the browser's location bar:

<http://datadryad.org/mn/object>

with the ability to access a particular file illustrated by the following:

<http://datadryad.org/mn/object/doi:10.5061/dryad.1030/1/dap>

or

<http://datadryad.org/mn/object/doi:10.5061/dryad.1030/1/doc>

Technical Documentation

The full technical documentation for the DataONE MN RESTful programming interface is available on the [DataONE RESTful API](#) page.

Design History

We aim to make this work with DSpace as closely as possible, because it is likely that other institutions will want to become member nodes (e.g., UIUC, MIT)

We started using the Discovery Solr index but moved to creating a Dryad specific one; we may go back to using the Discovery Solr index when it becomes more stable, but the Discovery module is under heavy development at the moment

Contents

- [1 Overview](#)
- [2 Instructions](#)
- [3 Technical Documentation](#)
- [4 Design History](#)

Overview

Dryad harvests the contents of partner repositories in order to provide a one stop entry point to cross-searching relevant materials. Current partner repositories harvested by Dryad include [TreeBASE](#) and [KNB](#). Materials harvested into Dryad are also searched when a search of the Dryad collection is performed. Results from the partner repositories are displayed in search tabs along with the Dryad search results.

Instructions

The integration of partner repository materials is visible when a search of Dryad is performed. The search results

BagIt Handshaking

page will return a list of Dryad results and, in separate tabs, results from the same search performed against the harvested partner repository resources. Before clicking the partner repository tab, a searcher will be able to see how many results are available from the hit count displayed on the search results tab.

Search Results		
Dryad (8)	TreeBASE (5)	KNB (14)
Kisel Y, Barraclough TG (2010) Data from: Speciation has a spatial scale that depends on levels of gene flow. <i>The American Naturalist</i> doi:10.5061/dryad.887		
Mitchell A, Mitter C, Regier JC (2000) Data from: More taxa or more characters revisited: combining data from nuclear protein-encoding genes for phylogenetic analyses of Noctuoidea (Insecta: Lepidoptera). <i>Systematic Biology</i> doi:10.5061/dryad.558		
Kuo MM, Sperling FAH, Caterino MS, Reed RD (2001) Data from: A partitioned likelihood analysis of swallowtail butterfly phylogeny (Lepidoptera: Papilionidae). <i>Systematic Biology</i> doi:10.5061/dryad.615		
Walters JR, Hardcastle TJ (2011) Data from: Getting a full dose? Reconsidering sex chromosome dosage compensation in the silkworm, <i>Bombyx mori</i> . <i>Genome Biology and Evolution</i> doi:10.5061/dryad.8716		
Altermatt F, Pearse IS (2011) Data from: Similarity and specialization of the larval versus adult diet of European butterflies and moths. <i>The American Naturalist</i> doi:10.5061/dryad.cb6pk		
Ohshima I, Yoshizawa K (2010) Data from: Differential introgression causes genealogical discordance in host races of <i>Acrocercops transecta</i> (Insecta: Lepidoptera). <i>Molecular Ecology</i> doi:10.5061/dryad.1390		
Groot A, Classen A, Staudacher H, Schal C, Heckel D (2010) Data from: Phenotypic plasticity in sexual communication signal of a noctuid moth. <i>Journal of Evolutionary Biology</i> doi:10.5061/dryad.1956		
Santos H, Burban C, Rousset J, Rossi J, Branco M, Kerdelhué C (2010) Data from: Incipient allochronic speciation in the pine processionary moth <i>Thaumetopoea pityocampa</i> (Lepidoptera: Notodontidae). <i>Journal of Evolutionary Biology</i> doi:10.5061/dryad.2001		

Technical Documentation

More detail on the OAI-PMH harvesting of KNB/TreeBASE content can be found on the [Harvesting Technology](#) page

Technical details on the way Dryad handles the display of the partner repository content in the search results can be found on the [Tabbed Searching Technology](#) page.

Design History

Mockups from the design phase of the Tabbed Searching implementation [are available](#). The design of Dryad's harvesting capabilities builds on those of [DSpace's](#).

Contents

- [1 Overview](#)
- [2 Workflow](#)
- [3 Configuration](#)
- [4 Testing](#)
- [5 Relation to DSpace](#)
- [6 Email Parsers](#)
- [7 Managing Content in the Review Workflow](#)

Overview

The Dryad journal-submit modules allows the import of metadata that is mailed from integrated journals to Dryad. The email is parsed for article publication metadata and that metadata is used to populate the form that the submitter sees when he or she comes to Dryad and inputs an accepted manuscript number.

Workflow

1. Journal sends email with article metadata to NESCent email and to the article author
2. The email is addressed to journal-submit@datadryad.org. Godaddy is configured to forward this email to journal-submit@nescent.org.
3. journal-submit@nescent.org is configured to route the email to dryad.journal.submit@gmail and to journal-submit-dev@nescent.org.
4. NESCent sends a copy of email to dryad.journal.submit Gmail address and also pipes the content to a `curl` command that POSTs the email content to the journal-submit webapp (this runs on the NESCent mail server). Likewise, the email to journal-submit-dev is piped to a script (journalWebAppDev.sh) that pipes the content to a `curl` command that POSTs the content to the journal-submit webapp that runs on dev.datadryad.org.
5. The journal-submit webapp reads the byte stream with the email content and detects the journal's name
6. The webapp then looks up the journal name in the journal-submit configuration file, DryadJournalSubmission.properties, to learn the `parsingScheme` to use. The value of this parameter is used to match on the parsing classes in the journal-submit's codebase
7. When a particular parser is determined to be the correct one to use, the webapp uses that to parse the remainder of the email, putting metadata values into a ParsingResult object
8. The ParsingResult class is then used to export the metadata into an XML file; the location of this file is determined by the `metadataDir` value from the DryadJournalSubmission.properties configuration file.
9. Once this file is written (using the manuscript number as the name of the file), the journal-submit webapp is done with its process and control moves to the standard DSpace/Dryad submission process, which uses the written XML file to populate the submission form when the author comes to Dryad to complete the submission process.

Configuration

Below is a sample configuration from the journal-submit webapp's configuration file, DryadJournalSubmission.properties. Each journal handled by the submission system needs an entry in this file (even if the journal is not an integrated journal (i.e., doesn't send Dryad the article metadata in an email format)).

```
# all the journals configured in this file (using their parsing scheme code)
journal.order=amNat, BJLS, bmcEvoBio, bmjOpen, ecoApp, ecoMono, ecology, evolution, EvolApp, ecoFront

# American Naturalist
journal.amNat.fullname = The American Naturalist
# directory in which the resulting XML metadata file is stored
journal.amNat.metadataDir = /opt/dryad/submission/journalMetadata/amNat
# the parsing scheme (used to match against parsing class)
journal.amNat.parsingScheme = amNat
# whether we can receive article metadata emails from the journal
journal.amNat.integrated=true
# who to notify when a submission is reviewed
journal.amNat.notifyOnReview=ryantestAmNatReview@scherle.org
```


BagIt Handshaking

```
# who to notify when a submission is archived
journal.amNat.notifyOnArchive=ryantestAmNat@scherle.org
```

The location of the `DryadJournalSubmission.properties` file is also configurable via Maven profiles. The default value should be in `${DRYAD_HOME}/config/DryadJournalSubmission.properties`, but if a different location is desired the following should be changed in the Maven profile used to build the project:

```
# the location of the configuration file for journal-submit webapp
<default.submit.journal.config>/opt/dryad/config/DryadJournalSubmission.properties</default.submit.jo
```

One might want to do this to have a different set of email addresses used for notification purposes so non-project staff don't receive emails from the development instance of Dryad.

Testing

The application ``curl`` can be used to test the `journal-submit` module on a development machine.

```
curl --data-binary @message.test http://localhost:9999/journal-submit
```

Indicate that the data sent should be in binary form with the ``--data-binary`` parameter and pass a reference to a file name using the `@` symbol (`message.test` in the example refers to a file on the local file system -- it should be relative to the place from which the script is run (or be an absolute file system path)). The module will output the XML that it generates or a stacktrace indicating the problem it found parsing the data submitted.

As a place to start debugging... the `journal-submit` webapp was originally written to concatenate textual values into an XML document, rather than using an XML-aware library. Recently an XML library was added to check the concatenated string to make sure it is well-formed XML. If problems appear in the future, this is a good place to start looking into them (since this check imposes restrictions that individual parsers may or may not have handled correctly). This check was added to the [EmailParser](#) class, which is an abstract class all parsers should implement.

Relation to DSpace

The `journal-submit` is a separate webapp (DSpace module) but it is related to the standard [DSpace submission process](#) (which [Dryad has heavily modified](#))

Email Parsers

The following is a list of the current set of email parsers.

`EmailParserForAmNat` - This parser and the one for `ManuscriptCentral` have a similar structure, but different email fields. `AmNat` has a larger set of field tags. The tags map directly to XML element names.

- ◆ Authors' names are: first last, degree/title (e.g. Dr., Prof.). Names are separated by semicolons.
- ◆ Classification terms are: Major: minor. Terms are separated by semicolons.

`EmailParserForBmcEvoBio` - This parser is similar to the `ManuscriptCentral` parser. It differs that it ignores line breaks in the abstract field (line breaks may be used to separate sections in the abstract). It also accepts author lists joined by 'and' and separated by commas, rather than joined by semicolons.

- ◆ Authors are first last. Names are separated by commas, final author joined by 'and' (no comma).

BagIt Handshaking

- ◆ Keywords (Classification terms) are: Major: minor. Terms are separated by line breaks.
- EmailParserForEcoApp - Restructured parser that attempts to better separate the parsing and XML generation stages. Email tags are mapped to java classes (in the xml child package), which are subclasses of the xom Element class. XOM is not used for the final output - after each element is constructed, it is serialized and appended into a field in the ParsingResult returned object.
- ◆ Authors' names are: first last. Names are separated by commas, final author joined by 'and' (with preceding comma).
- ◆ Parsing of Classification terms are currently unknown (do not appear in example messages)
- EmailParserForManuscriptCentral - Similar to AmNat, but a smaller set of tags.
- ◆ Authors' name are: last, first. Names are separated by semicolons
- ◆ Keywords (Classification terms) are comma-separated.

Managing Content in the Review Workflow

The "in review" workflow stage is a holding ground for data submissions associated with manuscripts in review.

For approving/rejecting items which are in the review stage the following command can be used

```
./{dspace.dir}/bin/dspace review-item {-i workflow_id|-m manuscript_number} -a {true|false}
```

This class requires 2 parameters. The first parameter indicates the item, and can take one of two forms:

- i the id of the workflow item (workflow_item_id instead of item_id)
- m the manuscript number associated with the item

The second parameter indicates the status of the item:

- a whether or not the item has been approved

Status: This feature is in development.

Content will be mirrored from the production Dryad system to the instance at the British Library. Initially, the simplest possible implementation will be used:

1. On production, export content to a local directory (as it is created)
2. rsync content to the BL server (cron)
3. On BL server, import content from local directory (cron)

Need to do:

1. Link this page to related pages
2. Complete this as a full "feature" page

Introduction

Dryad users can receive notifications of new content through the normal DSpace mechanisms, or via Twitter.

Twitter Feed

All newly archived items in Dryad are posted to the Twitter feed *datadryadnew*. This feed receives notifications via the service twitterfeed.com. Unfortunately, no one remembers the name of our account at twitterfeed, so it's currently black magic. If it fails, though, we have an alternate account ready to go at twitterfeed, with the username admin@datadryad.org and the same password as the twitter account.

Introduction

The "in review" workflow stage is a holding ground for data submissions associated with manuscripts in review.

Details

For approving/rejecting items which are in the review stage the following command can be used `"./{dSPACE.dir}/bin/dSPACE review-item"`. This class requires 2 parameters. The first parameter indicates the item, and can take one of two forms:

- i the id of the workflow item (workflow_item_id instead of item_id)
- m the manuscript number associated with the item

The second parameter indicates the status of the item:

- a whether or not the item has been approved (true or false should be included behind this parameter)

Contents

- [1 Overview](#)
- [2 Instructions](#)
- [3 Technical](#)
- [Documentation](#)
- [4 Design History](#)

Overview

Dryad enables users to download citations for data packages in the RIS and BibTex formats for importing into their desktop citation management software. Dryad also enables the bookmarking (or "liking") of data packages on a variety of social networking sites (e.g., Facebook, Delicious, CiteULike, Digg, etc.) Both these options take advantage of Dryad assigned DOIs to provide a consistent mechanism for referencing Dryad Data Packages.

Instructions

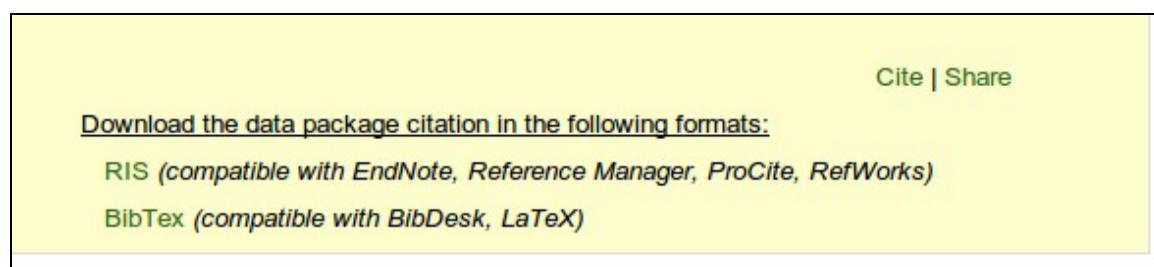
The option to download a citation for a Dryad data package is available underneath the citation display found at the top of every page that displays a Dryad data package. Underneath the citation, on the right side of the page, will be the option to cite or share the Dryad data package information.

BagIt Handshaking



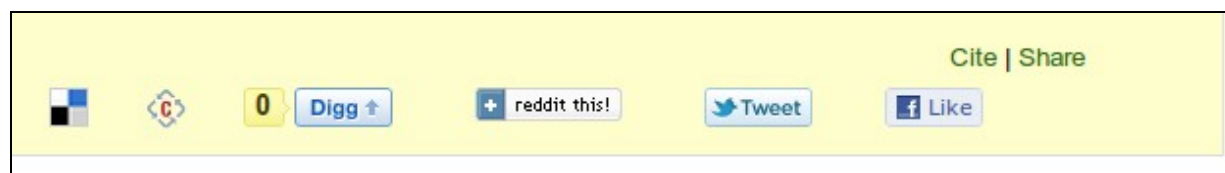
The screenshot shows the Dryad website interface. On the left, there is a navigation menu with links for 'Submit Data Now!', 'My Account' (including 'My Exports' and 'Login or Register'), and 'Browse' (including 'Authors'). The main content area displays the title of a data package: 'Data from: An integrative approach to delimiting species and evolutionarily significant units in a rare but widespread mycoheterotrophic orchid.' Below the title, it provides citation instructions for the original article and the Dryad data package, including the DOI: 10.5061/dryad.8582. A 'Cite | Share' link is visible in the bottom right corner of the main content area.

Clicking on the 'cite' option will display information about how the Dryad data package citation can be downloaded to the desktop. Clicking on 'RIS' or 'BibTex' will download the citation in that format.



This screenshot shows a yellow box with the 'Cite | Share' link in the top right corner. Below the link, it says 'Download the data package citation in the following formats:'. Underneath, there are two options: 'RIS (compatible with EndNote, Reference Manager, ProCite, RefWorks)' and 'BibTex (compatible with BibDesk, LaTeX)'. The 'BibTex' option is highlighted in green.

Alternatively, Dryad data packages can be bookmarked on a variety of social networking sites (Delicious, CiteULike, Facebook, etc.) To do this, click the 'share' link and then select from the displayed social networking sites.



This screenshot shows a yellow box with the 'Cite | Share' link in the top right corner. Below the link, there are several social sharing icons: a small square icon, a CiteULike icon, a Digg icon, a reddit icon, a Twitter icon, and a Facebook Like icon. The 'Share' link is highlighted in green.

Technical Documentation

More information about the technical details behind downloading a data citation and sharing a link to a Dryad Data Package can be found on the [Citation Sharing Technology](#) page.

Design History

Dryad's selection of citation download formats was based on looking at how others in the same space are providing the functionality, in particular the way PLoS ONE [handles it](#). We tested several citation management software packages (EndNote, etc.) to confirm that they could import the format of RIS that we export.

Dryad also used the linking mechanisms suggested by the social networking sites that we support. If a site supplied a !JavaScript to make the link we used that. If a site used an iframe to support the bookmarking activity, we used that.

Contents

- [1 Overview](#)
- [2 Instructions](#)
- [3 Technical Documentation](#)
- [4 Design History](#)

Overview

Dryad provides general statistical information about the number of data files and data packages in the repository as well as information about how many times each data package and data file has been downloaded and viewed. These statistics provide an indication about the amount of use that data packages and file receive and give an overall snapshot of Dryad's growth.

Instructions

To see the overall Dryad statistics, just visit the Dryad home page. On this page, near the top, is a sentence that shows the current date and how many data packages and data files are in Dryad at that point in time. This sentence also gives the number of distinct journals represented in Dryad. For an example, see the screenshot below.



BagIt Handshaking

There are also statistics on the data package and data file level within Dryad. They look the same except on the Dryad data package only the number of *views* are displayed while on the data file *views* and *downloads* for that particular data file are displayed.

The screenshot shows the Dryad website interface. At the top left is the Dryad logo, a green tree with the word 'DRYAD' below it. To the right is a 'Login' button and a 'Search Data' search bar. Below the logo is a red button that says 'Submit Data Now!' with a link 'See how to submit' underneath. On the left side, there are three main sections: 'My Account' with a link 'Login or Register', 'Browse' with links 'Authors' and 'Journal Title', and 'Information' with links 'Depositing Data', 'Using Data', 'Dryad Partners', 'Archiving Policy', and 'About Dryad'. The main content area is titled 'Data from: Towards a worldwide wood economics spectrum'. It contains a citation for the original article: 'Chave J, Coomes D, Jansen S, Lewis SL, Swenson NG, Zanne AE (2009) Towards a worldwide wood economics spectrum. Ecology Letters 12: 351-366. doi:10.1111/j.1461-0248.2009.01285.x'. Below this is a citation for the Dryad data package: 'Zanne AE, Lopez-Gonzalez G, Coomes DA, Illic J, Jansen S, Lewis SL, Miller RB, Swenson NG, Wiemann MC, Chave J (2009) Data from: Towards a worldwide wood economics spectrum. Dryad Digital Repository. doi:10.5061/dryad.234'. To the right of the package citation are links 'Cite | Share'. Below the citations, there is a table with three rows: 'Dryad Package Identifier' with 'doi:10.5061/dryad.234' and '342 views'; 'Individual Data Files' with 'Global Wood Density Database' and '1448 views 685 downloads'; and 'Abstract' with a paragraph of text: 'Wood performs several essential functions in plants, including mechanically supporting aboveground tissue, storing water and other resources, and transporting sap. Woody tissues are likely to face physiological, structural and defensive trade-offs. How a plant optimizes among these competing functions can have major ecological implications,'.

Technical Documentation

The Dryad statistics build on the statistics package provided by DSpace. More information about the implementation can be found on the [Statistics Technology](#) page.

Design History

Dryad's statistics build on DSpace's [Statistics module](#). We also use [Google Analytics](#) so looked at that in the process of adding the Dryad statistics functionality.

Contents

- [1 Overview](#)
- [2 Instructions](#)
- [3 Technical Documentation](#)

4 Design history

Overview

Dryad's submission system allows users to quickly and easily submit data packages. Key features include:

1. Integration with journals -- allowing authors to automatically fill the Dryad forms with information from the journal's description of an article, and automatically notifying journal editors when a Dryad submission is completed.
2. Minimal description requirements -- Dryad minimizes the amount of typing/clicking required by submitters. Descriptions are automatically propagated from the article description to descriptions for individual data files. Reasonable defaults are set for the various choices available.
3. Integration with partner repositories -- allowing submitters to use Dryad information as a starting point for submitting content to another repository.

See also the journal-oriented [Submission Integration](#) page.

Instructions

An overview of the Dryad submission process can be seen in the [Dryad submission video](#).

For a journal-oriented view of the process, see [Overview of the submission workflow](#).

Technical Documentation

Technical documents are available on several topics:

[Submission System Technology](#)
[Journal Metadata Processing](#)
[Integration with TreeBASE](#)

Design history

Mockups of submission features:

[Submission System Mockups](#)
[Login-Register Mockup](#)
[TreeBASE Submission Mockups](#)
[Repository Handshaking Mockup](#)

Back-end submission features:

[Submission System Updates Phase 3](#)

Contents

- 1 Overview
- 2 Instructions
 - ◆ 2.1 Initial submission to Dryad
 - ◆ 2.2 Initial submission to TreeBASE
- 3 Technical Documentation
- 4 Design History

Overview

Authors who submit content to Dryad have the option to forward their Dryad submission to TreeBASE. This saves the time of the author by automating the submission of data to multiple repositories, and create an explicit link between the entries in the two repositories for easier data reuse.

Alternately, authors who initially submit content to TreeBASE may create a link from a Dryad data package to the relevant item in TreeBASE.

Instructions

When a submitter decides to submit data to Dryad, and logs into the Dryad submission process, there are three simple stages to the submission process:

1. Describe the publication
2. Upload and describe the data files
3. Approve data for publication

The TreeBASE submission integration options occur at steps two and three, depending on whether the data is first being submitted to Dryad or has already been submitted to TreeBASE.

Initial submission to Dryad

At the second stage of the submission process, a submitter will see the option to "choose file" from their local machine. This will upload the data into Dryad. When this has happened, the submission form's page will change to indicate the file size of the uploaded data file.

BagIt Handshaking

Publication
Data from: Sample Data Package, Clarke, Kevin S.

Data file *
Please upload your data file or provide the identifier of a file located in another repository.

File	Size	Format	Action
sample.nexus	195Kb	Nexus	<input type="button" value="Remove"/>

Data file description
Title*:

Description:

If a file has been uploaded through the "choose file" interface, at the last stage in the submission process, the author will be given the option to upload the file that has been uploaded to Dryad to TreeBASE as well. Checking the checkbox and selecting TreeBASE from the repository dropdown will initiate the file's upload to TreeBASE.

BagIt Handshaking

1. Describe your publication
2. Upload and describe your data files
- 3. Approve data for publication**

Press a button to edit the description of your publication, add or modify all attached data files, or finalize the submission.

Describe publication:

Data from: Sample Data Package

Add and describe data files:

sample

Upload data files to partner repositories (optional):

☒ sample.nexus ▼

Finalize submission:

When you have added all data files for this publication, press the button below to finish the submission. A Dryad curator will review the submission to ensure the descriptions are complete and the files are not corrupted. After this brief review, your data will be archived in Dryad.

The author will receive an email when the Dryad submission has been received and another email when the TreeBASE submission has completed. In the second email, there will be a URL that can be visited in order to complete the submission within the TreeBASE system. This will require entering additional information that is not required for submission to Dryad.

Initial submission to TreeBASE

For the case, where data has already been submitted to TreeBASE, there is the option to link that data to Dryad. At the second stage of the submission process, a submitter will see the opportunity to enter an identification number for the data and the name of the repository in which the data has been submitted.

BagIt Handshaking

Publication
Data from: Sample Data Package, Clarke, Kevin S.
Data file *
Please upload your data file or provide the identifier of a file located in another repository.
<input checked="" type="radio"/> <input type="button" value="Choose File"/> No file chosen
<input type="radio"/> <input type="text" value="External file identifier"/> <input type="button" value="(please select a repository) ▼"/>
Data file description
Title*:
<input type="text"/>
Description:
<input type="text"/>

So, if the data has already been uploaded to TreeBASE, instead of uploading the data again, a submitter may just enter the TreeBASE identifier and select TreeBASE as the remote repository from the dropdown menu in the submission form. This will create a link between the Dryad data record and the data stored in the remote repository.

BagIt Handshaking

Publication
Data from: Sample Data Package, Clarke, Kevin S.

Data file *
Please upload your data file or provide the identifier of a file located in another repository.
☐ No file chosen
☒ ▼

Data file description
Title*:

Description:

Technical Documentation

More detail on the TreeBASE/BagIt handshaking can be found on the Dryad [BagIt Handshaking](#) page.

This integration is based on the following technologies:

BagIt -- A lightweight format for packaging digital content and ensuring that it is transferred intact.

OAI-PMH -- A protocol developed by the digital library community to allow harvesting of metadata from remote repositories.

We are evaluating the SWORD protocol to manage the transfer of BagIt packages, but we have not yet determined whether SWORD will be lightweight enough to justify its use.

Design History

For information on design decisions, look at the [BagIt](#) and [OAI-PMH](#) pages listed in the Technical documentation and at the [TreeBASE OAI Provider](#) page on this wiki. We also looked at [ADMIRAL: A data management infrastructure for research across the life sciences](#)

Contents

[1 Overview](#)

[2](#)

[Functionality](#)

[3 Workflow](#)

[4](#)

[Configuration](#)

[5 Relation to](#)

[DSpace](#)

Overview

Dryad provides the ability to download a citation for a data package stored in Dryad. It also provides the ability to 'bookmark' a link to a data package on a variety of social networking sites (e.g., Delicious, Facebook, Digg, Mendeley, etc.) Both of these options are available from the item view pages for data files and packages. If the option to download a citation (or bookmark) is made from a data file page, the information for that file's data package will be used as what's cited (or bookmarked) -- instead of the information from the data file.

Functionality

The functionality provided by the citation download is related to Dryad's DOI module. Inside that module is a [CitationServlet](#) Java class that resolves the DOI provided as the authoritative reference for that data package. Once the DOI is resolved to a DSpace record, the metadata for that record is extracted and converted into either the BibTex or RIS format and made available for download from the servlet.

The sharing works in a slightly different way. It uses the metadata from the data package but because we're just building a link to a remote social networking site, we can use the metadata that is already available to the XSLT that renders the item view page (and the links to the remote social networking sites). One thing this depends on is the ability to make the data package's metadata available on the data file's item view page.

The data package's metadata is included in the display for the data file thanks to the [org.datadryad.dspace.xmlui.aspect.browse.ItemViewer](#) class. This is a modification of the ItemViewer class that comes with DSpace. Because we've completely co-opted the ItemViewer, though, to represent the package to file relationship, we've moved this ItemViewer class into the org.datadryad.dspace package and changed the DSpace [Discovery aspect](#) to use this viewer (instead of the standard DSpace ItemViewer).

Question: Why is the ItemViewer in the DSpace Discovery module in the first place? Seems like there would be a better place for it.

Both of these mechanisms rely on the [DryadItemSummary.xsl](#) file to construct the links that provide the functionality and to present the options underneath the "How to cite a Dryad package" box. In the design stage, we decided we didn't want the cite and share options visible at first glance, but hidden in a div that is exposed to the user when s/he clicks on the option to "cite | share" (displayed under the citation display when they are visible).

The JavaScript that hides and displays the divs with the `_cite_` and `_share_` options is the [utils.js](#) file, a JavaScript file that contains some generic utilities used throughout the site.

[COinS](#) is also supported by Dryad. Sites like Mendeley and Zotero can read the COinS data in a page and enable the page to be bookmarked through their (Mendeley's and Zotero's) browser plugins. The COinS information is put into the HTML page via the Dryad theme's XSLTs, specifically the [DryadUtils.xsl](#) file, which provides some general templates for use by the other XSLTs.

Workflow

Sharing*

1. User visits item page (package or file)
2. XSLT pulls metadata from the item record or the page metadata (e.g., in the case of the data file, the ItemViewer has put Dryad data package metadata into the pageMeta element)
3. XSLT constructs a link from DOI, title, etc. and uses the remote site supplied iframe, HTML, or JavaScript to pass this information on through the link
4. XSLT displays this option in a hidden div whose content can be viewed by the user by clicking on the option to view it (under the citation information at the top of the page)
5. When a user clicks this option, the hidden div is made visible by the utils.js !JavaScript
6. The user clicks the link and is taken to the remote site with all or some of the form information supplied by the metadata values in the link they followed

Citation*

1. User visits item page (package or file)
2. XSLT presents an option to cite, which is hidden by default, but displayed by the utils.js !JavaScript when the user clicks the "cite" link
3. An option for RIS or !BibTex is presented to the user in the form of a parameterized link
4. The user clicks on a link and is sent to the CitationServlet with information about which format they want
5. The CitationServlet uses the DOI that is sent and resolves that to a DSpace item record
6. The servlet extracts the metadata from the DSpace item record
7. The servlet formats this metadata as either RIS or !BibTex, depending on which parameter it receives, and sends the data citation back to the browser as a data stream
8. The user is given the option to download the data to their file system where their citation management software can then import it

Configuration

The 'configuration' of both these options occurs in the code itself: in the share option, the methods of linking out to the remote social networking sites (as well as which sites are chosen) is done in the XSLT. We basically just use the code (either plain link, iframe, or JavaScript, that the remote site provides for use). Which ones are displayed depends on which ones are included in the ItemSummaryView XSL.

The configuration of which citation formats are supported (currently BibTex and RIS) is built into the CitationServlet itself. There are separate methods for rendering the item metadata into the official citation format. More formats could be supported by adding additional methods, though we've determined the RIS format covers most of the citation management software packages (Endnote, etc.)

Relation to DSpace

This functionality is related to the DSpace Discovery ItemViewer in that it is this class that is overlaid to provide the additional functionality required by Dryad. There are others in the DSpace community taking a different approach to providing this functionality. There are some social networking sites that will provide a drop-in mechanism for bookmarking on other sites. Some are using this and this might be something Dryad wants to

look at in the future. One downside to the current method is that some of the sites require resolution of additional content (through JavaScript or iframes). This causes a delay in the Dryad page loading while these remote resources are resolved, though the main content of the page displays almost instantly.

Contents

- [1 Overview](#)
- [2 Web-accessible features](#)
- [3 Command-line features](#)
- [4 Workflow](#)
 - ◆ [4.1 For Submissions](#)
 - ◆ [4.2 For Citation Downloads](#)
 - ◆ [4.3 For Identifier Services](#)
- [5 Configuration](#)
- [6 Implementation](#)
- [7 Relation to DSpace](#)

Overview

Dryad mints, manages, and registers [Digital Object Identifiers](#) (DOIs) for data packages and data files deposited into the Dryad Data Repository. This page documents the technical details of these DOI services.

General information about Dryad's DOI services can be found on the [Project Documentation wiki](#).

Web-accessible features

Return a minted a DOI without registering it:

```
_http://dev.datadryad.org/doi?item=10255/dryad.1922_  
* Returns DOI string in the form: doi:10.5061/dryad.1922  
* Exception for item not found: HTTP Status 404 - Item does not exist  
Note: the requested item must already exist in DSpace or a 404 is returned
```

Return a minted DOI after registering it:

```
_http://dev.datadryad.org/doi?item=10255/dryad.1922&register_  
* Returns DOI string in the form: doi:10.5061/dryad.1922  
* Exception for item not found: HTTP Status 404 - Item does not exist  
* Exception for registration problem: HTTP Status 500 - !DataCite !ServiceException  
Note: the requested item must already exist in DSpace or a 404 is returned
```

Look up a DSpace Item from a DOI:

BagIt Handshaking

```
_http://dev.datadryad.org/doi?lookup=doi:10.5061/dryad.1922_  
* Returns a DSpace pseudo-Handle in the form: 10255/dryad.1922  
* Exception for DOI not found: HTTP Status 500 ? Unknown DOI
```

Redirect to a DSpace Item page from a DOI:

```
_http://dev.datadryad.org/doi?redirect=doi:10.5061/dryad.1922_  
* Redirects the browser to the target associated with the DOI  
* Exception if supplied DOI is not found: HTTP Status 404
```

The first two functions can be called repeatedly and the same DOI will be returned (it returns a previously minted DOI if it already exists).

Command-line features

The DOI service can be managed using a command line call:

```
dryad/bin/dspace doi-util  
-h                Help... prints this usage information  
-s                Search for a known DOI and return it  
-m [DOI] [URL]    Mints a new DOI and places it in the local database  
-r [DOI] [URL]    Registers a DOI, minting if necessary  
-p <FILE>         Prints the DOI database to an output stream  
-c                Outputs the number of DOIs in the database
```

Database synchronization tool:

```
./dspace dsrun org.dspace.doi.DOIDbSync  
-s: to synchronize + report  
-r: to produce the report
```

Workflow

For Submissions

1. DOIs are minted at the point of submission to Dryad. When a data package is submitted, a call to mint a DOI (without registering it) is made to the DOI Service.
2. The data package should contain data files -- for a DOI to be registered for the data file, there must be a link in the metadata from the data file to the data package.
3. The data package then goes on to be curated by the Dryad Librarian.
4. If the data package is approved, the DOI is registered with !DataCite through the EZID DOI registration service.
5. If the package isn't approved, the DOI remains unregistered (and, currently, is not recycled for use with another package (change this?))
6. Lastly, the registered DOI is emailed to the submitter so that it can be included in the article and used to reference the published data package.

For Citation Downloads

1. DOIs are passed to the !CitationServlet when a user requests a citation download or uses one of the sharing services that Dryad supports (Delicious, Digg, etc.)

BagIt Handshaking

2. DOI Services resolve the DOI and extract the metadata from the record, making it available to be downloaded in RIS or !BibTex format.
3. The !CitationServlet uses Dryad's DOI Services currently, but might in the future use DSpace's `_Identifier Services_` if it becomes an official module and Dryad's DOI resolution is natively built into it.

For Identifier Services

1. Dryad's DOI Services are also used by the `_Identifier Services_` DSpace module. Dryad's DOI Services serve as the local DOI resolver for these DSpace services.
2. In the future, we may better integrate our DOI Services into this module.

Configuration

Configuration of the DOI Services module, requires additional parameters be set in the `dspace.cfg` configuration file. The Dryad project places these parameters in a Maven profile; they are then pulled into the `dspace.cfg` file when Dryad is built.

In the `dspace.cfg` file, the following parameters are used to configure the DOI services:

```
# URL that resolves DOIs
doi.hostname = [http://dx.doi.org http://dx.doi.org]
# Base URL of Dryad used in registering DOIs
dryad.url = [http://datadryad.org http://datadryad.org]
# DOI prefix associated with Dryad
doi.prefix = ${default.doi.prefix}
# Directory where DOI minter files should be stored
doi.dir = ${dspace.dir}/doi-minter
# File system location of the DOI database
doi.db.fspath = ${doi.dir}/doi.db
# Username and password of the CDL !DataCite Web service
doi.username = ${default.doi.username}
doi.password = ${default.doi.password}
# How long (# of chars) the DOI suffixes should be
doi.suffix.length = 5
# Local, static part of the suffix of the generated ID
doi.localpart.suffix = dryad.
# Whether the registration service should be used
doi.datacite.connected = ${default.doi.datacite.connected}
# URL for the DOI Services Web endpoint
doi.service.url=${default.doi.service}
# Indicates test mode for the Identifier Services connection to DOI Services
doi.service.testmode=false
```

These settings, put into your Maven profile (in most cases, the `settings.xml` file), are pulled into the `dspace.cfg` file when Dryad is built:

```
<!-- The real username and password of DOI registration service -->
<default.doi.username>USERNAME</default.doi.username>
<default.doi.password>PASSWORD</default.doi.password>
<!-- The DOI prefix for DOIs minted; for Dryad this is the value below -->
<default.doi.prefix>10.5061</default.doi.prefix>
<!-- Whether to rewrite URLs to use the local DOI resolver or the dx.doi.org one -->
<default.dryad.localize>true</default.dryad.localize>
<!-- Whether to register the DOIs minted or just pretend like you did -->
```

BagIt Handshaking

```
<default.doi.datacite.connected>false</default.doi.datacite.connected>
<!-- The actual endpoint of the DOI service -->
<default.doi.service>http://localhost:9999/doi</default.doi.service>
<!-- An index used for DataONE that works with the DOI registration process -->
<default.solr.dryad.server>http://localhost:9999/solr/dryad</default.solr.dryad.server>
```

Implementation

The core code is in `dryad/dspace/modules/doi`

Relation to DSpace

The Dryad DOI Services modules relates to the Identifier Services module being developed for the DSpace community by @tmire.

Currently, the Dryad DOI services modules exist as a separate DSpace module, but in the future some of these services might be integrated into the Identifier Services module.

A related package in DSpace is the Handle server. Dryad's DOI Services replace our use of the DSpace Handle server, though the Handle server continues to serve links published before we moved to DOIs.

The DSpace PersistentIdentifiers proposal is also relevant.

Contents

1 Data Access

- ◆ 1.1 Web Browser User Interface
- ◆ 1.2 Programmatic Data Access
 - ◇ 1.2.1 Sitemaps
 - ◇ 1.2.2 OAI-PMH
 - 1.2.2.1 Using resumptionTokens with OAI-PMH
 - 1.2.2.2 Programmatic access to individual data files using OAI-PMH
 - ◇ 1.2.3 DataONE API
 - 1.2.3.1 Programmatic access to data files using the DataONE API
- ◆ 1.3 Links to Data Packages/Files
- ◆ 1.4 RSS Feeds

2 SOLR search access

3 Other access mechanisms

4 Suggest Alternatives

Data Access

NOTICE: Dryad is in the process of phasing out the older Handle-style identifiers (those that contain "10255"). We will only continue to support Handle identifiers for navigation to a data package page, retaining the functionality of existing citations in the scientific literature. All other types of data access will be updated to use DOIs. As a result, many of the access mechanisms on this page will be

changed during 2011/2012.

Web Browser User Interface

Primary access to Dryad is through its web interface, where users most commonly search on authors, titles, subjects and other metadata elements. Data files archived by Dryad may be downloaded one-by-one from their Dryad data package Web pages.

Additionally, DSpace, the platform on which Dryad is built, supports several "hidden" ways to hack the system's URLs to get useful metadata from the Web interface.

Viewing full metadata: add "?show=full" to the end of the URL

- ◆ <http://datadryad.org/handle/10255/dryad.12?show=full>

Viewing the raw DSpace representation of a page add "DRI" to the URL

- ◆ <http://datadryad.org/DRI/handle/10255/dryad.12>
- ◆ NOTE: With the December 2011 release, this URL will be changed to make use of the DOI: <http://datadryad.org/resource/doi:10.5061/dryad.12/DRI>

Another way to view the raw DSpace markup is to add "?XML" to the end of the URL. This is less useful than the above method, though, because the page's content won't contain the externalized i18n strings.

- ◆ <http://datadryad.org/handle/10255/dryad.12?XML>

Viewing metadata in machine-readable (METS) format:

- ◆ <http://datadryad.org/metadata/handle/10255/dryad.1152/mets.xml>

Programmatic Data Access

In addition to the web interface, Dryad can be accessed programmatically through a [sitemap](#) or [OAI-PMH](#) interface.

Sitemaps

warning: The sitemaps are not working as of 2011-10-13. Dryad staff are investigating.

The [Dryad sitemap](#) provides access to the links to all Dryad's data package and file pages, with the timestamp of their last update. It is an XML formatted file that is [gzipped](#) for transmission. An example snippet of the XML follows:

```
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">

  <url>

    <loc>http://datadryad.org/handle/10255/dryad.721</loc>
    <lastmod>2010-02-26T11:17:10Z</lastmod>
  </url>
</urlset>
```

OAI-PMH

OAI-PMH is a harvesting protocol that may be used to access Dryad's metadata. The [specification](#) is available, as are [online tutorials](#), but we include a couple of examples of its use here for illustrative purposes.

Identify

Used to learn about the service

· <http://www.datadryad.org/oai/request?verb=Identify>

ListSets

Used to learn what sets of metadata. Dryad offers a data package set and a data file set.

· <http://www.datadryad.org/oai/request?verb=ListSets>

ListMetadataFormats

Used to learn what metadata formats can be returned by the service. Dryad currently offers [METS/MODS](#), [OAI-DC](#) (Dublin Core), [OAI-ORE/Atom](#), and [RDF/DC](#). The amount of information mapped into each format varies. For now, we recommend using the OAI-DC metadata format.

· <http://www.datadryad.org/oai/request?verb=ListMetadataFormats>

ListIdentifiers

Used to list Dryad's OAI identifiers. It requires `from` and `metadataPrefix` parameters to know what range of identifiers to return and what format the metadata should be in (from the options returned by the `ListMetadataFormats` verb). We may modify this to return DOIs in the future.

· http://www.datadryad.org/oai/request?verb=ListIdentifiers&from=2010-01-01&metadataPrefix=oai_dc

NOTE: It is highly recommended that you use this call in conjunction with the "set" parameter, so you retrieve the records of interest. Otherwise, you may retrieve records that Dryad has harvested from other providers

ListRecords

Used to list Dryad records. It requires `from` and `metadataPrefix` parameters so it knows the range of records to return. The records will be returned in the format associated with the `metadataPrefix` requested. Available formats can be discovered by using the `ListMetadataFormats` verb.

· http://www.datadryad.org/oai/request?verb=ListRecords&from=2010-01-01&metadataPrefix=oai_dc

NOTE: It is highly recommended that you use this call in conjunction with the "set" parameter, so you retrieve the records of interest. Otherwise, you may retrieve records that Dryad has harvested from other providers

GetRecord

Used to return a single record. It requires the OAI identifier of the record (the `identifier` parameter) and the format in which the record should be returned (the `metadataPrefix` parameter).

· <http://www.datadryad.org/oai/request?verb=GetRecord&identifier=oai:datadryad.org:10255/dryad.12>

Using resumptionTokens with OAI-PMH

OAI-PMH requests may result in partial results lists being returned. In these cases, the results list will contain a `resumptionToken` that can be used to retrieve the next page of results.

For example, for a call like:

BagIt Handshaking

http://www.datadryad.org/oai/request?verb=ListRecords&from=2010-01-01&metadataPrefix=oai_dc&set=hdl_10255_3

You will receive the first 100 records, ending with a resumptionToken of 2010-01-01T00:00:00Z/9999-12-31T23:59:59Z/hdl_10255_3/oai_dc/100

You can then retrieve the next 100 records with:

http://www.datadryad.org/oai/request?verb=ListRecords&resumptionToken=2010-01-01T00:00:00Z/9999-12-31T23:59:59Z/hdl_10255_3/oai_dc/100

Note that when using a resumptionToken, OAI expects you to only repeat the verb, not any of the other parameters that were part of the original request.

Programmatic access to individual data files using OAI-PMH

The process for a machine to locate and download a file from Dryad takes a few steps. We're working to streamline it, and to make it more standards (in particular OAI-ORE and RDF/Linked Data) compliant,

1. Obtain a Dryad Identifier in "short" form (e.g., 10255/dryad.1234)
 - ◆ To list all identifiers of data files, use the OAI-PMH interface and restrict the "set" to the Dryad data files, like:
 - ◇ http://www.datadryad.org/oai/request?verb=ListIdentifiers&from=2010-01-01&metadataPrefix=oai_dc&set=hdl_10255_3
 - ◆ For recent additions only, use the RSS feed for data files (described above)
2. Obtain the METS metadata
 - ◆ http://datadryad.org/metadata/handle/INSERT_SHORT_ID_HERE/mets.xml
 - ◆ This also applies to data packages, if the identifier you have is for a data package. In the METS metadata for a data package, elements `<dim:field>` with attributes `element="relation"` `qualifier="haspart"` `mdschema="dc"` will have the data file identifier (as a DOI) as value of the element. Remove the "doi:" to obtain the "short" form of the Dryad identifier.
3. (temporary step, while Dryad metadata is in transition) You will need to transform the DOI to a Handle-style identifier. Use the lookup service at http://datadryad.org/doi?lookup=INSERT_DOI_HERE
4. Then obtain the METS metadata for the data file as above.
5. Parse the METS metadata to locate the bitstream URL.
 - ◆ It is in the `<mets:FLocat/>` element in the `xlink:href` attribute. It will look like `/bitstream/handle/SHORT_ID/FILE_NAME?sequence=1` (the sequence number may vary)
 - ◆ If you are interested only in files of a particular type, look for the `<mets:file/>` element and check the value of its attribute `MIMETYPE`. For example, for MS Excel files the value should be `"application/vnd.ms-excel"`.
6. Prepend <http://datadryad.org> and download the file using the bitstream URL. For example,
 - ◆ <http://datadryad.org/bitstream/handle/10255/dryad.633/ApineCYTB.nexus?sequence=1>

DataONE API

WARNING: The DataONE API is not yet finalized. The format for calls to this API may change in the near future.

As part of Dryad's participation in the [DataONE](#) project, Dryad makes content available through a specialized API.

BagIt Handshaking

[List of objects](#)

[Sample metadata call](#)

[Sample file download](#)

[Technical documentation for the DataONE API](#)

Programmatic access to data files using the DataONE API

1. Obtain the DataONE ID of a Dryad object using the DataONE listObjects call:
<http://www.datadryad.org/mn/object> (e.g., dryad.1850/1/nex)
2. Retrieve the file: <http://www.datadryad.org/mn/object/doi:10.5061/dryad.1850/1/nex>

Objects with a DataONE ID ending in "/dap" are metadata objects, which adhere to the Dryad Application Profile (DAP). If you have the identifier of a DAP metadata object and want to know which file object it describes, you can retrieve its DataONE metadata, using a URL such as:
<http://www.datadryad.org/mn/meta/doi:10.5061/dryad.1850/1/dap>.

Links to Data Packages/Files

Dryad uses DOIs ([Digital Object Identifiers](#)) to identify Dryad data packages and files. A few simple examples follow. These may be resolved against the DOI resolver at <http://dx.doi.org> (when you do, remove the "doi:" prefix).

Data packages

- ◆ [doi:10.5061/dryad.1664](http://dx.doi.org/10.5061/dryad.1664)
- ◆ [doi:10.5061/dryad.642](http://dx.doi.org/10.5061/dryad.642)
- ◆ [doi:10.5061/dryad.1307](http://dx.doi.org/10.5061/dryad.1307)

Data files

- ◆ [doi:10.5061/dryad.1664/1](http://dx.doi.org/10.5061/dryad.1664/1)
- ◆ [doi:10.5061/dryad.642/1](http://dx.doi.org/10.5061/dryad.642/1)
- ◆ [doi:10.5061/dryad.1307/1](http://dx.doi.org/10.5061/dryad.1307/1)
- ◆ [doi:10.5061/dryad.1307/2](http://dx.doi.org/10.5061/dryad.1307/2)
- ◆ [doi:10.5061/dryad.1307/3](http://dx.doi.org/10.5061/dryad.1307/3)

RSS Feeds

There are a couple of feed options. Feeds are used by some browsers and all feed and news readers. They may also be used for programmatic access.

Everything -- data packages, data files, and metadata harvested from partner repositories

- ◆ http://datadryad.org/feed/rss_2.0/site

Data packages only

- ◆ http://datadryad.org/feed/rss_2.0/10255/3

Data files only

- ◆ http://datadryad.org/feed/rss_2.0/10255/2

SOLR search access

Dryad content can be searched using a [SOLR](#) interface.

Basic query: <http://datadryad.org/solr/search/select/?q=Galliard>

Field-specific query: <http://datadryad.org/solr/search/select/?q=dwc.ScientificName:drosophila>

Query that searches all text for a string, but limits results to two specified fields:

<http://datadryad.org/solr/search/select/?q=Galliard&fl=dc.title,dc.contributor.author>

Query that looks up Dryad data based on an article DOI:

<http://datadryad.org/solr/search/select/?q=dc.relation.isreferencedby:10.1038/nature04863&fl=dc.identifier,dc.title,dc.description>

Other access mechanisms

If you know of other community-developed services that can search or retrieve content that are not listed here, please alert us at help@datadryad.org

ROpenSci Dryad package for search and retrieval of Dryad data and metadata within R. [Tutorial](#)

Suggest Alternatives

We're interested in hearing what other forms of access people would like. If you have a suggestion for making Dryad's content more accessible, please let us know at help@datadryad.org.

Dryad is a member node in the [DataONE](#) network. DataONE is an NSF-funded !DataNet, a distributed organization that aims to provide persistent, robust, and secure access to well-described and easily discovered data from from the genome to the ecosystem, including Earth observational data from atmospheric, ecological, hydrological, and oceanographic sources.

Contents

1 Usage

- ◆ [1.1 Current Usage](#)
- ◆ [1.2 Identifiers and Versioning](#)

2 Workflow

- ◆ [2.1 Open Issues](#)

3 Configuration

- ◆ [3.1 DSpace-native Member Node API Implementation](#)

4 Relation to DSpace

- ◆ [4.1 TODO: Releasing Dryad's MN Implementation to General DSpace](#)

5 Relation to DataONE

- ◆ [5.1 DataONE member node architecture documents](#)
- ◆ [5.2 \(Deprecated\) Python Prototype Implementation](#)

Usage

The standard URL prefix for interacting with Dryad DataONE interface is "mn" (e.g., <http://datadryad.org/mn/>)

Current Usage

The table below includes the current usage and implementation status.

Method	REST Interface	Status
listObjects()	GET /object	Basic list works. objectFormat needs to follow the correct output type. Needs to respond to all possible parameters. Only supports the (required) XML response format.
get()	GET /object/ <i>ID</i>	Implemented but relationships between data files and data packages need to be improved.
describe()	HEAD /object/ <i>ID</i>	Implemented
create()	POST /object	
create()	PUT /object/ <i>ID</i>	
delete()	DELETE /object/ <i>ID</i>	
getSystemMetadata()	GET /meta/ <i>ID</i>	Implemented
	HEAD /meta/ <i>ID</i>	
getChecksum()	GET /checksum/ <i>ID</i>	Implemented
isAuthorized()	GET /isAuthorized/ <i>ID</i>	
getLogRecords()	GET /log	
listEvents()	GET /monitor/event	
ping()		
getObjectStatistics()		
getStatus()		
login()		
logout()		

Identifiers and Versioning

Dryad needs to keep a timestamp as part of the D1 identifier for metadata, since the Dryad metadata can change without changing the Dryad version number.

Since Dryad doesn't natively keep separate identifiers for data and metadata, the identifiers given to D1 will need to include something to distinguish them (e.g., "/metadata").

See the [DOI Usage](#) page for information on Dryad's native identifier handling.

When Dryad replicates data from other systems, it will store data and metadata in separate objects, using Dryad

BagIt Handshaking

metadata for Dryad's internal tracking purposes.

Sample identifiers:

Currently used by prototype (metadata): hdl:10255/dryad.105/mets.xml

Currently used by prototype (data, just getting first file): hdl:10255/dryad.105/mets.xml_data

Metadata: doi:10.5061/dryad.104/1/dap

Data: doi:10.5061/dryad.104/1/txt

Workflow

The Member Node module is built and deployed the same way as the regular DSpace implementation. Its code lives in (trunk)/dspace/modules/dataone-mn.

Open Issues

1. DOI-like identifiers that aren't registered DOIs (since they have the format on the end)
 1. Move the format to the beginning? txt/doi%3A10.5061/dryad.104/1. This may be a problem if the format becomes complex.
 2. Use a code like "obj" or "meta" at the beginning? What to do with other bitstreams?
 3. Actually register these as DOIs? Introduces a maintenance burden.
 4. Change the prefix from "doi" to "Dryad"? (Ryan likes this) Downsides?
2. system metadata -- rightsholder. what are other people doing about this?
3. The "txt" in "Data: doi%3A10.5061/dryad.104/1/txt" isn't great. It was intended to correspond to the objectFormat in the listObjects. Types.ObjectFormat is not yet fully defined, but it looks like it will be a MIME type or similar long string.
4. Why text/xml instead of application/xml in meta-guid -- understand the reasons for that for XML that is going into a browser but would assume XML from /meta will be application parsed/processed.
5. We're currently providing only data files (not packages), until DataONE decides on the packaging framework. When this happens, we will need to specify the relationships between package objects and file objects. This is different than within Dryad, because the D1 identifiers are slightly different.

For discussion in larger group:

1. Checklist for API compliance? (what methods are included in Level 1, Level 2, etc.)
 1. What do the coordinating nodes currently expect?
 2. Is there a test suite to validate the functionality of a MN API?
2. What is the process for adding a new type of science metadata to the system?
3. /log returns something from the python nodes, but this format hasn't been defined
4. Notifications when the API changes?
5. Need to specify the search API -- possible options on the /object command
6. For create() and update(), we typically do not assign an ID until curator approval -- what should we return???

Configuration

DSpace-native Member Node API Implementation

It is built and deployed the same way as the regular DSpace implementation (see [How To Install Dryad](#)).

An example get() call: <https://datadryad.org/mn/object/doi%3A10.5061/dryad.20/1/dap>

Relation to DSpace

We aim to make this work with DSpace as closely as possible, because it is likely that other institutions will want to become member nodes (e.g., UIUC, MIT)

TODO: Releasing Dryad's MN Implementation to General DSpace

1. Move settings to config file
2. Genericize the metadata output
3. Ensure packaging information is not tightly tied into the MN code

Relation to DataONE

DataONE member node architecture documents

[Member Node API](#)

[Member Node REST interface](#)

(Deprecated) Python Prototype Implementation

Roger Dahl created an initial implementation based on his generic Python member node, using metadata harvested from Dryad. It does not have current data, but it still serves as a reference implementation for many of the API methods.

An example get() call on the old prototype system:

http://129.24.0.11/mn/object/hdl%3A10255/dryad.105/mets.xml_data

Dryad technical documentation is primarily composed of documents from the following categories:

[Category:Help](#) -- General-purpose

[Category:Features](#) -- Documentation of user-facing features

[Category:Technical Documentation](#) -- Documentation of system internals

The documentation is compiled into a manual using [Special:Pdfprint](#).

(Ryan is working out some issues here. Once they are somewhat coherent, they should be moved to the DSpace wiki.)

Understanding various issues in Manakin's data flow.

Passing parameters

Parameters can be passed between Manakin pages in multiple ways.

In the Administrative aspect, a call to `/admin/item?itemID=123`:

1. the sitemap passes to a javascript function (`startEditItem` in `administrative.js`)
2. the javascript function does a `cocoon.request.get("itemID")`; to get the parameter value, does the edit function separately, and finally redirects to the normal item view page.

In the ArtifactBrowser aspect, a call to `/handle/123/4567`:

1. the sitemap calls various matchers (`HandleAuthorizedMatcher`, `HandleTypeMatcher`) and eventually calls `ItemViewer`
2. each of these calls calls `HandleUtil.obtainHandle`, which
 - ◆ the first time it is called for a given `Request`, calls `request.getSitemapURI()`; and parses the URI
 - ◆ on subsequent calls, it retrieves the existing handle from the `Request`

In the ArtifactBrowser aspect, a call to `/feedback`:

1. the sitemap explicitly sends parameters to the `FeedbackForm` (but it is unclear where these values come from)
2. these parameters are then available in the `parameters` field of `AbstractDSpaceTransformer` (parent of `FeedbackForm`)

Note: sometimes, an `objectModel` is passed around, and sometimes it is inherited from `AbstractDSpaceTransformer`. Why?

Contents

- 1 Overview
- 2
- Functionality
- 3 Workflow
- 4
- Configuration
- 5
- Troubleshooting
- 6 Relation to
- DSpace

Overview

The handle server is a TCP service that assists in resolving Handles. It is not needed for basic DSpace functionality, but it is used whenever an end-user dereferences a handle-format URL. These URLs are printed in older articles that have associated Dryad data (articles published before the introduction of DOIs). They are also available from various parts of Dryad, where the interface has not been completely updated to use DOIs.

Functionality

When a user resolves a handle, by putting a handle-format URL in their browser, they are redirected to the appropriate item page in Dryad. Everything else should be invisible to an end-user.

When the central CNRI server receives a request for handle resolution, it first queries the Dryad handle server. The Dryad handle server checks to see whether an appropriate entry exists in the Dryad database. If so, it returns the appropriate redirect URL, and the CNRI server passes this URL on to the user's browser.

Workflow

Startup:

1. Administrator runs `/opt/dryad/bin/start-handle-server`
2. Handle server reads some configuration from `config.dct` (port numbers and other handle settings) and other configuration from `dspace.cfg` (database connection information and logging information)
3. Handle server uses some services from `dspace.jar` to initiate the database connection.
4. Handle server runs as a daemon connected to a particular TCP port, and waits for requests.

For each request:

1. User sends GET request to the CNRI servers (e.g., <http://hdl.handle.net/10255/dryad.20>)
2. CNRI looks in their local configuration (from the uploaded `sitebndl.zip`) to determine the appropriate IP address of the local handle server.
3. CNRI sends a TCP request to the local handle server.
4. The local daemon receives the request, and performs a search in the DSpace database for the appropriate handle.
5. If the handle exists in the database, the daemon returns the appropriate redirection URL (e.g., <http://datadryad.org/handle/10255/dryad.20>)
 1. This URL is built using the server base URL set in `dspace.cfg` (???does the handle prefix come from `dspace.cfg` or `config.dct`???)
 2. CNRI simply forwards the URL to the user's browser for redirection.
 3. The user's browser loads the resultant page.
6. If the handle does not exist in the database, the daemon returns a "not found" indicator, and CNRI generates a "handle not found" page.

Configuration

Installing the handle server for the first time (or rebuilding it on a new machine):

1. Run `/opt/dryad/bin/dspace make-handle-config /opt/dryad/handle-server`
2. Answer the questions
 - ◆ It is best to provide generic contact information, such as `help@datadryad.org`
 - ◆ Take the defaults whenever possible.
 - ◆ Note that the http port is the port for the handle server -- it shouldn't be the same as a port you are already using for DSpace.
3. Take the `sitebndl.zip` file out of the new `handleInfo` directory and email it to `hdladmin@cnri.reston.va.us` with a request to update the IP address. NOTE: The DSpace manual specifies uploading this file to a

BagIt Handshaking

particular webpage. This should only be done when a new handle prefix is being requested. We do not expect that Dryad will ever use a new handle prefix.

4. Edit handle-server/config.dct:

- ◆ Replace YOUR_NAMING_AUTHORITY with the actual handle prefix (10255)
- ◆ Verify that these lines exist in the server_config section (or add them):
 - ◇ "case_sensitive" = "yes"
 - ◇ "storage_type" = "CUSTOM"
 - ◇ "storage_class" = "org.dspace.handle.HandlePlugin"

5. Run `sudo /opt/dryad/bin/start-handle-server`

6. Check for any errors in the log files in the handle-server directory

7. Ensure the handle server can get through the firewall on the ports you selected earlier.

8. Ensure that the handle server will start when the machine is rebooted.

This email thread has some useful info:

<http://dspace.2283337.n4.nabble.com/handle-server-location-td3293457.html>

Troubleshooting

See if there are error messages in any of the log files. Logs can be found in both `/opt/dryad/logs` and `/opt/dryad/handle-server`

- ◆ Ensure that requests are resulting in log entries. If not, the handle server may not be running, CNRI may be using the wrong IP address to contact the handle server, or the handle server's ports may be closed.

Ensure that the ports specified in config.dct are open to the world.

Ensure that the settings listed above for config.dct are in place, particularly the case-sensitivity.

Relation to DSpace

The handle server code used by Dryad is exactly the same as used by any standard DSpace installation. Only some of the configuration is different. In particular:

Dryad uses the handle prefix 10255.

Dryad identifiers include alphabetic characters, so the handle server **must** be configured to be case-sensitive.

Pathnames in the commands above are specific to the standard way that Dryad is installed.

Dryad harvests remote partner repositories in order to provide a search across all their contents. In some cases, Dryad has partnered with the remote repositories through a handshaking process (e.g., in the case of TreeBASE) and in other cases it hasn't (e.g., in the case of KNB). In either case, Dryad users will have their Dryad searches also performed against the contents of the remote repositories, and the search results displayed in the TabbedSearching system on the search results page.

Contents

1

Functionality

2 Workflow

3

Configuration

4 Relation to

DSpace

Functionality

Dryad puts content it has harvested from partner repositories in a separate collection (one for each partner repository). It then treats that collection differently from the collections that are a part of Dryad itself (DryadLab, Data Files, Data Packages, etc.) These harvested collections are treated differently from other collections in that their contents are searched and displayed in a separate tab on the search results page.

OAI-PMH is the harvesting protocol used to harvest these partner repositories. The specification is available online, as are simple tutorials. For quick samples of how to use OAI-PMH, see the [Data Access](#) instructions.

Workflow

The workflow for harvesting involves using the standard DSpace mechanism to setup a collection with a harvested source. As explained above, Dryad uses OAI-PMH as its harvesting mechanism. There is a [harvesting panel](#) within DSpace (login required) that will enable an administrator to control which collections have a harvested source and whether they are set for active harvest. Dryad marks all collections that have a harvested source as active harvests (unless there is a problem with the remote OAI-PMH provider).

The one difference in workflow between the DSpace OAI harvester and the Dryad one is that Dryad has made some modifications to account for issues we've seen with records from TreeBASE. These changes can be seen in the [OAIHarvester](#) class that Dryad overlays. Searching the code for "TreeBASE" will locate the changes made to the source. They just represent another step in the workflow for digesting TreeBASE records. At some point in the future, they may be able to be removed and Dryad will then again use the standard DSpace OAI-PMH implementation.

Configuration

The configuration for each harvest is set in the DSpace [harvest control panel](#)

TreeBASE:

```
* http://www.treebase.org/treebase-web/top/oai
* Harvest all sets
* Harvest "Simple Dublin Core"
* Harvest metadata only
```

LTER/KNB:

```
* http://metacat.lternet.edu:8080/knb/dataProvider
* Harvest all sets
* Harvest "Simple Dublin Core"
* Harvest metadata only
```

There is an option within the harvest control panel to test a harvest. It will determine whether the service is properly configured, but may not be able to tell if there are particular issues with the harvest that would prevent it

BagIt Handshaking

from successfully completing.

There is also a configuration option in the dspace.cfg file to allow harvesting to be suspended on Dryad restart, but this is currently set so that harvests also restart automatically.

```
# Determines whether the harvester scheduling process should
# be started automatically when the DSpace webapp is deployed.
# default: false
harvester.autoStart=true
```

Relation to DSpace

There has been very little customization of the OAI-PMH harvester. It is directly related to the the DSpace implementation.

DSpace uses the PostgreSQL database by default.

Common Commands

```
\c [name]
    Connect to a database (like "dspace")
\d [name]
    Describes the format of a table, view, sequence, etc.
\dt
    List all tables in the database
\h
    help with postgres
\q
    quit
\?
    help with sql
```

Dryad provides a statistical display that takes into consideration Dryad's data package / data file distinction. For Dryad, there is the concept of a data package that contains all the data files (and other related files) associated with a particular publication. Each Dryad data package may have a relationship to multiple data files. Dryad wants to display statistical information about both levels of information: the data package and the data files.

Contents

- 1
- Functionality
- 2 Workflow
- 3
- Configuration
- 4 Relation to
- DSpace

Functionality

On the home page, we want to display an overall message about the number of data packages and data files in the form of: "As of Apr 25, 2011, Dryad contains 596 data packages and 1445 data files, associated with articles in 67 journals." The date should be the date on which the page is being viewed.

We also want to display a "viewed" and "downloaded" count for each package and file (on both the Dryad Data Package and Dryad Data File item view pages). Currently, only data files are downloaded, but in the future we will allow downloads of complete packages in a compressed file format. For now, packages display a number of times they have been viewed and files display a number of times they've been downloaded and/or viewed.

For example:

<http://datadryad.org/handle/10255/dryad.618> (package view)

<http://datadryad.org/handle/10255/dryad.621> (file view)

Workflow

DSpace stores its usage statistics in a Solr index called "statistics" (it has a Web-based administrative interface through which test queries can be run). Records stored in this index have an `owningColl` (owning collection) and `id`. Some with have an `owningItem` (owning item). Records that have an owning item are records for bitstreams (indicating a download) that belong to a data file DSpace item. Dryad doesn't attach bitstreams directly to data packages so any item that has been downloaded is associated with the data file.

Relationships between data file and data packages are not stored in the statistics index (since this is a concept overlaid onto DSpace by Dryad). These relationships though can be gleaned from the item records in the "search" Solr index. The "search" index is where general searching takes place and the relationships between records are stored there as linked identifiers. The "search" index also has a Web-based administrative interface through which test searches can be performed.

Since Dryad stores data packages and data files in separate collections, overall statistics (like the type displayed on the Dryad home page) can be gathered by checking the total number of active records in each collection. The additional information of how many unique journals have publications represented in Dryad is determined by querying the unique names of journals in the prism.publicationName field of the "search" Solr index.

All these statistical functions are performed in Java code which then writes a value into the `pageMeta` (page metadata). The Dryad XSLT theme then takes these values and displays them on the appropriate page when that page is rendered into HTML.

The Java classes related to the gathering of these Dryad specific statistics are kept in the Dryad overlay of the xmlui module. The classes have been put into the org.datadryad.dspace.statistics package to indicate they are specific to Dryad and not just a slight modification of an existing DSpace function. Here are the classes used and a brief explanation of each:

org.datadryad.dspace.statistics.SiteOverview

Generates the overall statistical summary displayed on home page

BagIt Handshaking

`org.datadryad.dspace.statistics.ItemStatsOverview`

Pulls together the stats generated from the `!ItemPkgStats` and `!ItemFileStats` and puts them into the page metadata

Caches the statistical generation so it doesn't need to be run at each page visit

`org.datadryad.dspace.statistics.ItemPkgStats`

Generates statistics for individual data packages

`org.datadryad.dspace.statistics.ItemFileStats`

Generates statistics for individual data files

For more details, consult the Java classes in the [org.datadryad.dspace.statistics](#) package directly.

The XSLT code that pulls the individual data package and data file statistics from the page metadata for display on the page can be found in the [DryadItemSummary.xsl](#) file. The XSLT code that displays the site's overall statistical information can be found in the default [Dryad.xsl](#) file.

Configuration

There is a minimal amount of configuration needed in the `dspace.cfg` file for the Dryad statistics to work. They, of course, rely on the locations of the Solr server to be set via the ``solr.log.server`` and ``solr.search.server`` variables. The Dryad statistics code also requires two additional variables be set ``stats.datafiles.coll`` and ``stats.datapkgcoll`` to indicate which collections contain the files and data packages.

```
# The handle for the data package collection's stats (also used by DOI minter)
stats.datapkgcoll = 10255/3
# The handle for the data file collection's stats (also used by DOI minter)
stats.datafiles.coll = 10255/2
```

Dryad sets the Solr server variables in the `dspace.cfg` file rather than the `dspace-solr-search.cfg` file so that we can use the standard Maven profiles mechanism to override these variables, depending on which Dryad instance we're running (demo, dev, production, etc.)

Relation to DSpace

Dryad's statistical display relies on the statistics log index created by the standard [DSpace Statistics](#) module. It also relies on the "search" index created by the [Discovery module](#). If Solr index fields change in the Discovery module, this may result in the custom Dryad statistical functionality to break (for instance if the publication names are put into a different index or if the relationships between files and packages are stored differently).

Dryad's submission system is a heavily modified version of the DSpace submission system.

Contents

- 1 Functionality
- 2 Workflow
- 3 Configuration
- 4 Relation to DSpace
- 5 Random Details
- 6 Temporary: adding a file to an existing package

Functionality

(need to integrate @mire's documentation here)

Workflow

(need to integrate @mire's documentation here)

Configuration

(need to integrate @mire's documentation here)

Relation to DSpace

Dryad's submission system builds off the basic framework for the DSpace submission system, but it bypasses the user interface and substantially modifies some of the underlying classes.

(need to integrate @mire's documentation here)

Random Details

Temporary: adding a file to an existing package

(This will go away once versioning is fully in place)

1. Export a single data file from the target package (using the -m parameter)
 1. /opt/dryad/bin/dspace export -i 10255/dryad.8434 -t ITEM -d /tmp/export -n 8434 -m
 2. Make the 'export' directory an empty directory just for exporting to (importing from)
2. Delete the exported bitstreams
3. Put the new bitstream in place
4. Edit the contents file -- remove the bundle:TEXT line, change the filename of the primary bitstream
5. Edit the dublin_core.xml file
 1. Update the dc.title
 2. Remove the old DOI
 3. Ensure isPartOf is set correctly
6. Run the import

BagIt Handshaking

1. `/opt/dryad/bin/dspace import -a -c 10255/2 -e rscherle@nescent.org -s /tmp/export -m /tmp/export/1.map`
2. The ".map" file should be a file that doesn't exist -- it will receive the output of the import process.
7. Look in the map file to see the handle of the newly created DSpace item.
8. Verify that this item exists in DSpace.
9. Check that the other files attached to that package have had their DOIs minted
 1. <http://datadryad.org/doi?lookup=doi:10.5061/dryad.bogus1234/1>
 2. <http://datadryad.org/doi?lookup=doi:10.5061/dryad.bogus1234/2>
 3. <http://datadryad.org/doi?lookup=doi:10.5061/dryad.bogus1234/3>
 4. etc.
 5. If the DOIs are not minted, mint them now, using the DOIService
10. Mint the DOI in the local DOI service
 1. <http://datadryad.org/doi?item=http://datadryad.org/handle/10255/dryad.bogus1234567890®ister>
11. Edit the data package, making the hasPart relationship point to the new data file.
12. Ensure that navigation works between the package and file, and that the DOIs resolve correctly.

The URLs and DOIs in the examples above are fake, just used for illustrative purposes.

Dryad harvests records from other scientific data repositories (like [KNB](#) and [TreeBASE](#)) via [OAI-PMH](#). (See description of the [Harvesting](#) process.) Harvested data is put into its own DSpace collection (one collection per harvested resource). We want to be able to provide a search across all these collections but treat the display of them differently that we would just another search facet.

We want to present each data collection in a separate search tab. The Dryad tab should be the default, but we want to give the user the option to view the results from these other collections. We want to display a hit count so the user will know, without clicking on the tab, how many resources are available from the other collection. This requires some customization of the DSpace Discovery module.

Contents

- 1
[Functionality](#)
- 2
[Workflow](#)
- 3
[Configuration](#)
- 4
[Relation to DSpace](#)

Functionality

Since the additional functionality that we want to provide is something that is, at least at this point, unique to Dryad we wanted to make as few changes to the underlying Discovery module as possible, preferring instead to provide a thin layer over the Discovery module that would treat our harvested collections differently from other collections (and from other search facets in general).

This additional layer is implemented in the Dryad theme's XSLTs and in a few additional !JavaScript files, also kept in the Dryad theme. The values that allow for this functionality are hard-coded values in the XSLTs and !JavaScript files, which works fine for Dryad but would require a change to make the functionality work for

a generic DSpace that adopted the same use of collections as Dryad. Because collection IDs are hard-coded, they need to have consistent collection numbers across different Dryad instances.

Workflow

When a user searches Dryad, the Discovery module is used to query a Solr index and return results. This takes place through a mix of Solr and Discovery specific search and display syntax. The tabbed searching that sits on top of Discovery works directly with Solr, avoiding the Discovery module, with the exception that it must take the Discovery query that is embedded in the page's metadata and convert any Discovery specific syntax into the corresponding Solr syntax.

This translation is done so that Solr can be queried directly to get the number of hits for the same search performed against a harvested collection. Parsing the Discovery query syntax is also important because the tabbed search layer must create a URL with the Discovery syntax so that when a user clicks on a search tab s/he performs that search in Dryad (using the Discovery module). The process works like this:

1. User queries Dryad
2. Discovery module handles query, returns results, and puts query URL into the page's metadata
3. Dryad XSLT processes page metadata and uses the functions in the !DryadSearch.xsl file to parse out the elements of the query (deduping when necessary, changing syntax to pure Solr syntax, etc.)
4. Dryad XSLT puts the cleaned query URL into a class attribute for each tab that will be displayed in HTML
5. Dryad XSLT associates each tab with a JavaScript file for that harvested collection
6. Dryad XSLT creates a link back into Dryad with the parameters required to query that harvested collection using the Discovery module (each collection has a collection ID within DSpace that can be passed in as a location parameter to narrow a search to a collection)
7. The user's browser loads the HTML that is created by the Dryad XSLT
8. On page load, the browser runs the !JavaScript files associated with a query to each of the harvested collections -- they, each, query Solr for the same search against the a different collection than the one used by the Discovery module
9. Instead of returning lots of XML results, Solr just returns the wrapper information for such a search, which includes a total number of results found (this is controlled by setting the Solr search paramer rows to 0)
10. The JavaScript that performed the async search then updates the HTML page to include the hit count (in parameters) in the tab for the externally harvested collection
11. It ends with the tab containing the number of hits a search would retrieve and a URL for that search to be performed in Dryad; the user can now click on that tab and display the results in the main Dryad interface
12. This toggling of which tab is active takes places in the Dryad XSLT and is based on the collection IDs that are passed as location parameters in the URL query; these are currently hard-coded values in the XSLT so any additional Dryad instances need to insure that they use the same collection IDs for collections (which should happen automatically when a database is copied from one instance to another)

Configuration

Configuration is hard-coded in the XSLT for now. It consists of associating an externally harvested collection with a particular DSpace collection ID, which is then used in the [DryadSearch.xsl](#)

BagIt Handshaking

The tabbed search process also references the DryadUtils.xsl and is initiated from template calls in the Dryad.xsl

There is a common JavaScript called solr-common.js and the collection specific values are put into small !JavaScript files to be called for each collection:

- * solr-treebase.js
- * solr-dryad.js
- * solr-lter.js

(There is probably an opportunity for refactoring here: instead of using separate !JavaScript files for each, passing in two parameters into the common file from the XSLT)

Relation to DSpace

The Dryad tabbed search functionality is related to, and uses, the DSpace Discovery module. The version of DSpace used by Dryad currently is 1.6.2. The version of Discovery that Dryad uses is tagged version 0.9.4

Discovery is going to be more completely integrated into the DSpace core going forward (though will remain a distinct module as DSpace moves towards structuring the entire codebase using smaller, interrelated modules).