DLS COLLECTION SYSTEM

(DCS)

USER GUIDE

VERSION 2 (DCS_v2_8)

TABLE OF CONTENTS

1.0 Introduction

1.1 Overview

The DCS is a java-based web application for supporting distributed collection building and management, and metadata generation. The application is designed to manage and generate metadata records that are based on XML schemas. The application is role and user based and has protections against concurrent editing. The application supports interactive searching, sharing of metadata records through OAI and web services and customized collection workflows.

1.2 Why Use the Application

Most often collection building groups use the application to create digital collections for institutional repositories or digital libraries and hence share such collections with an

extensive audience. The application allows for the creation of such digital collections by using application-ready collection workflows and metadata formats.

If a collection building group already has existing metadata that needs management, the application supports this as well if the metadata is described by an XML schema. Other groups use the application because it is web-based and allows users to connect from anywhere. The application is flexible and customizable especially with its web service that can be used to create many different views or user interfaces over the stored data.

1.3 Intended Audience for User Guide

This Users Guide is comprehensive and detailed. It is intended for all levels of users of the application. But not all users need to read every section to be able to perform their work. Please read the next section on strategies for using the application and Users Guide.

1.4 Strategy for Using the Application and User Guide

The top column headings of Table 1 below describe several different types of users for the application and User Guide. Under each user type, there is a bulleted list of expected tasks for that user. Determine which type of user best describes your situation. Then as you move down the column, the table rows define further tasks expected of the user and the User Guide section numbers that provide task explanations and how to's. For your user type, read the appropriate User Guide sections on the table rows marked with an **X**.

Table 1: Strategies for Using the Application and Users Guide

	Developer/ Sys. Admin Type	***Collection Manager Type***	***Cataloger Type***	***NSDL TNS Staff***
Tasks to Perform and User Guide Sections to read.	• Local install & configuration • Set up collections • Add users • No cataloging • Create own metadata framework	• App is already installed and configured • Set up collections • Add users • Catalog resources	• Collections exist • I know the application URL, login & password • Catalog resources	• Manage collections at nsdl.org • Catalog resources
1. Install & configure the application (all sections of 2.0).	X			
2. Set up collections. Decide collection authorities (sections 3.1, 3.2 and 3.14).	X	X		X
3. Determine collection workflows. Create appropriate workflow statuses for each collection. It is *very* important that collection building groups give this careful thought (sections 5.1 through 5.5).	X	X		X
4. Add users. Assign user roles & collection access (all sections of 4.0)	X	X		X
5. Enter any new and/or existing metadata records into a collection (section 3.3) .	X			
6. Catalog records. (all sections of 6.0, 7.0 & 8.0).		X	X	X
7. Indicate cataloged records are complete (sections 5.4 and 5.5).		X	X	X
8. Address operational errors (all sections of 14.0).	X	X	X	X
9. Share metadata records: • Export records to the local file system (section 3.10). • OAI provide records (section 12.0). • Register your collection with the NSDL. Required to make collections visible at NSDL.org even if you registered your collection with the NSDL Data Repository (NDR) (section **Error! Reference source not found.**).	X	X		X
OPTIONAL: Custom Metadata Frameworks (section 15.0).	X			
OPTIONAL: Manage NSDL collections (section 16.0)				X

2.0 INSTALLATION

2.1 BACKGROUND

The software runs in the Apache Tomcat servlet container (tested only in Tomcat) and can be installed on Windows, Linux, Mac OS or UNIX systems. These instructions assume familiarity with Java Servlets, Java Server Pages (JSP), the Tomcat servlet container and related technologies. The installation procedure has been tested on LINUX, Windows XP and MacOS v10.3.

If you have not downloaded the application software yet, information on downloading it is available at: http://wiki.nsdl.org/index.php/Community:DCS.

2.2 STEP 1: OBTAIN AND INSTALL TOMCAT AND JAVA

Obtain and install Tomcat (7.x) and Java (1.7 or 1.8) if necessary. The Tomcat distribution provides instructions within the RUNNING.txt file. In this document, your Tomcat installation directory will be referred to as $TOMCAT.

2.3 STEP 2: EXPAND THE WAR FILE

Place the war file (DCS.war) in the webapps directory ($TOMCAT/webapps). Start/restart Tomcat to unpack the war files and install the web application. Startup may take a minute - if you want to monitor progress, you can watch the log file using the following Unix command: tail \-f \-n100 $TOMCAT/logs/catalina.out

NOTE: Each installed instance of the application has a name. This name is the name given to the war file. So if you rename the war file from DCS.war to biology.war, your application instance name will be 'biology' instead of 'DCS' and this name will appear at the top center of each page in the application. Throughout the rest of this documentation, the name of this war file and its directory is called **<appName>**. Whenever you see <appName>, replace <appName> with the name of the war file.

2.4 STEP 3: VIEW THE APPLICATION

Access the system with your browser at http://localhost:8080/<appName>. You will land on the application ***Home*** page.

NOTE: At the top of the page you will see a message to the effect that the Sample Collection must be re-indexed before it can be accessed. This is normal!

2.5 STEP 4: CONFIGURE THE APPLICATION

To configure the application, use the context descriptor template found at: $TOMCAT/webapps/<appName>/docs/templates/context_descriptor_template.xml.

The context descriptor template lists configuration parameters for the application. It describes their usage and suggests values (for more explanation, see Appendix 1: CONTEXT DESCRIPTOR PARAMETERS). Please note it is **strongly** recommended to supply

a directory path outside of the application folder for these configuration parameters; otherwise your data (records, editor of the records, status of the records) will be overwritten whenever you choose to update the application software.

Copy and edit the context_descriptor_template.xml file as desired, and then place its contents within $TOMCAT/conf/server.xml (within the **Host** XML element). Restart Tomcat to apply your configurations.

2.6 STEP 5: CONFIGURE APPLICATION FOR USER ROLES AND PERMISSIONS (OPTIONAL)

This optional step allows you to define authenticated users and to assign them a role, which defines the actions they have permission to carry out. Roles are used to control access to collections and system functionality.

There are two steps to enable the roles and permissions mechanism:

1. In the context descriptor of your configuration, (now in your server.xml), set the **authenticationEnabled** parameter to **true**.

2. A parameter must be passed to the JVM when it is invoked (i.e., when Tomcat starts). This parameter tells the JVM where to look for the java class necessary to support user authentication at login ($dcsConf corresponds to the value of the dcsConf parameter defined in the context descriptor).

   -Djava.security.auth.login.config==$dcsConf/auth/DCSlogin.config"

In UNIX, the recommended method for passing the required parameter is via a shell script that will start and stop Tomcat (and can be called by a system init script if desired). A template script file can be found at templates/DCS.sh_template. The template contains the necessary instructions for customizing it to your installation.

For WINDOWS, the recommended method is to pass the parameter via the "Apache Tomcat Properties" tool (found at $TOMCAT/bin/tomcat5w.exe). After opening the properties tool, click on the "Java" tab. Add the following line to the "Java Options" pane: -Djava.security.auth.login.config==$dcsConf\auth\DCSlogin.config

Now restart Tomcat to apply your change. When you try to access the application, you will be challenged for a username and password. Use the following:

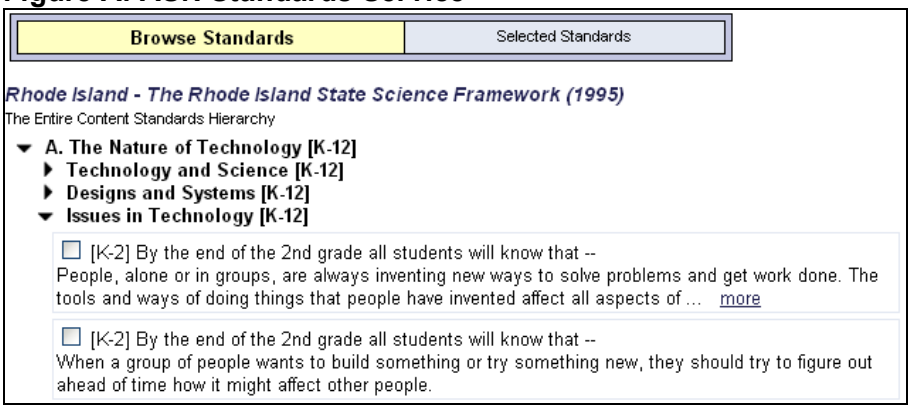
username: root
password: root!pass

NOTE: It is strongly suggested that you change the root user's password (see section 4.1 CHANGE THE ROOT PASSWORD).

2.7 STEP 7: CONFIGURE APPLICATION TO ENABLE DCS STANDARDS SERVICE

The DCS Standards Service helps catalogers to select and assign educational standards, such as the ASN Standards (http://asn.jesandco.org/). ASN standards have the form ("http://purl.org/ASN/resources/S10023EC") which is tedious to type and which has no semantic meaning.

Figure A: ASN Standards Service



The Standards Service (see Figure A) displays ASN Standards in textual form, in their hierarchical structures. When catalogers select a standard, it is inserted into the metadata as an ASN ID.

The Standards Service is configured via a single XML file which contains configurations for each framework that will use the service. The configuration for a particular framework will apply to each of the collections belonging to that framework.

A framework configuration specifies:

- the *framework*, e.g., 'Ncs_item'
- the *xpath* of the metadata field for which ASN standards will be displayed, e.g., '/record/educational/standards/asnID'
- the Standards Documents from which suggestions will be made. There is no need to manually configure the individual standards documents, this is done through the "Manager -> Standards" user interface in the DCS.

An example standards Configuration file can be found at $TOMCAT/webapps/<appName>/docs/templates/suggestion_service_config.xml.

To complete the configuration, perform the following steps,

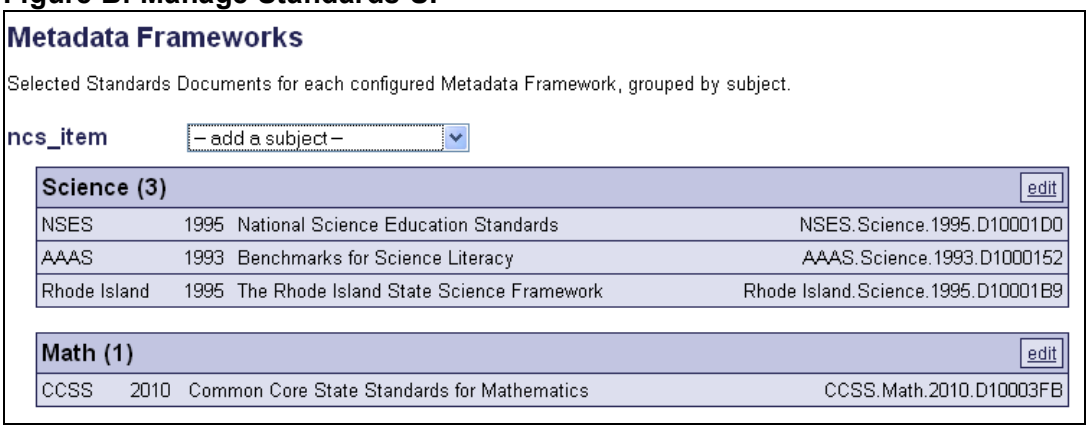
a. copy the template file, and save it to disk (e.g., $DCS_CONF/standardsServiceConfig.xml). Uncomment the frameworks for which you would like to use the standards service.

b. set the value for the **standardsServiceConfig** parameter in the context descriptor to point to the config file.

c. choose a location for the ASN Standards Library. This is where downloaded ASN Standards Documents are cached (the DCS downloads ASN Docs as necessary as you work). Configure the path to the location as the value of the **asnStandardsLibrary** param in the context descriptor.

NOTE: The standards service may be disabled or enabled on a per-collection basis via the "Settings -> Collection-Settings" page. By default, the standards service for a collection is active if the collection's framework has been configured with a Standards Service. To enable or disable the suggestion service for a particular collection, navigate to the settings for the collection and edit the "allowSuggestions" field to either allow or disallow suggestions.

Figure B: Manage Standards UI



Once you have started the DCS application and the standards service configuration has been processed, you can use the "Manager -> Standards" user interface in the DCS to enable specific standards documents for each configured framework. The set of enabled documents determines what standards can be cataloged for each framework. For example, the configuration for the Ncs_item framework shown in Figure B would allow catalogers to assign standards from 3 Science and 1 Math standards documents.

Once CAT is enabled for a metadata framework, enable CAT for a collection by completing the tasks outlined in:

1. Section 3.1 COLLECTION SETUP
2. Section 3.2 COLLECTION AUTHORITY
3. Section 3.16 ENABLING/DISABLING STANDARDS SERVICE FOR A Collection

If you would like to activate CAT for any other supported metadata frameworks, follow the procedure in section 15.9 ENABLING THE CAT SERVICE FOR A CUSTOM METADATA FRAMEWORK.

2.8 STEP 8: CONFIGURE APPLICATION TO USE BOUNDING BOX TOOL (OPTIONAL)

The Bounding Box Tool aids a cataloger in determining the bounding box coordinates (latitudes and longitudes) of a rectangle on the Earth's surface. A cataloger draws a box of interest on a map and then the Bounding Box Tool automatically determines the northern and southern most latitudes of the box and the westernmost and easternmost longitudes of the box. Then these values are inserted in the appropriate metadata fields.

Please note that setting a value for this configuration enables the Bounding Box Tool for the **Ncs_item** metadata framework described in Section 6.3.

To activate the Bounding BoxTool, edit the framework configuration file at: $dcsConf/frameworks for Ncs_item. Uncomment the following line:

<!-- <path pathName="box" inputHelper="bounding_box/Ncs_item_box_widget.html">/record/coverages/box</path> -->

If the application is already up and running, please note that you need to reload the Ncs_item metadata framework by:

1. Click on ***Settings*** and select ***Metadata Frameworks***.
2. Click ***Reload*** for the **Ncs_item** framework.

The Bounding Box Tool is now ready to use. For an image of how this looks in the application, see section 6.3 NCS_ITEM METADATA FRAMEWORK.

The Bounding Box geospatial functionality was developed in part at the Lawrence Hall of Science, U.C. Berkeley, for the SMILE Pathway.

2.9 STEP 9: OPTIONAL USEFUL CONFIGURATIONS

Appendix 1 describes other useful parameters that can be configured in the application. Please note that other sections of this Users Guide describe some of these parameters in more detail. Some of these include:

1. **collBaseDir** and **dcsConf** - hinted at in section 2.5
2. **authenticationEnabled** - described in section 2.6
3. **ndrServiceEnabled** - described in section **Error! Reference source not found.**
4. **ndrApiBaseUrl** - described in section **Error! Reference source not found.**
5. **DCSAgentHandle** - described in section **Error! Reference source not found.**
6. **DCSAgentPrivateKey** - described in section **Error! Reference source not found.**
7. **asnStandardsDocument** - described in section 2.7
8. **exportBaseDir** - described in sections 3.10 and 3.11
9. **logo** - described in section 10.0
10. **oaiPmhEnabled** - described in sections 12.2 and 12.3
11. **standardsSuggestionServiceConfig** - described in section 15.9

To see the current settings of some of these parameters, choose ***Settings*** from the main menu and then select ***App Configurations***.

3.0 COLLECTION MANAGEMENT

3.1 COLLECTION SETUP

When creating new collections, you will be asked for the following information:

- Collection full title
- Collection short title
- ID prefix for record identification numbers
- Collection authority
- Depending on the metadata framework chosen, you will be asked for one or all of the following additional pieces of information:
    - Metadata copyright
    - Metadata terms of use
    - A URL to a metadata terms of use
    - Service name - (used in annotation services)

These are explained in the setup procedure below, however, it is important to be prepared to enter this information since this User Guide does not have a section that explains how to do this after the fact. Once you have these pieces of information, make any collection in the application, by:

1. Chose ***Manage*** from the main menu and select ***Mange Collections***.
2. Click the ***Create New Collection*** button.
3. Enter a ***full title*** and ***short title*** for the collection. The short title will be the title displayed in the collection list in the application. Please note that short titles can only contain alphanumeric characters, hyphens, underscores and spaces. Apostrophes and quotes are not allowed.
4. Select the ***metadata format*** for the item records within the collection. Depending on the metadata format selected, enter additional information as requested. [Important: see note above about this.]
5. Enter an ***ID Prefix*** for the collection. This prefix is used in all record identification numbers, (e.g. ABC-000-000-000-000), where ABC is the ID Prefix you enter. Choose something easy to remember that provides information about the collection like: GEO for a geoscience collection.
6. Click the ***Submit*** button.
7. On the ***Collection Confirmation*** page, choose the authority of the collection. That is, where do you want records to be stored:
    a. On your local file system? Do this:
        i. Do *not* click the *Register* button.
        ii. Complete section 3.2 COLLECTION AUTHORITY below with the ***dcs*** option and return here.
    b. In the NDR? Do this:
        i. Click the *Register* button. You will be taken to the *NDR Admin* page.

ii. The name of the new collection should appear in the table (indicating it is registered in the NDR).

8. Add collection workflow statuses by completing section 5.0 WORKFLOW STATUSES.
9. Make records in the collection. See section 3.3 ENTER RECORDS INTO A COLLECTION.

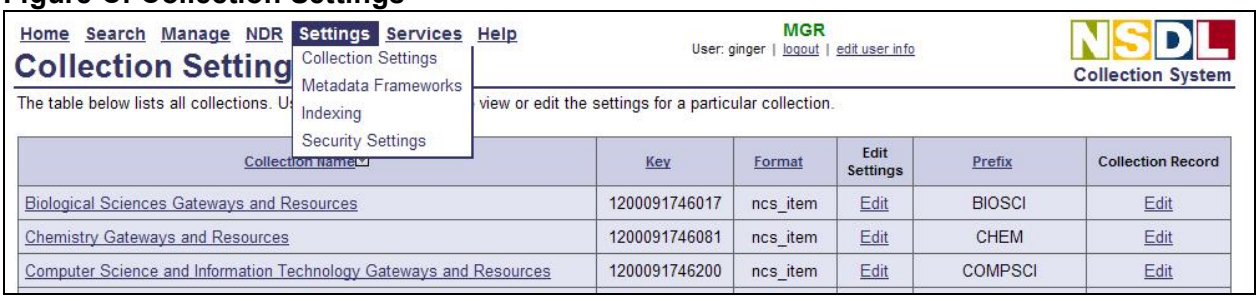
3.2 COLLECTION AUTHORITY

Collection authority refers to where you would like to store your records, that is, in the NSDL Data Repository (NDR) (**ndr** option) or on your local file system (**dcs** option)? By default, each newly created collection has an authority of 'unspecified' and records are stored using your local file system (**dcs** option). Collection authority can only be changed by users with a role of **Administrator**.

Collection authority should be determined and changed now. If you do not change it now you will be prompted later when you are completing collection workflow statuses. If you do not know which option to use, choose your local file system (**dcs** option). You can always change it to the NDR later but cannot easily do the reverse. To change the collection authority, do the following:

1. Chose ***Settings*** from the main menu (see Figure C) and select ***Collection Settings***.
2. Click on the desired collection name. This will take you to a view of the collection's configuration record.
3. Click the ***Edit Settings*** button. You are now on the ***Collection Configuration*** page.
4. Scroll down to the ***authority*** field. Choose either the **ndr** or **dcs**. (an authority must be entered to save this collection configuration record).
5. Click the ***Save*** button and then click the ***Exit Editor*** button.

Figure C: Collection Settings



3.3 ENTER RECORDS INTO A COLLECTION

There are several ways to enter records into a collection and the method that you use depends on where you are in the collection building process. Any of the method can be used at any time. The methods include:

- **Ingest existing records** - brings existing metadata records, in a compatible metadata format, into a collection. See 3.4 INGEST EXISTING RECORDS.

- **Create new records** - makes new records from scratch by direct cataloging. See section 3.5 CREATE NEW RECORDS.
- **Copy records within a collection** - makes new records in a collection by copying existing records. See section 3.6 COPY RECORDS WITHIN A COLLECTION.
- **Move records between collections** - makes new records in a collection by moving records in from another collection that uses the same metadata format. See section 3.7 MOVE RECORDS BETWEEN COLLECTIONS.

3.4 INGEST EXISTING RECORDS

To bring existing records into a collection, several criteria should be verified before ingesting. These criteria are:

- The collection to which records are to be ingested should exist in the application.
- Workflow statuses for the collection should exist and be complete so that ingested records can be assigned their appropriate workflow status. This is especially important if ingested records will need to have several different workflow statuses assigned.
- The records to be ingested must be in XML and the XML must be the same metadata format being used by the collection in which they are being ingested.
- No other users in the application (not just collection) should be editing, creating, moving, indexing or doing any tasks while ingesting is occurring.
- If the collection in the application already has existing records, the ingested records should not have duplicate record identification numbers of any record already in the collection.
- And if possible, ingested records should not have duplicate URLs, nor duplicate any URLs already in the existing collection.

If you have records that need different workflow statuses after being are ingested, it is best to complete the ingest process in batches. That is, repeat the ingest process for each different intended workflow status. For example, ingest all **In Progress** records and assign the **In Progress** workflow status after ingesting. Then ingest all **Done** records and assign the **Done** workflow status. Repeat until all records are ingested as described in the procedure below.

To ingest a batch of records, you must have **Administrator** access within the application and access to write to the computer file system on which the application resides. Also, it helps if you are an experienced user of the application and are comfortable doing batch operations. Then do the following:

1. Chose ***Home*** from the main menu.
2. Assuming that no records exist in the collection you are ingesting records into, click the ***Create*** link on the desired collection. This will make a single sample record. Save it and exit the ***Metadata Editor*** (this record creation is not necessary if records already exist in the collection).
3. Choose ***Home*** from the main menu. Click on the name of the desired collection to see the list of records within the collection. Look at the first record.

4. Note (copy or write down) the file location directory information of the record excluding the filename.xml part.
5. Logout out of the application.
6. Place the records to be ingested in the file location directory.
7. Stop/start the application's tomcat.
8. Login back into the application and you may see an error about your collection needing to be re-indexed. This is normal. This is what you are going to do next.
9. Choose ***Manage*** from the main menu then select ***Manage Collections***.
10. Find the name of your collection and then move to the ***Index Records*** column and click ***Index***. Wait for indexing to complete. If you experience errors, check your XML and re-ingest until 0 indexing errors are reported. If problem records persist, think about deleting these records and ingesting them later or making them manually in the collection (see sections 3.8 REMOVE RECORDS FROM A COLLECTION or 3.9 DELETE AN ENTIRE COLLECTION).
11. You must now validate the records for the collection to work properly. Locate the ***Validate Records*** column and click ***Validate***. You will be taken to the Validate Collection page.
12. Click on the 'Ignore cached validation results' box to toggle it on so that all records will be validated from scratch. This provides the most accurate results. Do not choose a status to be validated.
13. Click the ***Validate Collection*** button and wait for validation to complete. It could take a while and if you have errors you should address these until you have 0 validation errors (or the expected number of validation errors if ingesting invalid records). Depending on the nature of the validation error, you can fix these errors by re-ingesting updated records or editing each individual record through the ***Metadata Editor*** (see section 7.1 METADATA EDITOR) of the application.
14. To read any errors, click on ***See Report*** in the message box. Hit your browser's back button to return to the Validate Collection page.
15. When you are back on the Validate Collection page, click on the ***Return to manage collections*** button. Then click on the collection name to see item records as search results. Notice that your ingested records have a status of **Unknown**.
16. Assign an appropriate workflow status to the ingested records. Perform a search to find your newly ingested **Unknown** records (see section 9.3 QUICK SEARCH for search help). Use the application's batch mode ability to change the status of these records to something more informative like **New**, **Done**, etc. (see section 5.7 CHANGE THE WORKFLOW STATUS OF A BATCH OF RECORDS). This will take awhile, so be patient.
17. Repeat this process until all desired records are ingested and have a proper workflow status.

3.5 CREATE NEW RECORDS

Use this method when creating the first few new records in your collection or when copying an existing record is not appropriate. Be on the **Home** page of the application and click the ***Create*** hyperlink next to the collection name that you want to create the record in. The record is given a workflow status of **New**. You may be asked for a URL or land directly in the ***Metadata Editor*** (see section 7.1 METADATA EDITOR).

3.6 COPY RECORDS WITHIN A COLLECTION

Records are often copied to create new records in a collection. This is done when you want the new record to be very similar to an existing record and all that needs to be changed is generally the title, URL and description of the resource. A record can be copied as many times as you like and by any user with access to that collection. Follow the steps below to complete a record copy.

1. Be on the ***Search or View Record*** page (have the desired record on the screen), then click the ***Copy*** button.
2. When copying records, generally all information is copied except (and this is dependent on the metadata framework of the record):
    - title
    - description
    - URL
    - catalog record number (a new one is assigned)
3. You will land in the ***Metadata Editor*** (see section 7.1 METADATA EDITOR) and can begin editing the record directly. Once you save this record, it is given a workflow status of **New** and the last editor is assigned to be you (i.e. the user who copied the record).

Tips on copying.... If you would like to have the same record appear in multiple collections, copy the record multiple times and move a copy of the record to each desired collection.

3.7 MOVE RECORDS BETWEEN COLLECTIONS

A record can only be moved to a single collection, not multiple collections. To move records, the collections involved must be using the same metadata format. The metadata format of each collection is listed on the ***Home*** page of the application. To move records, a user must have access permission to both collections. Follow these steps to complete a record move.

1. Be on the ***Search*** or ***View Record*** page (i.e. have the desired record on the screen), then click the ***Move*** button. (please note that records can also be moved in batch mode; this is not covered by these simple instructions)
2. From the drop-down list, select the destination collection. Then click the ***Move*** button.
3. Click either ***OK*** or the ***go to the collection*** button. I generally click the ***Go to Collection*** button.
4. In the new collection, you might have to search for the new record and it will have a workflow status of **Imported**. Assign the desired workflow status. This will update the *Last Editor* information from **Unknown** to you (the changer of the workflow status).

3.8 REMOVE RECORDS FROM A COLLECTION

A record can be deleted from a collection by any user with access to that collection. Be on the ***Search*** or ***View Record*** page (i.e. have the desired record on the screen), then click the ***Delete*** button.

3.9 DELETE AN ENTIRE COLLECTION

Any collection can be deleted outright. This action cannot be undone and requires a user role of either **Manager** or **Administrator**. Follow these steps to delete a collection:

1. Choose ***Manage*** on the menu and select ***Manage Collections***.
2. From the ***Delete Collection*** column, selected the desired collection and click on the ***Delete*** hyperlink.
3. You will be asked to confirm.

3.10 EXPORT RECORDS FROM A COLLECTION

Records can be exported out of the application to a designated directory on the local file system. Since records are stored in a single directory, regardless of the workflow status of a record, it is very useful to export records to another place on your file system. For example, you could export all records that have a workflow status of **In Progress** to one directory location and export all records that have a workflow status of **Done** to different directory location. It does not matter if your collection authority (see section 3.2) is the **dcs** or the **ndr**. The export function can be used with both authorities.

You must have a user role of **Manager** or **Administrator** to export records. When records are exported, a copy of the record is placed in a directory that is defined by the combination of the **export base path directory** and the **destination path directory**. Both directory paths must be set in order to export records. And the **destination path directory** is always a relative directory path to the **export base path directory**. **Destination path directories** cannot use absolute directory paths.

When the application is installed, a default **export base path directory** is set to $TOMCAT/webapps/<appName>/WEB-INF/data/exported and no **destination path directory** is set. The **export base path directory** applies to every collection within the application. The WEB-INF directory is within the application and it will be deleted if the application software is updated. So make a copy of your exported records in another location before upgrading the application software.
If you would like to change the **export base path directory**, go to section 3.11 CHANGE EXPORT BASE PATH DIRECTORY. Then return here to follow the instructions for exporting records.

If you would like to change the **destination path directory**, go to section 3.12 CHANGE DESTINATION PATH DIRECTORY below. Then return here to follow the instructions for exporting records.

1. Choose ***Manage*** on the main menu and select ***Export Reports***.
2. Click the ***Export a Collection*** button.

3. Select a collection from the ***Collection to Export*** drop-down menu.
4. Choose a workflow Record Status to export if desired. All records are exported if no status is selected. Most often you will want to choose the **Done** records of your collection.
5. Set a destination path if desired. If one has been configured, it will show. If you would like to change it, go to section 3.12 CHANGE DESTINATION PATH DIRECTORY. Then return here to continue exporting records.
6. Click the ***Export*** button.
7. Please note that only valid XML records are exported regardless of the workflow status chosen. Any invalid records are not exported. A message will display indicating the number of records exported and the number of records not exported. The number of 'not exported' records indicates the number of invalid XML records.

3.11 CHANGE EXPORT BASE PATH DIRECTORY

The ***export base path directory*** is used to export records as described in section 3.10 EXPORT RECORDS FROM A COLLECTION. To change it, you must have access to the context descriptor. If you do not know your current or default ***export base path directory***, you can see in the application by:

1. Choose ***Manage*** on the main menu and select ***Export Reports***.
2. Click the ***Export a Collection*** button.
3. You will then see two dialog boxes. Just above the second box is the **export base path directory** where records will be exported to.
4. Open your context descriptor in $TOMCAT/conf/server.xml.
5. Modify the **exportBaseDir** parameter.
6. Restart tomcat. Your ***exportBaseDir*** is now set and it will be the export base path directory for all collections within the application.

3.12 CHANGE DESTINATION PATH DIRECTORY

The ***destination path directory*** is used to export records as described in section 3.10 EXPORT RECORDS FROM A COLLECTION. There are two methods that can be used to change/control it. Choose the method that works best for you.

- **Method 1: Affect Current Export Only**. The entered directory path affects the current export only. It is not remembered. This method can also be used to override method 2. This method requires a user role of **Manager** or **Administrator**.

    1. Choose ***Manage*** on the main menu and select ***Export Reports***.
    2. Click the ***Export a Collection*** button.
    3. You will then see two dialog boxes. In the second box add your own ***destination path directory*****.**

- **Method 2: Affect all Collection Exports**. The entered directory path affects all exports. It is remembered. It can be overridden by method 1. This method requires a user role of **Administrator**. Configure a destination export directory (relative to the export base path directory) by:

1. Choose ***Settings*** from the main menu and then select ***Collection Settings***.
2. Select the name of the collection you wish to modify and then choose the ***Edit Settings*** button.
3. Enter a directory name in the ***exportDirectory*** field that is relative to the base path.
4. The click the ***Save*** button.
5. Go back to ***Manage*** and ***Export Reports*** to export the collection.

If your destination path directory is not relative to your export base path directory, the application will give an error. If your destination path directory is relative to your export base path directory and does not previously exist, the application will make the directory path and then export the records.

3.13 Sample Collection Re-index Error

Upon initial installation and startup of the application, the homepage will display a message to the effect that the Sample Collection must be re-indexed before it can be accessed. This is normal. To fix this message do the following:

1. Chose ***Manage*** from the main menu and then select ***Mange Collections***.
2. Select the ***Index*** link for the Sample Collection table row.
3. Click ***OK*** to remove indexing messages

3.14 Using Test Collections as a Playground

Often it is useful to make a test collection of the metadata format you work with most often. This will allow you to try out different application actions, questions or concerns without affecting real records. This is very important if your collection is communicating to the NDR. It's difficult to undo NDR actions.

3.15 Assign Collection Access

Assigning collection access for particular users is often done when a user is created. However, it can be changed at any time by completing the following steps:

1. Chose ***Manage*** from the main menu and then select ***Mange Users***.
2. In the table, locate the user you want to adjust access for.
3. Click the ***edit access*** link and adjust the user's user role for each desired collection.
4. Scroll down. Click the ***Save*** button.

- Description of the collection
- Contact person for the collection
- Audience of the collection
- Appropriate grade ranges for the collection
- Expected number of items to be in the collection
- Information on how you plan to share the collection (e.g. OAI, in this application)

3.16 Enabling/Disabling Standards Service for a Collection

Before the DCS Standards Service can be used in the cataloging of item-level metadata in a collection, it must be activated for the collection. Be sure standards service is configured for the metadata framework the collection uses (see section 2.7). Then a user with the role of **Administrator** can enable or disable the service for each collection by doing the following:

1. Choose ***Settings*** from the main menu and then select ***Collection Settings***.
2. Click on the desired collection name to view the Collection Configuration Record.
3. Click on the Edit button.
4. Click choose next to the services field.
5. For the allowSuggestions field, choose true to enable or false to disable.
6. Then click the ***Save*** button to save your configuration changes.

4.0 User and User Roles

Users and user roles control access to collections and system functionality. Each person (user) working in the application can have an account with a distinct username (login), password and user role (cataloger, manager, administrator). There is also a root login and password included as part of the application's installation.

4.1 Change the Root Password

1. Login into the application using the root login (root) and password (root!pass).
2. At the top center of the homepage, click on the hyperlink ***edit user info.***
3. Enter a new value for the root password.
4. Reenter the new root password.
5. Click the ***Save*** button. Then click the ***Exit*** button. Be sure not to loose this information.

4.2 User Roles and Permissions Defined

The application has three primary user roles (**Cataloger**, **Manager** and **Administrator**). Each role has certain allowable tasks and actions with each higher user role able to do its own task/actions and those of the user roles less than it. That is, a **Manager** can do their tasks/actions as well as those tasks/actions of a **Cataloger**.

When you log into the application, the *Home* page displays all collections in which you have access permissions. The far right-hand column displays your user role associated with each collection. The list below describes the tasks and actions that can be accomplished by each assigned role. If you need to change your role for a collection, a user with the role of **Manager** or **Administrator** of that collection can change your permissions.

1. **Cataloger** – basic access to perform item-level cataloging on the collections to which they have access
   - Has access to the ***Home*** and ***Search*** areas.
   - Can edit, delete, move, view and copy any records

- Can change status of any records
- Can search any collections or records

2. **Manger** – greater access to perform collection-level functions on the collections to which they have access
   - Has access to the ***Home***, ***Search***, ***Manage*** and ***NDR*** areas
   - Can export records to the file system or sync records to the NDR
   - Can validate a collection
   - Can create/delete a collection
   - Can re-index a collection
   - Can add users to a collection
   - Can delete users from a collection
   - Can unlock locked metadata records
   - Can register collections with the NDR
   - Can ***not*** add/edit workflow statuses of a collection

3. **Administrator** – super-user access to control how the application operates; access to everything
   - Has access to the ***Home***, ***Search***, ***Manage***, ***NDR***, ***Settings*** and ***Services*** areas
   - Can re-index all collections at once
   - Can reload all controlled vocabularies across all collections
   - Can change default settings/workflow statuses for any collection
   - Can change where files get exported to on the file system
   - Can change how the various metadata frameworks operate
   - Can change security settings
   - Controls NDR, OAI and Web services
   - Can add other administrators
   - In the list of users, administrator users do not have a role associated with them because they have access to everything.

4.3 MANAGE USERS

It is easier to create and configure user access after your initial collections have been set up. This is because you can then assign the correct collection access to any newly created user. It's not mandatory to set up your collections first, just easier if most of your users are going to have the role of **Cataloger**.

To ***create a NEW*** user, decide upon the appropriate user role for each user. Then do the following:

1. Chose ***Manage*** from the main menu and select ***Mange Users***.
2. Click the ***Create New User*** button.
3. Enter a ***username*** for the new user. Usernames must be unique. To see existing usernames, click on the ***Registered Users*** link next to the form.
4. Complete the rest of the form, especially institution and email.
5. Determine whether you want this new user to have a role of administrator. Choose no or yes.

6. Click the ***Save*** button.
7. Then click ***edit access*** to assign collection access permission if the user does not have the role of administrator.

To ***modify*** existing user information follow the same steps above and just change what's needed.

To ***remove*** an existing user, be on the ***Manage Users*** page and click on the ***delete user*** hyperlink on the far right in the row of the user you want to remove. Please note that if the deleted user is listed as the last editor of any record, the name of the last editor for those records will change to the username of the deleted person. This username displays in italics to indicate they are no longer an active user in the application. You can still search for them by their username. See the example below for a person named Jane Clark with a username of clark who has been removed.

Before Jane Clark is deleted as a user.

Record ID	Last Editor	Status	Last Touch
MATH-000-000-000-001	Jane Clark	Done	2008-10-06

After Jane Clark is deleted as a user.

Record ID	Last Editor	Status	Last Touch
MATH-000-000-000-001	*clark*	Done	2008-10-06

5.0 WORKFLOW STATUSES

5.1 WORKFLOW STATUSES DEFINED

Every record in every collection has a workflow status. This is a label, like New, Needs Review, that tracks the progress or current status of a record. That is, workflow statuses are vital for collection management. Collections can have different sets of workflow statuses. This means one collection could call its finished records **Done** and another collection could call them **Published**. Each workflow status also has a definition like:

> **In Progress** - The record is being actively cataloged.

In a collection, a record can only have a single workflow status at any time. A history of workflow statuses for a record is maintained.

5.2 RESERVED AND DEFAULT WORKFLOW STATUSES

The application has reserved and default workflow statuses. Reserved workflow statuses cannot be changed at all. That is, the label and definition are as is. Default workflow statuses may have their labels and definitions changed or be deleted

altogether (except for **Done** which can only be renamed; see below). These reserved and default workflow statuses are intended to help get a collection up and running by providing a simple set of workable workflow statuses.

For any collection, you can add your own workflow statuses; see the section 5.3 SUGGESTED ADDITIONAL WORKFLOW STATUSES FOR A COLLECTION. The following is a list of the application's reserved and default workflow statuses. The definition of each workflow status is shown in italics and some notes about how to use the workflow status in the collection building process are provided.

- **Unknown** - *(Reserved) The metadata record has not been assigned a status*.
    - Use to indicate records that have been ingested into the application from the file system and not created directly using the application.
    - Unknown records should then be assigned a meaningful workflow status.
    - Cannot be re-named, deleted, nor added as a user-defined workflow status.
    - Definition cannot be modified.
    - However, a user can intentionally assign records to have an **Unknown** status.

- **Imported** - *(Reserved) The metadata record needs to have a status assigned*.
    - When making a record using programmatic methods or external applications like a web form or web service, make your application assign this status to newly created records. This allows easy searching for records created in such a manner so they can be processed.
    - Also automatically assigned by the application when moving a record between collections. This is because the application does not know what the status of the record should be when it has been moved to a new collection.
    - After records have been imported into a collection, they should be assigned a meaningful workflow status such as **New**.
    - Cannot be re-named, deleted, nor added as a user-defined workflow status.
    - Definition cannot be modified.
    - A user cannot intentionally assign records to have an **Imported** status. This can only be done by the application.

- **New** – *(Reserved) The metadata record is ready to be cataloged*.
    - Records made through using the ***Create*** link on the application ***Home*** page are assigned a status of **New**.
    - When records are copied, the copied record is automatically assigned a workflow status of **New**.
    - It is expected that new records eventually become completed items within a collection.
    - Cannot be re-named, deleted, nor added as a user-defined workflow status.

- Definition cannot be modified.
- Users can intentionally assign records to have a status of **New**.

- **Recommended** - *(Reserved) The metadata record is a recommendation for inclusion in the library*.
  - Use this status when a web form is used to create records within the application. For example, if you intend to have something like a *Recommend a Resource* you will be able to track the recommended resources by using this status.
  - This is a special status connected to forms. Only when the status of the record changes from **Recommended** to something else (e.g. **New**, **In Progress**, **Declined**) will NDR action occur. See section 11.1 WHEN ARE RECORDS WRITTEN TO THE NDR??
  - It is expected that most recommended records eventually become cataloged collection items.
  - Cannot be re-named, deleted, nor added as a user-defined workflow status.
  - Definition cannot be modified.
  - A user cannot intentionally assign records to have a **Recommended** status. This can only be done by the application.

- **Final Status** – *(Reserved) The metadata record is complete*.
  - Be sure to read section *5.4* CONCEPT OF THE FINAL STATUS WORKFLOW STATUS.
  - Definition cannot be modified. It is the same as the **Done** status.
  - Upon installation of the application, the **Final Status** is given the default label of **Done**.
  - In a collection, only records that are XML valid and have a **Final Status** (and whatever label you assign to the **Final Status**) are available to be exported through the application user interface export function.

- **Done** – *(Default) The metadata record is complete*.
  - Upon installation of the application, the **Final Status** is given this default label of **Done**.
  - Use to indicate that a record is ready to be part of the good and working set of records within a collection.
  - Can be re-named.
  - Definition cannot be modified.
  - Cannot be deleted.

- **In Progress** - *(Default) The metadata record is being cataloged*.
  - Use to indicate that a record is undergoing active cataloging. That is the record has generally been considered as a good record to include in the

collection and is undergoing cataloging but it is not ready to be quality assured (QA'ed) or considered done for the collection yet.
    - Can be re-named.
    - Definition can be modified.
    - Can be deleted.

- **Holding** - *(Default) The metadata record has a problem*.
    - Use when a record in the collection has an issue like a broken URL and needs to be temporarily not associated with the good and working set of records in the collection.
    - Use this workflow status over the concept of a deaccessioned workflow status when you expect the record can be fixed and returned to the good set of working records in the collection.
    - Can be re-named (e.g. **Broken**).
    - Definition can be modified.
    - Can be deleted.

5.3 Suggested Additional Workflow Statuses for a Collection

Every collection has a set of workflow statuses associated with it. At a minimum, these are the reserved and default workflow statuses described above. However, it is often beneficial to enhance these workflow statuses with additional statuses appropriate to your collection and its expected workflow. The following is a list of suggested workflow statuses for a more comprehensive collection workflow.

Use these or make up your own statuses. Either way, any statuses you would like to use beyond the reserved and default ones, must be added to each collection that you want them in. And for every status that you add, a definition/description of the status must be provided.

To add workflow statuses in addition to the default and reserved workflow statuses, go to section 5.5 Changing a Collection's Workflow Statuses to enter your selected additional workflow statuses. Do not enter reserved or default workflow statuses.

- **Needs Resource Review** – *The metadata record needs a review for appropriateness*.
    - Use when the content of a resource that is being cataloged may need further review.

- **Needs QA** – *The metadata record needs a final check*.
    - Use to indicate that the record needs a final check of quality assurance before being considered a complete and done record within a collection.

- **Deaccessioned** - *The metadata record has a permanent problem*.
    - Use to indicate that the record cannot be fixed. Often occurs when a record with a workflow status of **Holding** cannot be fixed.

- Use to indicate when a record needs to be removed because the content of the resource has changed (for the worse) or the content no longer falls within your collection policies.
- It is not expected that the record will return to the collection. You could also just delete these kinds of records rather than marking them **Deaccessioned**.
- Use the workflow status of **Deaccessioned** to indicate that the record was once considered a done and complete record of the collection but not anymore.

- **Declined** - *The metadata record is not appropriate for the collection*.
  - Use to indicate that the record is not appropriate for your collection.
  - This is a good status to include when you need to keep track of records *recommended/suggested* to the collection (perhaps through a web form) but that are not *accepted* into the collection.

5.4 Concept of the Final Status Workflow Status

While the ability to have different workflow statuses across collections is beneficial, it poses a challenge to the internal workings of the application to know what records are in a final or done state. That is, are **Done** records or **Published** records or some other workflow status the final state of a record?

To solve this issue, the application uses the concept of a **Final Status** workflow status for every collection. This is a special status that says use one of the workflow statuses associated with a collection as a final or done status. Upon installation of the application, the **Final Status** is given the default label of **Done**, but you may choose a different label. Go to section 5.5 Changing a Collection's Workflow Statuses to enter your own desired label for the **Final Status** workflow status.

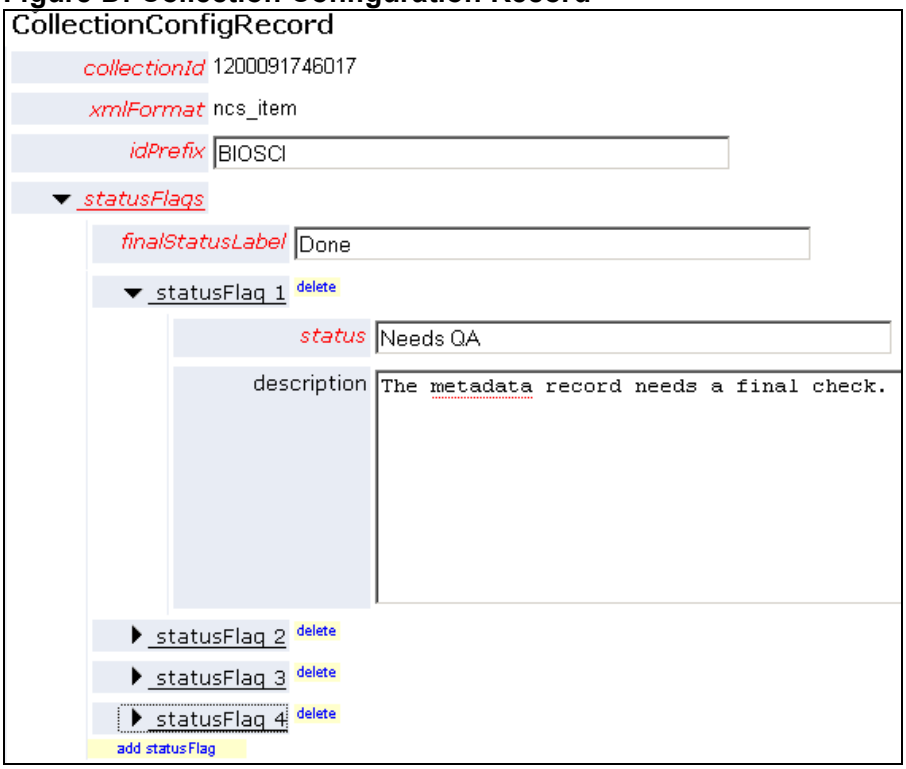
5.5 Changing a Collection's Workflow Statuses

Once a set of workflow statuses and their definitions have been determined, associate these with a collection. You must be a user with a role of **Administrator** to change collection workflow statuses in the process described below.

1. Chose ***Settings*** from the main menu and then click ***Collection Settings***.
2. Click on the desired collection name. This will take you to a view of the collection configuration record.
3. Click the ***Edit Settings*** button.
4. Click the ***statusFlags*** entry (see Figure D below).
5. Enter a ***finalStatusLabel*** (it should be either a default workflow status like **Done** or your own workflow status label).
6. Click on add ***statusFlag***. A status and description field will now display.
7. There may already be an **In Progress** and **Holding** workflow status present. Change these if you want.
8. Enter your own desired workflow status in the ***status*** field. Enter the definition of the workflow status in the ***description*** field.

9. Repeat steps 6 and 8 for as many workflow statuses you would like to enter.
10. Click the ***Save*** button.
11. If an authority error occurs, it means you did not complete the collection authority setup previously. Do so now by choosing either the **ndr** or **dcs**. For an explanation, go to section 3.2 COLLECTION AUTHORITY.
12. Click the ***Save*** button and then click the ***Exit Editor*** button.

Figure D: Collection Configuration Record



5.6 CHANGING A RECORD'S WORKFLOW STATUS

Any user can change the status of a record in a collection in which they have access permissions. To assign or change the workflow status of an individual record, do the following:

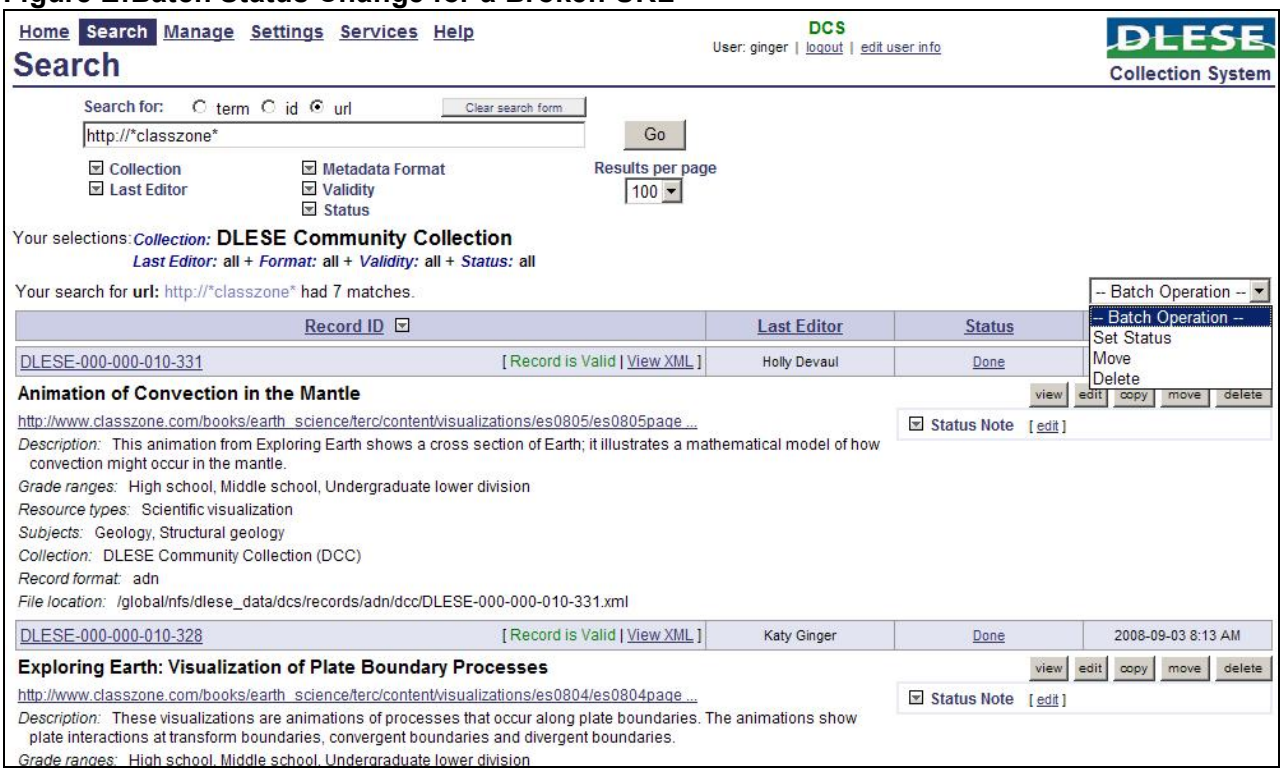
1. Be in search or view mode (not edit record mode) and have the desired record on the screen.
2. Locate the ***Status Note*** drop down link (to the right of the title and URL).
3. Click the ***edit*** link next to the ***Status Note***.
4. In the ***Status*** dialog box, select a status from the drop-down list.
5. In the ***Status Note*** dialog box, enter a note explaining why the record has the status it has. For example, if assigning a status of ***Needs Resource Review*** you might enter a note like: 'Investigate to see if the resource has embedded educational standards.'
6. Click the ***Submit*** button (the status history of the record is presented below the ***Submit*** and ***Cancel*** buttons).

5.7 CHANGE THE WORKFLOW STATUS OF A BATCH OF RECORDS

Changing the workflow status of a batch of records is a very convenient to affect change across many records. This is often required when many records are affected by the same problem such as a temporary or permanent broken URL. Because you'd probably want to suppress the exposure of these broken URLs within your collection, you could change the workflow status of these records from say, **Done** to **Holding** to **Deaccessioned**. The steps below illustrate how to accomplish a batch status change for an example broken URL of http://www.classzone.com.

1. Be on the ***Search*** page and conduct a search (e.g. find the records with the broken domain of www.classzone.com) to bring up the desired records. See Figure E for this search. Notice that the search box has http://*classzone* and the URL toggle switch selected. This returned 7 results. The first two results are displayed in Figure C with a status of **Done**.
2. On the right side of the search results, click on the ***Batch Operations*** drop-down. It is open in Figure E.
3. Click ***Set Status***. This will take you to the ***Batch Status Change*** page.
4. Set the ***Status*** you would like to change the records to. Add a ***Status Note*** if appropriate.
5. If you would like to remove any records from the batch status change, click on the record for which you do not want the batch status change to affect and then click ***Remove Selected Items from List*** button.
6. Click the ***Submit*** button when you are ready to change the workflow status of the remaining records.

Figure E:Batch Status Change for a Broken URL



5.8 Changing or Deleting an Existing Collection Workflow Status

If you need to change or delete an existing workflow status from your collection, you can do so. This is not a recommended practice because it can be confusing to users of the application. This is because the old changed or deleted workflow status remains attached to records carrying that workflow status (until those records are changed to a different workflow status). The old (changed or deleted) workflow status can still be searched on because it is attached to the records that still have it as a workflow status. The old workflow status can no longer be assigned to any records when editing the workflow status of a record. To change or delete an existing collection workflow status, go to section 5.5 Changing a Collection's Workflow Statuses.

6.0 Metadata Frameworks

6.1 Supported Metadata Frameworks

The application's Metadata Editor allows editing of any supported metadata framework defined by an XML schema. The application supports the following metadata frameworks:

1. **Ncs_item** – for cataloging educational resources (classroom activities, lesson plans, Earth datasets, visualizations, reports, theses, journals, audio files, video), news and opportunities. This metadata format is converted to NSDL_DC (the NSDL metadata framework) when metadata records are saved to the NDR. Use this framework if you

will be sharing your metadata with NSDL. See section 6.3 NCS_ITEM METADATA FRAMEWORK for more information.

2. **dlese_anno** - for cataloging annotations (educational standards, comments, reviews, formal evaluations) that are to be associated with existing metadata records. Records created in this metadata framework are not available for sharing directly with NSDL. Further processing is required.

3. **adn** – for cataloging educational resources (classroom activities, lesson plans, Earth datasets, visualizations, reports, theses, journals, audio files, video). This framework is a bit more extensive than ncs_item above because it has specific fields for geospatial, temporal and place name information as well as more educational fields. This metadata format is converted to NSDL_DC (the NSDL metadata framework) when metadata records are saved to the NDR. Use this framework if you would like to contribute to the Digital Library for Earth System Educaiton (DLESE) or want more specific educational, geospatial or temporal fields. See section 6.4 ADN METADATA FRAMEWORK for more information.

4. **DCS_collect** - for cataloging collection-level information like collection title, collection contacts metadata sharing information. NSDL uses this framework to manage its collections. If you need to manage collections that are not created using this application, then you could use this framework to manage collection-level metadata. Otherwise this metadata framework is generally not used.

If one of the above metadata frameworks does not meet your needs, you might be able to use other metadata frameworks that are partially supported (METS, MARC, DC). Please contact NSDL: http://nsdl.org/about/contactus/. Or you may choose to construct your own metadata framework and load it into the application. See section 15.0 METADATA FRAMEWORK CUSTOMIZATIONS.

6.2 CONTROLLED VOCABULARIES

Some of the supported metadata frameworks contain controlled vocabularies. These controlled vocabularies cannot be changed. You must make your own metadata framework to change existing controlled vocabularies. See section 15.0 METADATA FRAMEWORK CUSTOMIZATIONS.

6.3 NCS_ITEM METADATA FRAMEWORK

Since the ncs_item metadata framework catalogs a wide variety of resources, it has been divided into several major categories as depicted in the table below. Within each category, there are several metadata fields that support the purpose of each category.

Table 2: NCS_ITEM Metadata Categories and Fields

Category	*Category Purpose*	*Metadata Fields within Category*
General	General characteristics of the resource that provides a basic	Title, description, alternative (title), url, recordID number,

	level of description in order to facilitate search and discovery of the resource.	subject (keywords), language (of the resource), tableOfContents, bibliographicCitation, abstract
Educational	Characteristics pertaining to the classroom or educational use of the resource.	EducationLevel, types (resource), audience, educational standards (includes ASN ids), mediators, interactivityLevel, accessiblity, interactivityType, instructionalMethods
Contributor Information	Information about the creators/publishers of the intellectual content of a resource	Contributors, creators, publishers
Rights	Right information about a resource that may include intellectual property, copyright and terms of use	Rights, rightsHolders, accessRights, license, provenance, accrualMethod, accrualPeriodicity, accrualPolicy
Technical	Information about making the resource function.	Mime types, format, medium, extent
Relations	Information about related resources (for example: a related resources required by the current resource, a resource the current resource references or is referenced by)	Relations, sources, hasPart, isPart of, isReferencedBy, references, isRequiredBy, requires, isVersionOf, hasVersion, isFormatOf, hasFormat, isRplacedBy, replaces
Dates	A point or period of time associated with an event in the lifecycle of the resource.	Date, created, available, issues, modified, valid, dateAccepted, dateCopyrighte, dateSubmitted, temporal, coverage
Coverages	A resource's spatial/geographic and/or temporal coverage like named places (countries, cities, etc.) or time periods (epochs, date ranges, etc.).	Coverage, box (northLimit, southLimit, westLimit, eastLimit, name, upLimit, downLimit, zUnits), point, spatial, temporal

When you use the **Metadata Editor** (section 7.1) to catalog an ncs_item metadata record, the user interface is divided into the above categories. So to enter the field of resource type, one clicks on the **Educational** category to get to that metadata field.

In terms of **required metadata** to make a catalog record, the **ncs_item** metadata framework requires two fields to produce a catalog record that has valid XML (and works in the NDR). These are:

- **General**: url
- **General**: recordID (application makes by default)

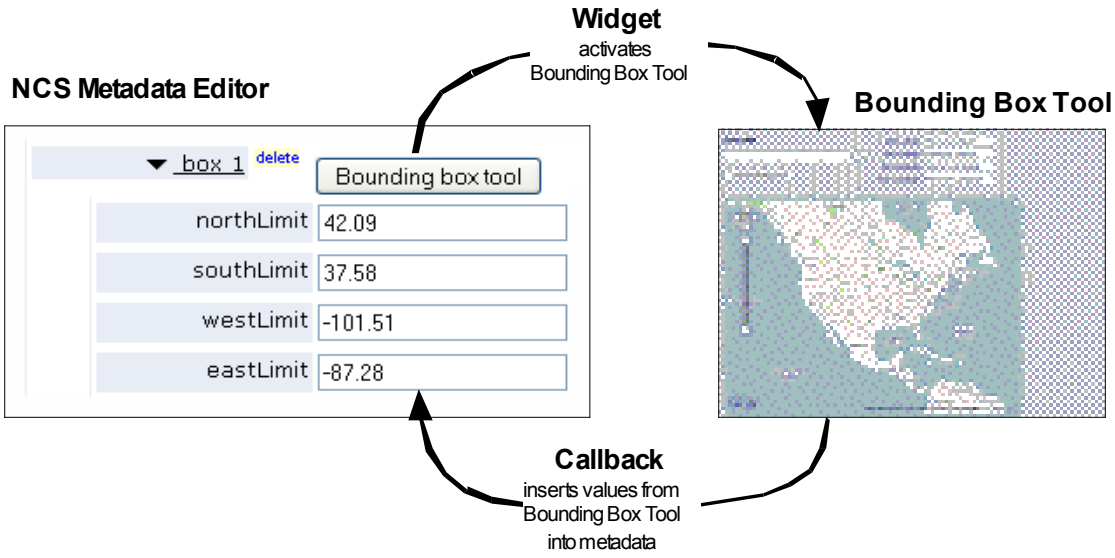
In terms of **required metadata for resource description** and search and discovery within digital libraries, the ncs_item metadata framework requires the following fields:

- **General**: title, description, subject, language
- **Educational**: educationLevels, types, audiences
- **Contributor Info**: choose either contributors, creators or publishers
- **Rights**: rights or accessRights
- **Technical**: mimeTypes

All other metadata is optional and can be used to provide better description of the resource you are cataloging.

To support entry of geospatial information, the ncs_item metadata framework uses the Bounding Box Tool (Figure F) to gather latitude and longitude information. This tool is **not** automatically enabled. To enable it, see section 2.8 STEP 8: CONFIGURE APPLICATION TO USE BOUNDING BOX TOOL (OPTIONAL).

Figure F: Bounding Box Tool for ncs_item



6.4 ADN METADATA FRAMEWORK

The ADN metadata framework is similar to but more extensive than the ncs_item metadata framework. The ADN metadata framework documentation is available at: http://www.dlese.org/Metadata/adn-item/index.php.

6.5 DLESE_ANNO METADATA FRAMEWORK

The application offers access to the Digital Library for Earth System Education (DLESE) annotation metadata framework. In the application, this annotation framework is listed as dlese_anno. The framework is the mechanism for associating additional content, metadata, educational standards or ratings with a resource such as to:

1. Capture comments, reviews, contextual information or feedback when such information cannot be put in the original metadata. Includes annotator information.
2. Capture metadata that is not available in the original metadata (e.g. DLESE.org uses it to capture state educational standards).
3. Connect an annotation or comment to a specific web page (URL) of a resource.
4. Provide a star rating of a resource.

Table 3: dlese_anno Metadata Categories and Fields

Category	*Category Purpose*	*Metadata Fields within Category*
Service	Administrative information about the annotation	Name (of the annotation service), pathway (reserved for DLESE Reviewed Collection -- do not use), recordID, record creation date, itemID (i.e. recordID of the resource being annotated)
Annotation	The content and contributor of the annotation.	Title, type, format, status, content, rating, description, url, contributor

In terms of **required metadata** to make a catalog record, the **dlese_anno** metadata framework requires the following fields:

- **Service**: name, recordID, created, itemID
- **Annotation**: type, content (either description, rating or url), contributors (either as person or organization; person and organization also have additional required metadata)

Other metadata is optional and can be used to provide better description of the resource you are cataloging.

More documentation on this metadata framework is available at: http://www.dlese.org/Metadata/annotation/index.php.

Records created in this metadata framework are not available for sharing directly with NSDL. Further processing is required.

7.0 CATALOGING

7.1 METADATA EDITOR

The actual procedure of cataloging records occurs in the application's **Metadata Editor**. To enter the editor to work on a record, see section 3.5 CREATE NEW RECORDS.

Each metadata framework has its own required metadata fields. These required fields are described in section 6.0 METADATA FRAMEWORKS. Within the **Metadata Editor**, these required fields show as red and must be completed with proper data in order for a

record to generate valid XML behind the scenes. Your own collection building activities may require additional metadata to be completed. Be sure your collection building group is in agreement on the fields it is cataloging and how.

7.2 SAVING AND VALIDATING A RECORD

After you have entered metadata, it is important to save the record by clicking the **Save** button. The save may take a minute or so depending if you are using the NDR. After saving, always click on the ***Validate*** button. This will tell you whether the metadata on that page is valid or if you accidentally forgot to complete a field.

7.3 CATALOGING BEST PRACTICES

Some of the metadata frameworks that are supported in the application contain cataloging best practices. Blue hyperlinks labeled 'Best Practices' appear under the metadata field name when there are cataloging best practices available for that field. Best practices provide the following information:

- XPATH of the metadata field name
- Name of the metadata field name
- Definition of the metadata field
- Things to do when cataloging the metadata field
- Things to avoid when cataloging the metadata field
- Possible examples
- The controlled vocabulary for a metadata field, including term definitions

At this time, the ncs_item (essentially the nsdl_dc metadata framework) does not have a complete set of cataloging best practices, but the ADN and dlese_anno frameworks do.

7.4 DUPLICATE AND SIMILAR URLS

The application has the capability to test for duplicate or similar URLs. For the supported metadata frameworks (section 6.1), duplicate URLs within a single collection are not allowed so that you do not catalog the same resource twice. Similar URLs are flagged and the user is asked if they would like to catalog the similar URL. It is highly recommended to keep this setting. If you wish to change it, follow the procedure in section 15.8 DUPLICATE URL CHECK FOR A CUSTOM METADATA FRAMEWORK.

7.5 SPECIAL CHARACTERS

Special characters (like tildas and copyright symbols) may be used in the application as long as they are properly entity or UTF-8 encoded. If a character is not properly entity or UTF-8 encoded, the application will not let you save the metadata record.

Generally when copying from web pages or MS Word documents, the application will be able to encode special characters into UTF-8 automatically. If the record won't save because of a special character, then entity encode it. The best entity encoding method is to use the ampersand, pound sign, number, semicolon text string. For example, the ñ symbol has an entity encoding of ñ. Please see the cataloging best practices in the fields of title, description for further explanation.

7.6 See the XML of a Record

Sometimes it is beneficial to see the XML that is generated behind the scenes when a record is created or edited. You can do this on the ***Search*** page by clicking on the [View XML] hyperlink that is associated with every record near the Last Editor column. Please note that the ncs_item XML does ***not*** look like nsdl_dc XML. nsdl_dc XML is generated later in the process from ncs_item XML.

8.0 Catalog and Tool Training/Help

Users of the application can participate in catalog and tool training sessions. Please contact http://nsdl.org/about/contactus/ about participating and the level of training needed. Also use this contact form for any additional questions. Please note when your collection building group is developing a cadre of catalogers/ editors or other personnel who will be using the tool, this group may also attend training as they are the users who will need it and/or will benefit from it the most.

9.0 Searching

1.1 Search by Keywords, URL or Record ID

The application provides multiple methods of searching, that is by keyword term, URL or record ID number. By default the search is set to keyword terms. To initiate a URL or record ID number specific search, a user must toggle “on” the appropriate radio button above the search entry box.

9.1 Quick Collection Search

On the application ***Home*** page, a user can click the hyperlinked name of a collection and be taken to search results that list all records within that collection.

9.2 Quick Search by Collection, Last Editor, Metadata Format, Validity or Workflow Status

When searching, a user has the option of specifying search criteria. These criteria can be a collection name, last editor, metadata format, XML validity, workflow status or any combination thereof. Just use the application’s drop-down menus on the ***Search*** page.

9.3 Search Tips for Record Ids

Below are examples of searching by record Ids.

- To find the record DLESE-000-000-003-764: enter **DLESE-000-000-003-764** in the search box and toggle the ***id*** button on.
- Using a wildcard with ID numbers: **DLESE-000-000-*12** returns all DLESE IDs that end with 12. Toggle the ***id*** button on.
- Verifying that a word is in a record using its ID number: **DLESE-000-000-000-012 geological** ensures that the word geological appears in record DLESE-000-000-000-012. Toggle the ***term*** button on not the ***id*** button.

9.4 SEARCH TIPS FOR URLS

You can search by URL or website. Like in Google, you can use **site:** in place of http://. (So **site:dlese.org** is the same as **http://dlese.org** OR **http://www.dlese.org**.)

- **http://www.marsquestonline.org/index.html** returns the MarsQuest site itself.
- Wildcards: **http://*marsquestonline*** (or **site:*marsquestonline***) returns all resources with marsquestonline in the URL.
- Qualifiers: **http://*marsquestonline* canyon** (or **site:*marsquestonline* canyon**) returns resources with marsquestonline in the URL **AND** the word canyon in another part of the record.

9.5 WILDCARDING

- Wildcards (indicated by *) are supported in keywords or URLs. Wildcards **cannot** appear at the beginning of a term (*ocean) but can be used at the beginning of a URL (site:*nasa.gov).
- **http://*marsquestonline*** returns all resources with marsquestonline in the URL (whereas **http://www.marsquestonline.org/index.html** just returns the MarsQuest site itself).
- **http://*marsquestonline* canyon** returns resources with marsquestonline in the URL **AND** the word canyon in another part of the record.

9.6 GENERAL SEARCH TIPS

- By default, the application searches for terms you enter in the keyword box within an entire record.
- Spelling matters. If your search does not retrieve any resources, check the spelling or try alternative search terms. In most cases, the application automatically searches for different forms of the keywords. For example, a search for **ocean** returns records with oceans or oceanic; those with the exact match (ocean) are given priority.
- Searches are not case sensitive. That means capital letters and lower case letters are treated the same way.
- You can search using multiple terms and phrases.
- If you use quotes (single or double is fine) around a phrase, only records that have all of the words in order will be returned in your search results.
- You can use wildcarding (indicated by *) to find records that contain term(s) in the resource or URL but are not restricted solely to them.

9.7 USING AND AND OR

When using or qualifying a search with the terms of AND or OR, be sure to capitalize the terms to differentiate them from regular words. Some examples include:

- 2 (or more) words with no qualifier: If you type **marine ocean**, the first results will contain BOTH words (marine AND ocean). The words need not be beside each other in the resource.
- 2 words with OR: If you type **marine OR ocean**, you will get resources with either marine OR ocean.
- 2 words in quotes: If you type **"marine ocean"**, you will get resources that have the exact phrase.

- Excluding terms from search results: You can omit resources that contain a given term by using the "!" symbol directly before a term (no space in between). For example: **ocean !sea** yields resources with the term ocean but not those that contain the term sea.

9.8 USING PARENTHESES

You can group terms using parentheses and make combinations using AND and OR. For example (**ocean wave**) OR (**sea wave**) returns resources that contain both ocean and wave OR both sea and wave in their description.

9.9 SETTING THE NUMBER OF SEARCH RESULTS

A user can set the number of results per page from 10 to all. Just click on the ***Results per Page*** drop-down. For performance reasons with large collection over 500 records, set the result per page to a maximum of 100 rather than all.

10.0 LOGO BRANDING THE APPLICATION

The application allows collection building groups to specify a logo that will appear in the upper right corner of each page of the application. By default, the application loads the National Science Digital Library (NSDL) logo. To change the logo, you will need to modify the context descriptor and therefore need access to the server file system location of where the application is installed. Then follow these steps:

1. Prepare a logo image file. The application logo image is displayed as 135 pixels wide by 35 pixels high. So your image file should be of similar proportions. The file can be in any web-displayable format, such as "gif", "jpg", or "png".
2. Place the logo image file in the "images" directory of the web application (e.g., $TOMCAT/webapps/<appName>/images).
3. Open your context descriptor in $TOMCAT/conf/server.xml
4. Add (or uncomment) the **logo** parameter and set the parameter value to your logo image filename (include the file extension).
5. Restart tomcat. Your custom logo will appear after tomcat is restarted.

11.0 NSDL DATA REPOSITORY (NDR) INTERACTIONS

11.1 WHEN ARE RECORDS WRITTEN TO THE NDR?

All collections that have a collection authority of **ndr** interact with the NDR. The records within such collections are written to the NDR upon create and save operations. However, if you are using an application that creates records with a workflow status of **Recommended**, then these records are saved to the local computer file system only and not the NDR. When a user changes this **Recommended** record to anther workflow status, the record is then written to the NDR regardless of the new workflow status.

11.2 MAKE RECORDS ACCESSIBLE TO NSDL.ORG SEARCH (FOR ITEM-LEVEL RECORDS CREATED IN THE DCS)

Item records within a collection are available to NSDL.org search (generally in about 48-72 hours) if:

- A **Final Status** workflow status has been assigned to the record.
- The record has valid XML.
- The collection builder has contacted NSDL to have an NSDL collection record made and this collection record is complete.
- The NDR knows about the item record through either OAI harvesting or having the application interact directly with the NDR API. If you are using the http://DCS.nsdl.org/mgr (MGR) application instance, this step has been taken care of for you. If you are not using the MGR application instance, please contact NSDL http://nsdl.org/about/contactus/ to request 'DCS application agent handles for your locally installed software'

If a record has a workflow status of **Final Status** or **Done** but is not XML valid, it will never make it into NSDL.org search. Go to section 14.3 XML VALIDATION ERRORS to read how to resolve errors involving **Done** but invalid XML records.

11.3 WHAT HAPPENS WHEN ITEM-LEVEL RECORDS ARE DELETED OR DEACCESSIONED?

When an item-level record has a workflow status besides the **Final Status** (or whatever label like **Done**, **Finished**, that you have given the **Final Status**), it is in the NDR but not available to NSDL.org search. If you delete the record, it is removed from the application, NSDL.org search and the NDR. This excludes the workflow status of **Recommended** as described in section 11.1.

12.0 OAI INFORMATION

12.1 OAI PROVIDER ASSUMPTIONS

The application includes an OAI provider so that metadata records can be shared via the OAI protocol version 2.0. Only an OAI provider is included and not an OAI harvester. Access to the setup of the OAI provider is located under the ***Services*** menu. A user must have a role of **Administrator** to access this area.

Before describing how to setup the OAI provider, it is important to keep the following ideas in mind because these control how the OAI provider acts.

- Each collection is treated as an OAI set. Collections cannot be combined to form aggregate sets.
- Each OAI set (collection), can be enabled or disabled individually from the ***Manage***, ***Manage Collections*** menu. On the Manage Collection table, just toggle the ***Services*** checkbox on or off for OAI set enabling and disabling.
- If the OAI set (collection) is disabled, then the OAI provider provides no matching records.

- Only valid XML records are provided as long as records have a **Final Status** workflow status or whatever label is being used for the final status workflow status (see section 5.4 Concept of the Final Status Workflow Status). If a record has the final workflow status but is XML *invalid*, the record will not be provided. There is no error message that tells you this. You should periodically check that all records with a final workflow status are XML valid. See section 14.3 XML VALIDATION ERRORS.
- The OAI provider provides all metadata in its native metadata format. This is true for supported metadata frameworks and any metadata frameworks users create within the application.
- Supported metadata frameworks are OAI provided in their native format and often provided in nsdl_dc as well (see section 6.1 SUPPORTED METADATA FRAMEWORKS).
- If you have created and added your own metadata framework to the application, your native metadata framework will be OAI provided.
- If you would like to OAI provide your own developed native metadata in another format like nsdl_dc, you need to write and add to the application your own XSLT stylesheet (.xsl file) that transforms your native metadata framework to the desired metadata format (see section 15.7 ADDING AN XSLT STYLESHEET (.XSL FILE) TO TRANSFORM METADATA).
- Collections that use the NDR as authority (storage area) rather than the local file system can be provided using OAI.

12.2 OAI PROVIDER SETUP

To share records using the OAI provider, complete the process below:

1. From the ***Services*** from the menu, select ***OAI Services***.
2. Click on the ***Edit Repository Info*** button.
3. Change the repository name and administrator's email.
4. Add a repository description and namespace (really repository identifier). This is required regardless of what the application indicates.
5. Click the ***Save*** button.
6. Return to ***OAI Services*** page and click on the ***Enable Provider Access*** button.
7. Your application is now an OAI provider. If desired, change the response length of certain OAI protocol verbs by clicking on the ***Edit Response Length*** button. Click the ***Save*** button when done.

12.3 OAI ENABLE/DISABLE

On the ***OAI Services*** page, the ***Enable/Disable Provider Access*** button controls access to the entire provider. If on the ***Manage***, ***Manage Collections*** page you have individual collections enabled with services, these collections will not be provided until the ***Enable/Disable Provider Access*** button is enabled on the ***OAI Services*** page and vice versus with disable.

12.4 OAI PROVIDER INFORMATION

Once the OAI provider is setup and enabled, you need to give potential OAI harvesters the proper information to harvest your repository. At a minimum, you must give potential harvesters your **baseURL**. And if you would like certain sets or metadata formats to be

harvested, then give potential harvesters **set spec** and **metadata format** information as well.

The application includes this OAI information. The notes below describe where to find this information in the application. Please note that a user must have a role of **Administrator** to access this information.

- **Base URL** and **repository name**: Base URL is assigned by the system and you will have completed the repository name when editing the repository information in section 12.2 OAI PROVIDER SETUP. From the ***Services*** menu, select ***OAI Services***. Look at the ***Repository Information*** table entries of base URL and repository name.
- **Set spec** and **metadata format**: From the ***Settings*** menu, select ***Collection Settings***. Set specs are the values listed in the **Key** column. Metadata formats are the values listed in the **Format** column.

13.0 WEB SERVICES

Web services provide programmatic access to records stored in the application's repository. There are two web services:

- **Search Web Service** -
  - Used to find records within the application repository. Use the **Repository Web Service** below to add records programmatically to the application repository.
  - Supports textual searching over the metadata, searching by date and field, sorting by date and field, discovery of controlled vocabularies of certain metadata frameworks and checking the existence of a URL.
  - Documentation for the query language is available within the application (from the **Services** menu, select **Web Services** and click the **Search Web Search specification**) or go directly to http://nsdl.org/nsdl_dds/services/ddsws1-1/index.jsp and http://wiki.nsdl.org/index.php/Community:Search_API_FAQ.
- **Repository Web Service** -
  - Used to add records programmatically to the application repository. Use the **Search Web Service** above to find records in the application repository.
  - Use to insert records into the application repository by third-party applications (e.g. web forms that recommend resources). A clients Internet Protocol (IP) address must be added to the Configure Authorized Client IPs section of the application. From the ***Services*** menu, select ***Web Services***. Then enter the IP address of the client you are authorizing to access the repository. The click ***Set IP Addresses***.
  - Documentation is available within the application: From the ***Services*** menu, select ***Web Services*** and click the ***Repository Web Service specification*** link.

14.0 OPERATIONAL ERRORS

14.1 OPERATIONAL ERRORS TO CHECK FOR

In the daily operation of the application, there are several operational errors that can naturally occur. These are user caused and therefore need to be user corrected. The most common operational errors that need corrections are described below.

14.2 NAVIGATION ERRORS

When navigating through the application, it is very important to use the navigation provided within the tool rather than the forward and back buttons within your browser. If you receive a page expired error, it is mostly likely caused by using the browser buttons rather than the application's navigation. You may need to start your task again. There are some pages within the application (e.g. when viewing the XML that is generated for each metadata record) that do not provide navigation. Use your browser buttons in these cases. They are generally obvious.

14.3 XML VALIDATION ERRORS

This is an important error that requires immediate attention. The error indicates that the XML of a record does not conform to the XML schema of the particular metadata framework. This means the record is either missing required metadata, controlled vocabulary information or may even be incorrect in structure (happens if the metadata framework changed). Every record is either XML **valid** or XML **invalid**. This is indicated in ***Search*** results or on the ***View Record*** page with the hyperlinks of [Record is Valid] [Validate Record]).

Any user can address validation errors as long as they have access to the particular collection in which the invalid records reside. It is especially important to fix validation errors. The scenarios below describe the importance of validation and how to make fixes to records:

- **Why validation errors matter -** Any records with invalid XML are not accessible to other application services like OAI, web services, exporting, or transforming to other desired metadata frameworks like nsdl_dc. This means the records will not appear in desired collection discovery mechanisms outside the application. Even if a record has a **Done** or **Final Status** workflow status, it still will not be accessible outside the application if it has invalid XML.

- **How to discover validation errors for all records in collection -** go to the ***Manage*** page and select ***Manage Collections***. Click the ***Validate*** hyperlink in the Validate Record column for the desired collection. Click the **Ignore cached validation results** box to toggle it on for the most accurate validation results. Then click the ***Validate Collection*** button. On the Validate Collection page, refresh the validation report and wait for it to finish. Then click on the ***See Report*** in the

message box to review any errors. A user must have a role of **Manager** or **Administrator** to complete this task.

- **How to discover validation errors on Final Status workflow status records within a collection -** go to the ***Manage*** page and select ***Manage Collections***. If there are any collections that have a red number in the ***Final Status and Invalid*** column, these need to be addressed until the number reads 0. This occurs when a metadata record has a Final Status (most likely **Done**) and is invalid.

- **How to discover validation errors on individual records -** On the **Search** or **View Record** pages, the [Validate Record] hyperlink rather than [Record is Valid] hyperlink is displayed for those records that contain invalid XML. Or conduct a search using the Validity drop-down menu to find all non-valid records. Click on the [Validate record] hyperlink to see the XML errors within an invalid record.

14.4 NDR SYNC ERRORS

If a collection is using the NDR as its storage location, communications between the application and the NDR may occasionally be disrupted. When this happens you will see a red ***Sync Error*** on individual records on the ***Search*** results page. To address these errors, a user must have a user role of **Manager** or **Administrator**.

- **Why sync errors matter** - It means the record did not make it successfully into the NDR and it is not available to services external to the application, nor NDR services.

- **How to test for the sync error** - It is a good idea to test to see if this error is occurring rather than discover it by happenstance.
  1. Choose ***NDR*** from the menu and select ***Admin***.
  2. If there are any collections that have a red number in the Sync column, these need be addressed until the number reads 0.
  3. Click ***Sync*** in the Sync to NDR column. This should take care of the problem. If not, contact, http://nsdl.org/about/contactus/.

14.5 INDEX ERRORS

The most common causes for indexing errors are 1) ingesting records from outside the application into the application and 2) initial installation and startup of the tool. Either way, the ***Home*** page displays a message to the effect that the ABC Collection or Sample Collection must be re-indexed before it can be accessed. To address these errors, a user must have a user role of **Manager** or **Administrator**. To remove the error message, do the following:

1. Chose ***Manage*** from the main menu and then select ***Mange Collections***.
2. In the table row for the Sample Collection or whichever collection is giving you the error, select the ***Index*** link.
3. Wait for indexing to complete. This may take several minutes depending on the size of the collection. Then click OK to remove indexing messages.

14.6 Locked Records

When a record is created or opened for editing, it becomes locked so that other users cannot work on the same record. However, records can also become locked if your browser crashes while you have a record open or if you walk away from working on a record without exiting the ***Metadata Editor*** properly. If you try to access such a record, you will receive the locked record error message. Locked records are not available for editing. To address these errors, a user must have a user role of **Manager** or **Administrator**. To unlock a record, complete the steps below:

1. Go to the ***Manage*** page and select ***Sessions***.
2. The table lists which records are locked. Select the desired record. Be sure you have the correct record so that you do not interrupt someone's session. The click ***Unlock Selected Records***.

15.0 Metadata Framework Customizations

15.1 What Can Be Customized?

Collection builders may use the application to support their own:

- Metadata frameworks
- Controlled vocabularies
- Layout of vocabularies within the application's **Metadata Editor** (section 7.1)
- Cataloging best practices

The application does not allow you to provide different controlled vocabularies or cataloging best practices for existing supported metadata frameworks listed in section 6.0.

Because explicit instructions for completing customizations are incomplete, collection builders should contact NSDL http://nsdl.org/about/contactus/ for the latest information and updates.

Please note that NSDL's support of customization is outside regularly supported application functions and therefore customization help, and associated costs, is negotiated with each customer, client or collection building group.

15.2 Adding a Custom Metadata Framework

To use your own metadata framework within the application:

1. Make an XML schema or schemas that define it. Please note that NSDL provides training on designing XML schemas to support custom metadata frameworks within the application.
2. Decide on a short label for the name of the framework. This must be unique within the application. (e.g. dlese_anno for the DLESE annotation metadata).
3. Put the XML schemas at a URL or within the application. That is, use one of the following examples to decide where you store the schemas and determine a correct path to them:

    a. Absolute path for file-based Microsoft Windows: file:///C:/..../schema.xsd (spaces are okay in the path)
    b. Relative to WEB-INF: WEB-INF/metadata-frameworks/dlese_anno/annotation.xsd
    c. Absolute URL: http://www.dlese.org/Metadata/annotation/1.0/annotation.xsd
4. Make a framework configuration file with parameters below. See Appendix 3: FRAMEWORK CONFIGURATION PARAMETERS for a description of each parameter. See a sample frameworks configuration file at: $TOMCAT/webapps/<appName>/docs/templates/custom_framework.xml.
    a. xmlFormat
    b. schemaInfo
    c. schemaURI
    d. rootElementName
5. In the directory path you defined for the **dcsConfig** context descriptor parameter (most likely in server.xml), there is a frameworks subdirectory. Name this newly created configuration file with the short label you gave the framework (e.g. dlese_anno.xml) and place it in the frameworks directory. IMPORTANT: Do not rename or edit the framework_config.xml file.
6. To load the framework in the application, you do not need to start/stop your servlet container (e.g. Tomcat). Click on ***Settings*** and select ***Metadata Frameworks***. If you are already there, click F5 to refresh. You should then see your new framework under the Available Frameworks heading. Click load. Your framework is ready for use.
7. If you change the XML schemas supporting your metadata framework, click the ***Reload*** link on this page to make the changes active.
8. If you need to modify the framework configuration file you made in step 5, you can now edit this file directly in the application interface by clicking on ***Settings*** and selecting ***Metadata Frameworks*** and clicking the ***Edit*** link for the framework.

This section is required to make the application work with your own metadata framework. Please note that the layout of the user interface in the **Metadata Editor** is based on the structure of your XML schema. Adjust your XML schemas to change the layout of the user interface. NSDL provides training on designing XML schemas for use in the application. Refine the layout of any controlled vocabularies within your metadata framework using section 15.4 CUSTOMIZING USER INTERFACE LAYOUT OF CONTROLLED VOCABULARIES.

15.3 CUSTOMIZING CONTROLLED VOCABULARIES

To support a controlled vocabulary for a particular metadata field within your metadata framework, use the XML schema construct of restriction and enumeration to list the terms that are valid. The **Metadata Editor** in the application will automatically pick these up when the XML schemas for your metadata framework are loaded into the application. This section is not required to make the application work with your own metadata framework.

15.4 Customizing User Interface Layout of Controlled Vocabularies

Because controlled vocabularies can be long or hierarchal, it is important to group the layout of such terms so that catalogers can easily accomplish their tasks. The application allows for customizing the layout of the user interface for such controlled vocabularies.

This is done by creating a set of XML files called "groups" files and placing them in the application or at a URL. An existing "groups" file used by the National Science Digital Library (NSDL) is available at: http://ns.nsdl.org/DCS/ncs_item/1.02/groups/nsdlEdLevel-DCS-groups-cataloger-en-us.xml.

Once you have created as many "groups" files as you need, create a "groups listing" file that can be used by the application to know which "groups" files you want processed. An existing "groups listing" file used by the NSDL is available at: http://ns.nsdl.org/DCS/ncs_item/1.02/build/groups-list.xml.

To have the application recognize your "groups" files, modify the framework configuration record by adding a reference to the "groups listing" file (i.e. a reference to groups-list.xml). Please note the procedure below works only for loaded metadata frameworks and applies on a per framework basis only and not a per collection basis. Add the "groups" reference by doing the following:

1. On the ***Settings*** menu, select ***Metadata Frameworks***.
2. Click the ***Edit*** link for the framework.
3. For the **vocabLayoutURI** parameter, enter a path to the "groups listing" file. Use the path examples under section 15.2 as a guide.

Please note that this section is not required to make the application work with your own metadata framework but rather it supports users and catalogers in accomplishing their tasks more easily, effectively and efficiently.

15.5 Customizing Cataloging Best Practices and Vocabulary Definitions

Cataloging best practices support catalogers in:

- selecting the appropriate controlled vocabulary terms for a metadata field
- providing information on how to complete a metadata field
- providing definitions for controlled vocabulary terms

The application supports the ability to add such cataloging best practices to each field within a metadata framework. This is done through a set of XML files called "fields" files and placing them in the application or at a URL. An existing "fields" file used by the National Science Digital Library (NSDL) is available at: http://ns.nsdl.org/DCS/ncs_item/1.02/fields/nsdlEdLevel-DCS-fields-en-us.xml.

Once you have created as many "fields" files as you need, create a "fields listing" file that can be used by the application to know which "fields" files you want processed. An

existing “fields listing” file used by the NSDL is available at: http://ns.nsdl.org/DCS/ncs_item/1.02/build/fields-list.xml.
To have the application recognize your “fields” files, modify the framework configuration record by adding a reference to the “fields listing” file (i.e. a reference to fields-list.xml). Please note the procedure below works only for loaded metadata frameworks and applies on a per framework basis only and not a per collection basis. Add the “fields” reference by doing the following:

1. On the ***Settings*** menu, select ***Metadata Frameworks***.
2. Click the ***Edit*** link for the framework.
3. For the **fieldInfoURI** parameter, enter a path to the “fields listing” file. Use the path examples under section 15.2 as a guide.

Please note that this section is not required to make the application work with your own metadata framework but rather it supports users and catalogers in accomplishing their tasks more easily, effectively and efficiently.

15.6 CUSTOMIZING SEARCH RESULTS DISPLAY

The metadata fields displayed in the application’s search results are adjustable. For simplicity, it is best to leave the display as is for the existing supported metadata frameworks and adjust the display for any custom developed metadata framework. For the information displayed, there is a custom part and a standard part as shown below.

The standard part is displayed for all records for all metadata frameworks (custom and existing supported frameworks). The standard part includes the collection name, metadata record format (metadata framework being used) the file location and if using the NDR, an NDR handle.

Figure G: Customizing Search Results

The custom part of the display is defined on a per-framework basis, and is done by creating a JSP file named in a way that it can be automatically found by the application. Custom (or framework-specific) display JSP files should be placed in the $TOMCAT/<appName>/browse/searching/synopses directory and should be named by appending “_search_info.jsp” to the framework’s XML format (see section 15.2 part 4a.).

For example, the display page for the “ncs_item” format is named, “ncs_item_search_info.jsp”. The JSP page works by plucking values from the metadata record using xpaths and then displaying the values as HTML.

To get you started with this customization, the application includes an inactivated metadata framework called custom_framework and a corresponding example JSP page called custom_framework_search_info.jsp. To activate these and to play/explore JSP customizations, do the following:

1. Activate the custom_framework metadata framework by copying the custom_framework.xml (framework configuration) from $TOMCAT/webapps/<appName>/docs/templates to $dcsConf/frameworks directory.
2. Go to the ***Settings*** menu and select ***Metadata Frameworks***.
3. Under the available frameworks heading, load the custom_framework metadata framework.
4. Go the ***Manage*** menu and select ***Manage Collections***. Create a collection using the custom_framework metadata framework.
5. Go to the ***Home*** page menu and create an item record in this new collection.
6. Go back to the ***Home*** page menu and click the new collection name. You will see the search results display formatted using the custom_framework_search_info.jsp JSP page from the $TOMCAT/<appName>/browse/searching/synopses directory.
7. View and edit the JSP file to understand how it works. (Note: you may have to hit the refresh button on your browser to see changes to the JSP file).
8. If you created your own metadata framework, apply this procedure to customize the search page.

WARNING: When re-installing the application for an update or other reason, custom JSP pages should be saved off to a temporary location and then added back after the re-install. Otherwise any custom JSP page you develop will be removed when the application is re-installed.

This section is not required to make the application work with your own metadata framework but rather it provides a search interface with meaningful results.

15.7 Adding an XSLT Stylesheet (.XSL File) to Transform Metadata

If you added your own native metadata framework and would like the OAI provider included with the application to provide such metadata in another format like nsdl_dc, then you must add an .xsl file to make this happen. You are responsible for writing the .xsl. To add the .xsl to the application:

1. Place the .xsl file in the $TOMCAT/webapps/<appName>/WEB-INF/xsl_files directory.
2. This .xsl file is called a format converter and a context-param for it must be added to the web.xml file in $TOMCAT/webapps/<appName>/WEB-INF/
3. Make your entry look similar to this example:

```
<!-- To configure a format converter that uses an XSL stylesheet, the param-
name must begin with the string 'xslconverter' and must be unique. The param-
value must be of the form 'xslfile.xsl|[from format]|[to format]'.-->
```

```
<context-param>
   <param-name>xslconverter - Converts from dlese_anno to oai_dc</param-name>
   <param-value>dlese_anno-v1.0.00-to-oai_dc.xsl|dlese_anno|oai_dc</param-
value>
</context-param>
```

WARNING: When re-installing the application for an update or other reason, custom XSL files should be saved off to a temporary location and then added back after the re-install. Otherwise any custom XSL files you develop will be removed when the application is re-installed.

15.8 Duplicate URL Check for a Custom Metadata framework

If you would like to prevent duplicate URLs from being cataloged within collections using your custom metadata framework, you can do this by entering a value for the **valueType** parameter in the framework configuration file.

Please note this procedure works only for loaded metadata frameworks and applies on a per framework basis only not a per collection basis.

1. On the ***Settings*** menu, select ***Metadata Frameworks***.
2. Click the ***Edit*** link for the framework.
3. Enter the value 'uniqueUrl' for the **valueType** parameter. See Appendix 3: Framework Configuration Parameters for more details.

15.9 Enabling the CAT Service for a Custom Metadata Framework

The CAT service, described in section 2.7, can also be activated for your own custom metadata framework. There are several important decisions and products that you must obtain before enabling the CAT service:

- **Product 1**: Obtain the standards documents from ASN as described in section 2.8.
- **Product 2**. Obtain a suggestion service plug-in specific to your custom metadata framework. Contact NSDL at http://nsdl.org/about/contactus/ and ask for plug-in to be developed. You will need to provide access to or the files of your custom metadata framework.
- **Decision 1**: Decide if you want to access standards information using a directory of files, through a defined data type within your custom metadata framework schemas or a single file.
- **Decision 2**: Decide which content of the ASN standards documents you want to access. The more files the service traverses, the slower it will be. For example, you may decide you only want to deal with the standards documents that deal with science.

Please note that the file at:
$TOMCAT/webapps/<appName>/docs/templates/suggestion_service_config.xml
illustrates three possible scenarios of decision 1, that is standards are accessed by using a:

- Directory of file and the Metadata Editor displays the NSES science standards as the first item in the list of available standards. Other standards are selectable but the NSES just come up first.
- Datatype in a custom metadata framework
- Single file

After you have obtained the products and made the decisions above, enable the CAT service:

1. Add a new config element to the suggestion_service_config.xml by copying the appropriate existing set of elements depending if you decided on a standards directory, schema datatype or single file.
2. Indicate the **xmlFormat**. Use the value of 4a under section 15.2.
3. Indicate the **xpath** to the XML element within your custom metadata framework for which the service is to make suggestions.
4. In the sub-element of stdSource, enter an appropriate value: an absolute path to a directory, a schema datatype or single file.
5. Save the file. A good location to put it is in the directory defined by the **dcsConfig** context descriptor parameter.
6. In the appropriate context descriptor file, (mostly likely server.xml) set the **standardsSuggestionServiceConfig** parameter to a value that points to the modified suggestionServiceConfig.xml file. NOTE: If a value is set for the **standardsSuggesionServiceConfig** parameter, then the **asnStandardsDocument** parameter is ignored.

16.0 NSDL COLLECTION RECORDS (SPECIAL SECTION)

This section is **only** for application users who control collections that appear at NSDL.org and need to manage the NSDL collection records directly. This uses the MGR application instance (DCS) installed at Cornell University.

16.1 CREATING NSDL COLLECTION RECORDS IN THE MGR APPLICATION INSTANCE

NSDL collection records must be created using the **Submit a Collection** form at http://recommend.nsdl.org/collection.php. Form usage ensures that a default brand image is created if no brand image is provided. And it ensures the creation of a metadata record that is XML valid.

If you do not use the form, you will have to upload (via sftp) an image to NSDL.org and place it in the proper directory on the computer server. If you do not know how to do this, be sure to use the **Submit a Collection** form when creating an NSDSL collection metadata record. Records are created immediately upon form submittal. You can go to the MGR instance of the application and start working with the record. It will land in the MGR application instance with a workflow status of **Recommended**.

16.2 DCS_COLLECT REQUIRED METADATA

Table 4: DCS_collect Metadata Categories and Fields

Category	*Category Purpose*	*Metadata Fields within Category*
General	General characteristics of the resource that provides a basic level of description in order to facilitate search and discovery of the resource.	Title, description, alternative (title), url, recordID number, subject (keywords), language (of the resource), tableOfContents, bibliographicCitation, abstract
Educational	Characteristics pertaining to the classroom or educational use of the resource.	EducationLevel, types (resource), audience, educational standards (includes ASN ids), mediators, interactivityLevel, accessiblity, interactivityType, instructionalMethods
Contributor Information	Information about the creators/publishers of the intellectual content of a resource	Contributors, creators, publishers
Rights	Right information about a resource that may include intellectual property, copyright and terms of use	Rights, rightsHolders, accessRights, license, provenance, accrualMethod, accrualPeriodicity, accrualPolicy
Technical	Information about making the resource function.	Mime types, format, medium, extent
Relations	Information about related resources (for example: a related resources required by the current resource, a resource the current resource references or is referenced by)	Relations, sources, hasPart, isPart of, isReferencedBy, references, isRequiredBy, requires, isVersionOf, hasVersion, isFormatOf, hasFormat, isRplacedBy, replaces
Dates	A point or period of time associated with an event in the lifecycle of the resource.	Date, created, available, issues, modified, valid, dateAccepted, dateCopyrighte, dateSubmitted, temporal, coverage
Coverages	A resource's spatial/geographic and/or temporal coverage like named places (countries, cities, etc.) or time periods (epochs, date ranges, etc.).	Coverage, box, point, spatial, temporal
Collection	Characteristics of the collection including collection purpose (educational/research), collection topic, collection contact, NSDL Pathway designation, OAI	dateTime, brandURL, imageHeight, imageWidth, altText, OAIvisibility, pathway, pathways.name, collectionSubject,

	harvesting information, brand image and an indication of whether NSDL can share the collection via OAI.	collectionPurpose, contacts, oai.baseURL, format, repositoryName, protocol, frequency, OAI description, collection notes, setSpec

For a collection to be loaded in the NDR, the **required metadata** are:

- **General**: url
- **General**: recordID (application makes by default)
- **General**: title, description, subject, language
- **Educational**: educationLevels, types, audiences
- **Contributor Info**: choose either contributors, creators or publishers
- **Rights**: rights or accessRights
- **Technical**: mimeTypes
- **Collection**: dateTime, brandURL, imageHeight, imageWidth, altText, contacts (all completed if the **Submit a Collection** form is used to create the collection)
- **Collection.contacts**: who is responsible party and OAI admin email addresses
- **Collection.pathway**: true or false
- **Collection.pathways.name**: choose the Pathway list
- **Collection purposes**: education, research (for browsing)
- **Collection subjects**: choose one or more topic areas (for browsing)
- **Collection OAI visibility**: controls metadata going to the public NSDL OAI provider
- **Collection.ingest**: (if a collection uses OAI)
- **Collection.ingest.OAI base URL**: the URL to the OAI provider of the collection
    - **format**: metadata format, typically oai_dc or nsdl_dc
    - **frequency**: a decimal number representing the number of **months** between harvests (0 for no harvests, 1 for once a month, 2 for once every two months, etc. etc.)
    - **respositoryName, protocol, setSpec**

16.3 NSDL COLLECTION RECORDS AND THE NDR

All records are written to the NDR upon creation and save operations except for those new collection records that land in the DCS through **Submit a Collection**. These collection records are given a workflow status of **Recommended** are only saved to the computer server file system. When the record is acted upon to change its status to something else other than **Recommended**, then the NDR becomes involved.

16.4 NSDL OAI HARVESTING FOR NEWLY CREATED COLLECTIONS

In order for the NSDL OAI Harvest Manager to see a new NSDL collection record, change the workflow status of the collection record to **Done**. You then have several options:

- Kick off a test harvest which does everything but ingest. The results will go to the MR-Ingest email list.

- Wait for a production harvest to kick off based on the information supplied in the collection record. This may take up to a month or more. The results will go to the MR-Ingest email list.
- Kick off a production harvest based off the information supplied in the collection record. The results will go to the MR-Ingest email list.

If something does not look good, change the NSDL collection record back to another workflow status like **In Progress** and fix the problems discovered during harvesting. This change to **In Progress** will keep the collection record out of NSDL OAI Harvest Manager and out of NSDL.org search.

16.5 Make NSDL Collections Accessible to the Public NSDL OAI Provider

The NSDL collection record controls access to the item-level metadata records of a collection. First, the collection record must be accessioned for items to be available and secondly, the proper OAI information needs to be completed. If the OAI visibility field is set to **public**, item records are available to all NSDL systems and services. If the OAI visibility field is set to **private** or **protected**, then item-level records of the collection are not eligible for the public NSDL OAI provider. Here is what the different terms mean:

- **public** – items within the collection are publicly visible to the NSDL OAI provider and all NSDL systems and services
- **protected** – items within the collection are ONLY visible in non-public NSDL OAI interfaces (OAI server that NSDL uses to create the interface at NSDL.org)
- **private** – items within the collection are not visible to any NSDL OAI services (includes the OAI server that supports NSDL.org); owners of the items may access the items directly through NDR API service calls like find results and related objects, etc.

16.6 What Happens when NSDL Collection Records are Deleted or Deaccessioned?

When an NSDL collection record in the MGR application instance is given a workflow status other than **Done**, the collection and all of its item-level records remain in the NDR but are not available to NSDL.org search. If you delete the collection record, it is removed from the MGR application instance. And eventually the item records are removed from the NDR.

16.7 How To Connect An DCS-Created Collection of Items to Its NSDL Collection Record?

This is the process of connecting a collection of items created and managed by the DCS to the NSDL collection record representing the collection in the NDR. Before starting this process, be sure the collection exists in the DCS instance and that an NSDL collection record exists (by the same name) in the collection of NSDL Collection Records. Then complete the following steps:

1. Go to the collection record within the collection of NSDL Collection Records. Click on the NDR handle.

2. Scroll down to DCS:collectionMetadataFor. Click on handle. Copy the MetadataProvider handle that appears in the response box to a scratch file. Label it metadata provider.
3. Continue to scroll down in the response area and copy the nsdl:aggregatedBy handle to a scratch file. Label it aggregator.
4. Go to ***Settings*** and click on ***Collection Settings***.
5. Click on the name of the collection you are trying to connect. Then click the ***Edit Settings*** button at the top.
6. Erase the existing values (if any) for the metadataProvider and aggregator.
7. Copy your above metadata provider handle into metadataProvider.
8. Copy your above aggregator handle into aggregator.
9. Click the ***Save*** button. You may have to wait awhile for things to process, especially if there are existing records in the collection.
10. You should not have to do anything to the existing items records. But it is a good thing to verify that they are being associated properly with the new metadata provider.
11. Go the collection record within the collection of NSDL Collection Records. Click on the NDR handle.
12. Scroll down to DCS:collectionMetadataFor. Click on handle. Copy the handle that appears in the response box to the NDR Object Handle box at the top of the page.
13. Click the ***List Members*** button. The resulting handle list should match the items in the collection as it exists in the DCS.
14. Alternatively, this check in steps 11-13 can be completed by clicking the ***NDR*** menu and choosing ***NDR Admin***. Then click the collection name. Then scroll down to the bottom of the table and click on the ***Metadata Records*** hyperlink. The Members List of handles should match the items in the collection as it exists in the DCS.

16.8 How to make the DCS Take Over an Orphan Collection?

When an NSDL collection becomes an orphan with no caretaker, the collection may be moved to the MGR DCS instance for management by NSDL TNS staff.

Appendix 1 CONTEXT DESCRIPTOR PARAMETERS

This is a listing of parameters within the context descriptor template. Some are described in more detail in different sections of this User Guide. Context descriptor parameters control configuration of the application as a whole. Operation parameters, application information parameters and NDR parameters are described.

A sample context descriptor template is at:
$TOMCAT/webapps/<appName>/docs/templates/context_descriptor_template.xml

OPERATIONS PARAMETERS

instanceName - defines a label to use as the name of your application, which will appear at the top of each page of the application. If you do not desire such a label, supply a value of "".

logo – the file name of an image that appears in the header of each page. This file must be placed in the "images" directory of the expanded application directory within webapps. The logo image is displayed at 35 pixels tall by 135 pixels wide.

authenticationEnabled - (true | false) supply any value BUT "true" here to disable the authorization system. Note: when authentication is NOT enabled there will be no login and no support for roles and permissions.

collBaseDir - the full path to your metadata records. Replace the value here with a path in your file system. If the path does not exist it will be created when the application starts. It is _strongly_ recommended to supply a path outside of the application folder for this important information otherwise the information will be overwritten during a software update.

dcsConfig - the directory under which your application configuration files will go, including: user information, collection-specific configurations and configurations for the metadata frameworks supported by your application instance. It is _strongly_ recommended to supply a path outside of the application folder for this important information otherwise the information will be overwritten during a software update. A suggested value is: $TOMCAT/DCS_conf

exportBaseDir - the file system directory path where all metadata record files are exported to. Replace the value with one appropriate for your system. It is _strongly_ recommended to supply a path outside of the application folder for this important information otherwise the information will be overwritten during a software update.

autoExportStartTime - the time of day to run the autoExporter in HH:mm, for example 0:35 or 23:35. The autoExporter runs once a day at this time, beginning the day following when the application is started. AutoExport is off by default and therefore will

not be active if this parameter is not specified in the context descriptor. If the parameter IS present in the context descriptor, a value of “0” or "disabled" prevents it from running.

OPTIONAL PARAMETERS FOR APPLICATION INFORMATION

It is recommended to place indexing and other application information outside the application folder. This is information that is regenerated by the application (like on a start/stop of Tomcat). However, it is best to store this information outside the application folder so that any customizations you make or data the application caches are retained.
indexLocation - the directory where the index is located and built. A suggested value for this parameter is:

$TOMCAT/var/<appName>_app_info/index

repositoryData - the directory where the application’s persistent data files reside (information that controls services in the application or data that is cached and needed by the application). A suggested value for this parameter is:

$TOMCAT/var/<appName>_app_info/repositoryData

repositoryConfigDir - the directory to property files that control the behavior of the Indexer. A suggested value for this parameter is:

$TOMCAT/var/DCS_app_info/repositoryConfig

OPTIONAL NDR PARAMETERS

These parameters are necessary to allow the application to interact with the NDR (NSDL Data Repository).

ndrServiceEnabled - (true | false) must be "true" to enable interaction with NDR.

ndrApiBaseUrl - URI of NDR server (e.g., "http://ndr.nsdl.org/api").

DCSAgentHandle - handle of NDR Agent Object (DCS.nsdl.org) for this application. Note: This value must be obtained from the NDR Administrators (contact NSDL at http://nsdl.org/about/contactus/).

DCSAgentPrivateKey - path to a local file holding the private key representing the Agent for this application. Note: Consult with NDR Administrators about how to create your key (contact NSDL at http://nsdl.org/about/contactus/).

asnStandardsDocument - quick configuration for standards suggestion service for **ncs_item** framework. Value is a path to local file or directory containing ASN (Achievement Standards Network) standards documents.

standardsSuggestionServiceConfig - path to local file containing full CAT service configuration. NOTE: if this parameter is present, the **asnStandardsDocument** parameter above is ignored.

oaiPmhEnabled - enables the OAI provider. Set to true to enable OAI provider or set to anything other than true to disable. By default, the OAI service is enabled.

Appendix 2 COLLECTION CONFIGURATION PARAMETERS

Collection configuration parameters provide individual collection management information. This is a listing of parameters and their definitions as they appear in the collection configuration record (i.e. document order). Some parameters are described in more detail in different sections of this User Guide.

To create a sample collection configuration file, create a new collection (e.g. a test collection). Then under the ***Settings*** menu choose ***Collection Settings***. Choose ***edit*** for the new collection you created. You are now editing a sample collection configuration file.

collectionId - the internal identification number (id) used by the application to identify a collection. It tends to be a large integer (e.g. 1231459660617).

xmlFormat - the name of the XML format of the metadata framework being used by the collection.

idPrefix - a label (prefix) of alphanumeric characters used to construct metadata record identification numbers (record ids). Record ids look like NSDL-000-000-000-123 where NSDL is the idPrefix. Care should be taken when changing this value, since the result will be multiple idPrefixes within the same collection.

statusFlags - the set of labels (and definitions) used to indicate the status of metadata records within a collection. See section: 5.0 WORKFLOW STATUSES.

tuples - name and a value parameters for storing collection-specific default values for metadata fields (as opposed to the framework-specific default values specified in the framework configuration). For example, the framework configuration for the metadata framework of **adn** specifies a “termsOfUse” metadata field. By using the tuples parameters, each collection that uses the **adn** metadata format can then specify its own default value for “termsOfUse” like “Terms of use consistent with the Collection Development Policy”.

exportDirectory - a path (relative to the system-wide ***exportBaseDir***) where exported records for this collection are written to the local file system. If the path is relative to the ***exportBaseDir***, but does not exist, it is created at export time.

authority - a value of **ndr** means this collection is synced with the NDR (the NSDL Data Repository). Any other value (e.g. **dcs**, the default) means the collection is not synced.

ndrInfo - information used by the system to coordinate interactions with the NDR. The values for the ndrInfo parameters are generated automatically by the application when a collection is synced to the NDR. In this case, the interaction that is coordinated is the

writing of the metadata records to the NDR for the collection. However, for a collection to be recognized as an NSDL collection, authorized values for these fields must be obtained from the NSDL.

metadataProvider handle – (an ***ndrInfo*** parameter); a value identifying this collection's metadataProvider object in the NDR.

aggregator handle - (an ***ndrInfo*** parameter); a value identifying this collection's aggregator object in the NDR.

ndrOaiLink - legacy field no longer used and you may not see.

services - provides a list of cataloger assist services that can be enabled or disabled. Currently the only service provided and controlled is the CAT service, which suggests potentially relevant educational standards for cataloged resources.

allowSuggestions - used to enable/disable the CAT service for a collection. A value of false disables the CAT service for the collection. When the CAT service is configured for use by a particular metadata framework, it is enabled by default for all collections that use that metadata framework.

Appendix 3 FRAMEWORK CONFIGURATION PARAMETERS

Framework configuration parameters provide information about a metadata framework beyond what is contained in the XML schemas that defines the framework. This is a listing of parameters and their definitions as they appear in the framework configuration record (i.e. document order). Some parameters are described in more detail in different sections of this User Guide.

A sample framework configuration record is at:
$TOMCAT/webapps/<appName>/docs/templates/custom_framework.xml

name - an optional human-readable name for the framework. Used for user interface labels but does not affect internal application operation.

xmlFormat - the name given to the XML format of the metadata framework (read-only)

schemaInfo - groups parameters of schema-related information.

schemaURI - the URI of the metadata framework schema.

rootElementName - the name of the root XML element of the metadata framework schema (including a namespace if relevant). Because a schema can contain several top-level element definitions, this parameter is required to inform the application which XML element is intended to be the root element.

paths - a set of ***path*** parameters that specify defaults and behaviors for individual xpaths within the metadata framework schema (see SCHEMA PATHS below)

editorInfo - groups parameters providing information for the metadata editor.

renderer - the Java class used to render the metadata editor pages.

baseRendererLevel - controls the page layout of metadata fields displayed in the metadata editor by indicating a level in the XML instance document at which to begin editor pages. A value of “2” means render all XML elements directly below the root element on a single page. A value of “3” means render the XML elements across multiple editor pages with each page starting with an XML element at the second level within the XML instance document.

firstPage - indicates which metadata editor page is shown when a user enters the editor (relevant only when ***baseRendererLevel*** is greater than 2).

rebuildOnStart - (true | false) indication of whether the metadata editor pages re-generate each time the system starts.

editorPages - specifies the ***editorPage*** and ***pageLabel*** parameter pairs for each metadata editor page.

editorPage - the name of the XML element from the instance document that is rendered as the top-level element for a particular metadata editor page.

pageLabel - a nice name for the metadata editor page that will render a set of XML elements.

fieldInfoURI - specifies a URL or file location for the XMLfields loader file (e.g. fields-list.xml) that provides cataloging best practices and controlled vocabulary definitions (see section 15.5 CUSTOMIZING CATALOGING BEST PRACTICES AND VOCABULARY DEFINITIONS).

vocabLayoutURI - specifies a URL or file location for the XML controlled vocabulary groups loader file (e.g. groups-list.xml) that controls the layout and formatting of controlled vocabularies (see section 15.4 CUSTOMIZING USER INTERFACE LAYOUT OF CONTROLLED VOCABULARIES).

standaloneInfo and ***recordsDir*** - legacy parameters that can be ignored.

SCHEMA PATHS

As noted above, the ***path*** tags enable specification of default behaviors for individual metadata framework XML schema elements (which become fields in the metadata editor). Here are the various properties that can be assigned to paths:

path - the full xpath uniquely identifying the path to an XML element within the schema (e.g. /record/general/recordID)

pathName - a short name/label for the ***path*** parameter that can be used to identify the xpath.

defaultValue - an initial value for the XML element defined in the ***path*** parameter if no other value can be determined from the ***valueType*** parameter.

valueType - indicates a behavior to follow to determine a default value for the XML element defined in the ***path*** parameter. The values for this parameter and their resultant behavior are listed below:

- *date* - the current date (yyyy-MM-dd format) is inserted.
- *utcDate* - the current date (yyyy-MM-ddTHH:mm:ssZ) is inserted.
- *collectionInfo* - a value determined by the ***tuples*** parameter in the collection configuration record (see Appendix 2: COLLECTION CONFIGURATION PARAMETERS).
- *url* - values entered in the metadata editor are normalized.

- *uniqueUrl* - uniqueness for this field (within a given collection) is enforced and the URLs are normalized.
- *inputHelper* - identifies a file that provides a unique interaction in the **Metadata Editor** for the **path** parameter (NOTE: for inputHelper to work, the **path** parameter must refer to an XML complex datatype within a metadata framework). This feature is beta.

requiredByMinimalRecord - when true, a value is inserted for the XML element defined in ***path*** when a record is created. If a ***valueType*** is specified, then it determines the value. Otherwise, the ***defaultValue*** is used (if specified). If neither ***valueType*** or ***defaultValue*** are specified, a place-holder string (e.g. “description_goes_here”) is inserted in the new record.

requiredByCopyRecord - when true, the value from the source record is not carried forward to the copied version. However, a new value is inserted in the copied record using the procedure described in the ***requiredByMinimalRecord*** parameter. Use this parameter to ensure values from the source record are not inserted into the copied record.

readOnly - if true, the XML element defined in ***path*** cannot be edited.

maxLen - used to format text-entry fields in the metadata editor. If this editor field is rendered as a text input, then the field’s size, in characters, is determined by the value of ***maxLen***.