User Interface Administration Guide

The User Interface Administration Guide is targeted at systems administrators responsible for installing the DC Reference UI.

Installing the User Interface

Prerequisites

Understanding of Physical Architecture

Before carrying out the installation of the DC Reference User Interface (Reference UI, or simply, UI) web application you should read the documentation and subsections on the Physical Architecture of the system so you have an understanding of how and where the components should be installed.

Working Data Conservancy Service (DCS) Installation

Prior to installation of the Reference UI application you should already have a working installation of the Data Conservancy Service (DCS), because the Reference UI leverages the DCS to perform storage and archiving functions. See the Installation section of the System Administration Guide and verify that you have a working installation of the DCS.

Assumptions

The following installation instructions for the Reference UI web application assumes:

1. That the Reference UI web application is installed on the same server that the DCS is installed on (e.g. the "dcservice" machine), and that
2. The DCS web application has already been installed.
3. That the Reference UI web application is being installed into the same instance of the Servlet Container (e.g. Jetty) that the DCS is installed in.

The following pages provide information on how to install the DC Reference User Interface web application.

Database Installation

Installation

The DC Reference UI uses a database to store:

- User login and profile information
- Relationships between Projects, Collections, Data Items, and other objects in the Reference UI data model

Password Encryption
Please note: user passwords are ***not*** encrypted when stored in the database.
This will certainly change in future releases.

The current versions of the DCS and Reference UI require the Derby Relational Database. (Future releases will support more database platforms such as Postgres). The System Administration Guide provides the installation instructions for the Derby Relational Database as part of the DCS deployment.

Importantly, the Reference UI requires its own instance of Derby. You cannot reuse the same instance that is used by the DCS (technically, this is not true, however, to minimize the documentation and configuration required, we are supporting this approach), so you will need to allocate a location for the Reference UI Derby Database, e.g. `$DCS_HOME/ui/dcsui`.

Configuration

Because this documentation assumes that you are installing the Reference UI on the same server as the DCS, there is no configuration necessary. The required steps have already been performed when installing the DCS.

If you were installing the Reference UI on a different server, or within a different Servlet Container than the DCS, you would need to insure that `$DERBY_HOME/lib/derby.jar` is present in a shared classloader of your Servlet Container.

Jetty Installation

Installation

As previously stated, the installation instructions assume that the Reference UI is installed on the same application server, within the same servlet container, that the DCS has been installed to. The Jetty application server installation instructions are included as part of the System Administration Guide in the DC Service section.

Configuration

No additional configuration is required assuming the instructions in the System Administration Guide for configuring Jetty were followed.

DC Reference UI Installation

1. Prerequisites

1. Working installation of the DCS.
2. A Derby instance has been allocated per Database Installation

2. Install Reference UI Web Application

1. Ensure that the Jetty application server has been stopped.
2. Download the dcs-ui.war file (see Release Notes & Downloads for the download location).
    a. Save it as `dcs-ui.war`
3. Copy `dcs-ui.war` to the Jetty application server webapps directory `$JETTY_HOME/webapps`

3. Setup

Create the DC User Interface configuration file $JETTY_HOME/resources/ext.properties with the following content (customized for your deployment):

```
archiveService = org.dataconservancy.ui.services.ArchiveServiceImpl

dcs.ui.db.init = true
dcs.ui.db.driverClassName = org.apache.derby.jdbc.EmbeddedDriver
dcs.ui.db.url = jdbc:derby://localhost:1527/usr/local/dcs/ui/dcsui;create=true

dcs.ui.id.hibernate.ddl = create

dcs.ui.scheme = http
dcs.ui.hostname = dcservice.dataconservancy.org
dcs.ui.port = 8080
dcs.ui.contextPath = /dcs-ui

dcs.connector.scheme = http
dcs.connector.host = dcservice.dataconservancy.org
dcs.connector.port = 8080
dcs.connector.contextPath = /dcs
dcs.connector.maxOpenConnections = 30
dcs.connector.connectionTimeout = 120
dcs.connector.connectionPoolTimeout = 30
```

You will need to customize the the values of the following properties to suite your environment:

1. dcs.ui.db.url: This URL should include the path to the $DCS_HOME directory, with /ui/dcsui appended to it
    a. If your $DCS_HOME was /var/dcs, then the value of dcs.ui.db.url would be jdbc:derby:/var/dcs/ui/dcsui;create=true
2. The following properties, when combined together, should be the URL for the DC Reference UI. In this example, the DC Reference UI URL is `http://dcservice.dataconservancy.org:8080/dcs-ui`:
    a. dcs.ui.scheme: this is the scheme used in the url: http or https
    b. dcs.ui.hostname: this is the host use to connect to the UI
    c. dcs.ui.port: this is the port used to connect to the UI
    d. dcs.ui.contextPath: this is the UI servlet's context path in the servlet container
3. dcs.connector properties: These properties, when combined together, should be the the same value as `dcs.baseurl` from `$JETTY_HOME/resources/dcs.properties`:
    a. In this example `dcs.baseurl` is `http://dcservice.dataconservancy.org:8080/dcs`
    b. dcs.connector.scheme: this is the URL scheme used to connect to the DCS: http or https
    c. dcs.connector.host: this is the host used to connect to the DCS
    d. dcs.connector.port: this is the port used to connect to the DCS
    e. dcs.connector.contextPath: this is the DCS servlet's context path in the servlet container

If you wish to enable email notification for your installation, define and set values for the following properties (sample property values are documented here, but you should set values appropriate for your SMTP environment):

```
# Enables email notifications
# When set to false, all email notifications
# are disabled.
emailServiceEnabled = true

# SMTP server and port number used to send email notifications
smtpServer = smtp.gmail.com
portNumber = 465

# If the server in use is SSL enabled, set this property
# to true. Else, set it to false.
sslEnabled = false

# If the SMTP server requires authentication, set
# this property to true. Else, set it to false
authenticationEnabled = false

# The username and password used when SMTP auth is enabled
# Note: these fields should not be commented out when
# SMTP auth is disabled, just set them to invalid values
username = un@real.org
password = youShallNotPass
```

4. Start & Stop the UI

Database Table Initialization
The UI initializes database tables depending on the value of two properties.

1. The **`dcsui.db.init`** property. When this property is set to **`true`**, certain database tables will be deleted, re-created, and seeded with bootstrap data. When the property is set to **`false`**, the database is left alone.
2. The **`dcs.ui.id.hibernate.ddl`** property. When this propert is set to **`create`**, certain database tables will be deleted and re-created. When this property is set to **`validate`**, the UI checks if the tables are present, but otherwise leaves the database alone.

DCS Restart
Because the DC Reference UI and the DCS are installed in the same servlet container, starting and stopping the UI will also start and stop the DCS

When starting the UI web application for the first time:

1. Set the property **`dcs.ui.db.init`** in the ext.properties file to **`true`**.
2. Set the property **`dcs.ui.id.hibernate.ddl`** in the ext.properties file to **`create`**.
3. Start the UI by starting Jetty: `$JETTY_HOME/bin/jetty.sh start`
    a. Wait for Jetty to successfully start
4. Set the property **`dcs.ui.db.init`** in the ext.properties file to **`false`**.
5. Set the property **`dcs.ui.id.hibernate.ddl`** in the ext.properties file to **`validate`**.
    a. Subsequent restarts of the DC UI web application will not modify the database.

Stop the UI by stopping Jetty: `$JETTY_HOME/bin/jetty.sh stop`

5. Verify

The URL for the DC Reference UI is similar to the URL for the DCS, because they are both installed on the same server, in the same servlet container. If the DCS URL (the value of `dcs.baseurl` in `$JETTY_HOME/resources/dcs.properties`) is `http://dcservice.dataconservancy.org:8080/dcs`, then the UI URL is `http://dcservice.dataconservancy.org:8080/dcs-ui`.

Ensure that Jetty is running.

From your web browser, you should be able to connect to http://dcservice.dataconservancy.org:8080/dcs-ui and see the following (you should automatically be redirected to http://dcservice.dataconservancy.org:8080/dcs-ui/home/home.action):

6. Re-Initialize

To clear out the DC UI database:

1. Stop Jetty
2. Edit $JETTY_HOME/resources/ext.properties
    a. Change the value of the **`dcs.ui.db.init`** property to **`true`**
    b. Change the value of the **`dcs.ui.id.hibernate.ddl`** property to **`create`**
3. Start Jetty.
4. Edit $JETTY_HOME/resources/ext.properties
    a. Change the value of the **`dcs.ui.db.init`** property to **`false`**
    b. Change the value of the **`dcs.ui.id.hibernate.ddl`** property to **`validate`**

Configuring the User Interface

This section provides information on the system property files and settings for the dcs ui web application. These properties:

- configure connections between system components
- specify connection details (logins, urls, ports, database drivers etc) for system components and databases
- configure email notification setttings
- provision default system users
- specify DC software version
- configure system behavior

Dot Properties Files

Default Users

Logging

Notifications & Error Handling

Data Disciplines & Metadata Formats

Dot Properties Files

There are several properties files used to configure the DC UI.

1. Internal & External Property Files

dcs-ui uses the int.properties and ext.properties files to configure various components of the DCS User Interface. The Spring framework is configured to use the properties in these files to construct various objects.

Spring looks in int.properties located in WEB-INF/classes/int.properties of the dcs-ui.war and looks in ext.properties on the classpath (e.g. $JETTY_HOME/resources/ext.properties). int.properties is part of the dcs-ui.war whereas ext.properties is created manually as part of the dcs-ui installation and configuration.

1.1. Property Overriding

ext.properties "shadows" int.properties. If you define a property of the same name in both ext.properties and int.properties, the value defined in ext.properties will override that defined in int.properties.

1.2. Internal & External Property File Properties

- The implementation of the Archive Service interface. This shouldn't need to be changed unless you customize the implementation in some way
    - `archiveService = org.dataconservancy.ui.services.ArchiveServiceImpl`

- DC User Interface Database Connection Properties
    - `dcs.ui.db.init = true`
        - Set this property to 'true' to initialize/clear out the DC User Interface database (e.g. when deploying a new DC installation).
        - This property should be set to 'false' once DC application server has been started.
    - `dcs.ui.db.driverClassName = org.apache.derby.jdbc.EmbeddedDriver`
    - `dcs.ui.db.url = jdbc:derby://localhost:1527//usr/local/dcs/ui/dcsui;create=true`
    - `dcs.ui.db.username = dcsui`
    - `dcs.ui.db.password = dcsui`
    - `dcs.ui.id.hibernate.ddl = create`
        - Set this property to 'create' to initialize the DC User Interface database when first run (creates the schema, destroying any previous data)
        - Set this property to 'validate' subsequently (validate the schema, makes no changes to the database).
        - Set this property to 'update' to update the schema at the start of the session.
        - Set this property to 'create-drop' to drop the schema at the end of the session.

- DC User Interface URL Properties: used to construct links to pages in the UI (e.g. when sending emails to users)
    - `dcs.ui.scheme = http`
    - `dcs.ui.hostname = dcservice.dataconservancy.org`
    - `dcs.ui.port = 8080`
    - `dcs.ui.contextPath = /dcs-ui`

- DCS Connector URL Properties: used to construct links to DCS APIs (e.g. when the UI needs to contact the DCS to perform an operation)
    - `dcs.connector.scheme = http`
    - `dcs.connector.host = localhost`
    - `dcs.connector.port = 8086`
    - `dcs.connector.contextPath = /dcs`
- DCS Connector connection Properties: used to control the number of connections established between the DC UI and the DCS.
    - `dcs.connector.maxOpenConnections = 30`
    - `dcs.connector.connectionTimeout = 120`
    - `dcs.connector.connectionPoolTimeout = 30`

- Controls whether or not the UI will send email notifications. In a test environment, setting this to `false` will prevent any emails from spamming potentially legitimate email addresses.
    - `emailServiceEnabled = true`

2. defaultUsers.properties

Located in WEB-INF/classes/defaultUsers.properties of the dcs-ui.war.

See information in Default Users

3. displaytag.properties

Located in WEB-INF/classes/displayTag.properties of the dcs-ui.war.

This is the configuration file for displaying tag tables.
TODO: Understand what this means.

4. revisionProperties

Located in WEB-INF/classes/revision.properties of the dcs-ui.war.

Contains the DCS-UI build and revision numbers and timestamp. This file should not be altered.

5. StripesResources.properties

Located in WEB-INF/classes/stripesResources.properties of the dcs-ui.war.

Default Resource Bundle file for the Stripes Framework. This file should not be altered. Values are placed in here for the following:

- Form Field Labels
- Error messages for:
    - Standard validation error messages
    - Converter error messages
    - Error messages used in custom ActionBean classes

Default Users

Default DCS system users are defined in the defaultUsers.properties system property file located in:

- $JETTY_HOME/webapps/dcs-ui/WEB-INF/classes/defaultUsers.properties

By default this defines:

Admin User

user: admin
password: ilovec00kies!
registration status: APPROVED
roles: ROLE_USER, ROLE_ADMIN

System User

user: user
password: ilovetoc0unt!
registration status: APPROVED
roles: ROLE_USER

For information on registration status and roles see Registration Status.

Logging

1.1. About

The DC Reference UI (and the DCS) uses the Logback logging framework to manage logging.

The logging system is configured using WEB-INF/classes/logback.xml in the dcs-ui web application. Please see the Logback configuration guide for information on how to configure logging.

1.2. Customizing Logging

Both the DCS and the Reference UI are distributed as WAR files. Each of them have a `logback.xml` configuration file. This would normally mean that adjusting the logging of the applications means unpacking the WAR file and editing the appropriate `logback.xml` file.

However, Logback supports system property `logback.configurationFile` which will override the logging configuration found in the packaged WAR files. If you configure your servlet container to start with the `logback.configurationFile` property defined, you can customize the logging output of both applications. Note that the value of the property should be a valid URL, so if you want to reference a file on the file system, it should take the following form (for example): `file:/usr/local/jetty/resources/logback-ext.xml`

1.3. Example config

Here is an example Logback configuration file that could be used:

```
<configuration scan="true" scanPeriod="10 seconds">

  <!--
    | Appenders
    +-->

  <appender name="CONSOLE" class="ch.qos.logback.core.ConsoleAppender">
    <encoder>
      <pattern>%-3p [%t]: %c{3}@%L %d %m %n</pattern>
    </encoder>
  </appender>

  <appender name="FILE" class="ch.qos.logback.core.FileAppender">
    <file>/usr/local/jetty/logs/dcs-ui.log</file>
    <append>false</append>
    <encoder>
      <pattern>%-3p [%t]: %c{3}@%L %d %m %n</pattern>
    </encoder>
  </appender>

  <appender name="DCS" class="ch.qos.logback.core.FileAppender">
    <file>/usr/local/jetty/logs/dcs.log</file>
    <append>false</append>
    <encoder>
      <pattern>%-3p [%t]: %c{3}@%L %d %m %n</pattern>
    </encoder>
  </appender>

  <!--
    | Loggers
    +-->

  <root level="ERROR">
    <appender-ref ref="CONSOLE"/>
    <appender-ref ref="FILE"/>
  </root>

  <logger name="org.dataconservancy" additivity="false" level="DEBUG">
    <appender-ref ref="CONSOLE"/>
    <appender-ref ref="DCS"/>
  </logger>

  <logger name="org.dataconservancy.ui" additivity="false" level="WARN">
    <appender-ref ref="CONSOLE"/>
    <appender-ref ref="FILE"/>
  </logger>

  <logger name="net.sourceforge.stripes" level="WARN">
    <appender-ref ref="CONSOLE"/>
    <appender-ref ref="FILE"/>
  </logger>

  <logger name="org.springframework" level="WARN">
    <appender-ref ref="CONSOLE"/>
    <appender-ref ref="FILE"/>
  </logger>

  <logger name="org.springframework.security" additivity="false" level="WARN">
    <appender-ref ref="CONSOLE"/>
    <appender-ref ref="FILE"/>
  </logger>

  <logger name="org.springframework.jdbc" additivity="false" level="WARN">
    <appender-ref ref="CONSOLE"/>
    <appender-ref ref="FILE"/>
  </logger>

  <logger name="org.apache.solr" level="WARN">
    <appender-ref ref="CONSOLE"/>
```

```
    <appender-ref ref="FILE"/>
  </logger>

  <!--
    | "Useful" classes
    |
    | Adusting the log level of these classes can be useful (e.g. to DEBUG or TRACE),
    | because they give specific insight into what the system is doing, without
    | requring you to set the entire package (like Spring Security) to DEBUG or TRACE.
    +-->

  <!-- Logs interactions of the UI with the DCS Archive -->
  <logger name="org.dataconservancy.ui.services.ArchiveServiceImpl" additivity="false" level="DEBUG">
    <appender-ref ref="CONSOLE"/>
    <appender-ref ref="FILE"/>
  </logger>

  <!-- Logs the mapping of the DataSet UI business object to a DCP package and back -->
  <logger name="org.dataconservancy.ui.dcpmap.DataSetMapper" additivity="false" level="TRACE">
    <appender-ref ref="CONSOLE"/>
    <appender-ref ref="FILE"/>
  </logger>

  <!-- Setting this logger to DEBUG will reveal all of the SQL executed by Spring JDBC -->
  <logger name="org.springframework.jdbc.core.JdbcTemplate" additivity="false" level="WARN">
    <appender-ref ref="CONSOLE"/>
    <appender-ref ref="FILE"/>
  </logger>

  <!-- Setting this logger to DEBUG will reveal the workings of Spring MVC -->
  <logger name="org.springframework.web" additivity="false" level="DEBUG">
    <appender-ref ref="CONSOLE"/>
    <appender-ref ref="FILE"/>
  </logger>

  <!-- Setting this logger to TRACE will reveal the workings of the Request Util -->
  <logger name="org.dataconservancy.ui.api.support.RequestUtil" additivity="false" level="TRACE">
    <appender-ref ref="CONSOLE"/>
    <appender-ref ref="FILE"/>
  </logger>

  <!-- Setting this logger to DEBUG will reveal the workings of the ProjectController -->
  <logger name="org.dataconservancy.ui.api.ProjectController" additivity="false" level="DEBUG">
    <appender-ref ref="CONSOLE"/>
    <appender-ref ref="FILE"/>
  </logger>

  <!--
    | "Noisy" classes
    |
    | These classes are typically noisy at DEBUG or INFO levels, so we call them
    | out here, and mute them by setting them to ERROR or WARN levels.
    +-->

  <logger name="net.sourceforge.stripes.vfs.DefaultVFS" additivity="false" level="ERROR">
    <appender-ref ref="CONSOLE"/>
    <appender-ref ref="FILE"/>
  </logger>

  <logger name="net.sourceforge.stripes.tag.layout" additivity="false" level="ERROR">
    <appender-ref ref="CONSOLE"/>
    <appender-ref ref="FILE"/>
  </logger>

  <logger name="net.sourceforge.stripes.controller.UrlBindingFactory" additivity="false"
level="ERROR">
    <appender-ref ref="CONSOLE"/>
    <appender-ref ref="FILE"/>
  </logger>
```

```
  <logger name="org.apache.http.impl.client.DefaultRedirectStrategy" additivity="false" level="ERROR">
    <appender-ref ref="CONSOLE"/>
    <appender-ref ref="FILE"/>
  </logger>

  <logger name="org.apache.http.impl.client.DefaultRequestDirector" additivity="false" level="ERROR">
    <appender-ref ref="CONSOLE"/>
    <appender-ref ref="FILE"/>
  </logger>

  <logger name="org.springframework.beans.factory.support.DefaultListableBeanFactory"
additivity="false" level="ERROR">
    <appender-ref ref="CONSOLE"/>
  </logger>

  <logger name="net.sourceforge.stripes.controller.multipart.DefaultMultipartWrapperFactory"
additivity="false" level="ERROR">
    <appender-ref ref="CONSOLE"/>
    <appender-ref ref="FILE"/>
  </logger>

  <logger name="net.sourceforge.stripes.util.ResolverUtil" additivity="false" level="ERROR">
    <appender-ref ref="CONSOLE"/>
    <appender-ref ref="FILE"/>
  </logger>

  <logger name="net.sourceforge.stripes.format.DefaultFormatterFactory" level="ERROR">
    <appender-ref ref="CONSOLE"/>
    <appender-ref ref="FILE"/>
  </logger>

  <logger name="org.apache.solr.core.SolrCore" additivity="false" level="WARN">
    <appender-ref ref="CONSOLE"/>
    <appender-ref ref="FILE"/>
```

```
  </logger>

</configuration>
```

Notifications & Error Handling

1. Overview

The Reference UI has a centralized exception handling and notification framework that is responsible for firing, managing and sending out notifications.

When an event (e.g. exception, user registration request, deposit status) results in a notification the notification service is configured with:

- An email address to send notifications from
- An email address (or addresses) to send notifications to
- An email subject line

Templates for message bodies (discussed below) are packaged in the `dcs-ui.war` file, but they can be edited to support custom email message bodies.

Currently the UI sends E-mail messages for the following events:

- Each time a Java exception is thrown
- Each time a new user registers for an account
- When a user's registration is approved
- When a user deposits a Data Item

2. Configure Behavior

2.1. Enable messaging in `ext.properties`

To enable the email notification service edit the `ext.properties` file located in `$JETTY_HOME/resources` and define the following properties; if they are already defined, ensure that their values are set correctly.

Importantly the `emailServiceEnabled` property needs to be set to `true`. If the value is `false`, all email messaging is disabled.

```
# Enables email notifications
# When set to false, all email notifications
# are disabled.
emailServiceEnabled = true

# SMTP server and port number used to send email notifications
smtpServer = smtp.gmail.com
portNumber = 465

# If the server in use is SSL enabled, set this property
# to true. Else, set it to false.
sslEnabled = false

# If the SMTP server requires authentication, set
# this property to true. Else, set it to false
authenticationEnabled = false

# The username and password used when SMTP auth is enabled
# Note: these fields should not be commented out when
# SMTP auth is disabled, just set them to invalid values
username=un@real.org
password=youShallNotPass
```

3. Customizing E-mail Messages

3.1. Customizing E-mail Senders, Recipients, and Subjects

3.1.1. Senders

The default sender is defined by the property `fromAddress`

Senders for each of the above events can be defined using the following properties:

- `registrationNotificationSender`
- `approvalNotificationSender`
- `depositStatusNotificationSender`

3.1.2. Recipients

The only event that allows a recipient to be set is the Java Exception Notification. You must set a recipient by defining the property `exceptionNotificationTo`. Multiple recipients can be defined by separating them with commas.

The other events automatically determine their recipients:

- Registration notification is automatically sent to all Instance Administrators
- Approval notification is automatically sent to the User whose registration was approved
- Deposit status notification is automatically sent to the User who deposited the Data Item

3.1.3. Subjects

Subjects are set by defining the following properties:

- `registrationNotificationSubject`
- `approvalNotificationSubject`
- `exceptionNotificationSubject`
- `depositStatusNotificationSubject`

3.2. Customizing E-mail Bodies

E-mail message bodies are actually Velocity templates, which allow some degree of customization to the email messages sent by the Reference UI.

The templates are located inside of the `dcs-ui.war` file, so you will need to unpack the WAR in `$JETTY_HOME/webapps` into a directory named `dcs-ui` (and remove the `dcs-ui.war` file). After unpacking the war file, you should be able to list the available templates in `$JETTY_HOME/webapps/dcs-ui/WEB-INF/classes/templates`.

3.3. email-exception-template.vm

This is the email message body that is sent each time the system encounters an exception.

The body of the message can be customized by editing the Velocity template in `WEB-INF/classes/templates/email-exception-template.vm`

3.4. new-registration-notification.vm

This is the email message body that is sent each time a new user registers for an account.

The body of the message can be customized by editing the Velocity template in `WEB-INF/classes/templates/new-registration-notification.vm`

3.5. user-approval-notification.vm

This is the email message body that is sent each time a User's status is updated from PENDING to APPROVED.

The body of the message can be customized by editing the Velocity template in `WEB-INF/classes/templates/user-approval-notification.vm`

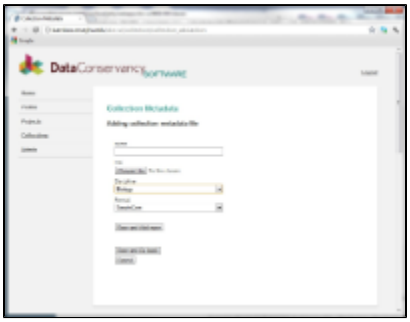
Data Disciplines & Metadata Formats

1. Disciplines

Disciplines exist as a controlled vocabulary in DCS and are used to describe metadata formats. For example

Format Name	Version	Identifier	Discipline(s)
STC	1.0	dc:format:metadata/STC	Astronomy
AVM	1.0	dc:format:metadata/AVM	Astronomy
Audubon Core	1.0	dc:format:metadata/AudubonCore	Biology
TaxonX	1.0	dc:format:metadata/TaxonX	Biology
CGDSMFGDC	1.0	dc:format:metadata/CGDSMFGDC	Earch Science
ISO19115	1.0	dc:format:metadata/ISO19115	Earch Science
DarwinCore	1.0	dc:format:metadata/DarwinCore	Biology

Note that currently these are used when asserting a Discipline and Metadata format for Collection Metadata files. Currently DCS does not perform validation of specific metadata file formats against particular schemas. It is up to the Collection creator to ensure the metadata file is well formed and adheres to the stated metadata format schema.



1.1. Updating Disciplines

System administrators may need to add additional disciplines and data formats for creators of Collections to select from when asserting metadata file formats. Currently this is performed by manually editing the file dcs-ui/WEB-INF/classes/BootstrapData.sql

This file is run on system startup (including all the other SQL files) if the "dcs.ui.db.init" property is "true" (see Internal & External Property Files in Dot Properties Files).

This file contains sql insert statements that create new entries in the **discipline** and **relationships** tables. For example:

```
---
--- Disciplines
---
INSERT INTO DISCIPLINE VALUES ('dc:discipline:Biology', 'Biology')
---
--- Relationships between Disciplines and MetadataFormats (Obverse)
---
INSERT INTO relationships VALUES ('dc:discipline:Biology', 'dc:format:metadata/TaxonX', 'AGGREGATES')
---
--- Relationships between Disciplines and MetadataFormats (Inverse)
---
INSERT INTO relationships VALUES ('dc:format:metadata/TaxonX', 'dc:discipline:Biology',
'IS_AGGREGATED_BY')
```

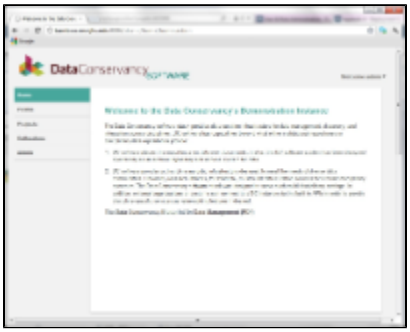
Verifying the Installation

This sections provides instructions for confirming that the dcs-ui web application has bee installed correctly.

Confirm System Login

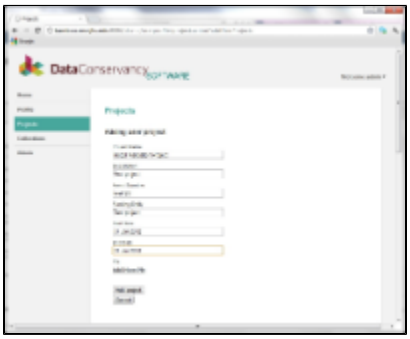
- Use a web browser and navigate to the deployed Reference UI web application as configured during the installation process. For example: http://dcservice-install.dataconservancy.org:8086/dcs-ui
- Confirm that you see the Reference UI login screen.

- Click on the Login link and login as an admin user (default admin login account is "admin" password "ilovec00kies!")

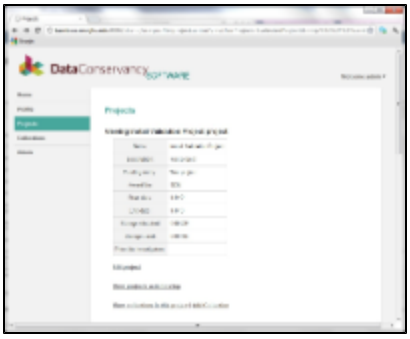


Confirm Creation of Entities in the Archive

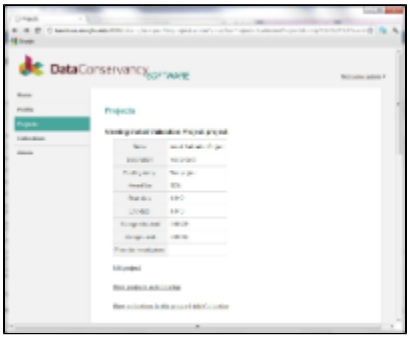
- Click on Projects in the left hand side menu to show the main Projects page, and then click on the "Add Project" link. In the "Add User Project" enter information into the form and click "Add Project".



- Confirm that the project appears in the list of projects on the main Projects page.
- Click on the name of the test project in the projects list to view the project and click on the "Add Collection" link towards the bottom of the page.



- Enter information into the required fields and click on the "Add Collection" button.



- This will take you back to the test project's details screen you previously entered. Click on the "View collections in this project" link and confirm that the Collection just entered appears.

- Click on the Collections name and then click on the Deposit Data link towards the bottom of the page.

- Using the Choose File button select a test data file and click the Deposit button.

- In the Collections view that appears click on the "View Data for Collection" link.
- Confirm that the deposited data appears in the "Data for Collection" screen.

- From the "Data for Collection" screen click on the "Download" link and confirm that the file is downloaded. Note that the file is downloaded by submitting a HTTP request to the DCS web application.

UI Backup & Recovery

The following components of the UI should be backed up.

- The Derby Relational Database used by the UI
- The dcs-ui.war file
- The UI Application configuration files

You should also ensure that the components of the DCS system are backed up according to the DCS Backup & Recovery documentation.

Derby Relational Database Backup

Please see the Derby database documentation on Backup and Recovery

The UI War file

Ensure that the dcs-ui.war file that was downloaded as part of the installation has been copied to a secure backup location.

Application Configuration Files

Ensure that any application configuration files that have been customized are copied to a secure backup location.

Relationship and interaction of the DCS and UI

For information on the interaction of the dcs services layer with the User Interface web application see High Level Overview of UI Relationship to DCS.