



Administrator's Guide for Metacat 1.10.0

Guide Version 1.1

February 2011



NCEAS

National Center for Ecological Analysis
and Synthesis (NCEAS)



Knowledge Network for Biocomplexity

Administrator's Guide for Metacat 1.9.3
Guide Version 1.0, August 1, 2010

1	Introduction.....	3
1.1	What's in this Guide	4
1.2	Metacat Features.....	4
2	Downloading and Installing Metacat.....	6
2.1	System Requirements.....	6
2.2	Installing on Linux.....	6
2.2.1	Quick Start Overview.....	6
2.2.2	Downloading Metacat	7
2.2.3	Installing and Configuring Required Software.....	9
2.2.4	Installing Metacat.....	16
2.2.5	Optional Installation Options (LSID Server).....	19
2.2.6	Troubleshooting	22
2.3	Installing on Windows	23
3	Configuring Metacat.....	27
3.1	Initial Configurations.....	27
3.2	Logging in to Metacat.....	30
3.3	Required Configurations	31
3.3.1	Global Properties (server, ports, etc).....	32
3.3.2	Authentication Configuration	34
3.3.3	Skins Configuration (look & feel)	34
3.3.4	Database Configuration.....	35
3.3.5	Geoserver Password Configuration (Highly Recommended)	37
3.4	Additional Configuration	38
4	Accessing and Submitting Metadata and Data	41
4.1	A Brief Note about How Information is Stored	41
4.2	Using the Registry	41
4.3	Using HTML Forms (the HTTP Interface).....	46
4.4	Using the EarthGrid API.....	61
4.5	Using Morpho.....	62
4.6	Creating Your Own Client	63
5	Metacat's Use of Geoserver	66
5.1	Installing and Configuring	67
5.2	Spatial Queries.....	73
6	Replication	75
7	Harvester and Harvest List Editor.....	82
7.1	Configuring Harvester	82
7.2	Configuring a Harvest Site (Instructions for Site Contact).....	84
7.3	Running Harvester	92
7.4	Reviewing Harvest Reports.....	94
8	Event Logging.....	96
9	Enabling Web Searches: Sitemaps.....	98
10	Creating a Java Class that Implements AuthInterface.....	99
11	Appendix: Metacat Properties.....	99

1 Introduction

Metacat is a repository for metadata (data about data), which helps scientists find, understand and effectively use the data sets they manage or that have been created by others. Thousands of data sets are currently documented in a standardized way and stored in Metacat systems, providing the scientific community with a broad range of ecological data that--because the data are well and consistently described--can be easily searched, compared, merged, or used in other ways.

Not only is the Metacat repository a reliable place to store metadata and data (the database is replicated over a secure connection so that every record is stored on multiple machines and no data is ever lost to technical failures), it provides a user-friendly interface for information entry and retrieval. Scientists can search the repository via the Web using a customizable search form. Searches return results based on user-specified criteria, such as desired geographic coverage, taxonomic coverage, and/or keywords that appear in places such as the data set's title or owner's name. Users need only click a linked search result to open the corresponding data-set documentation in a browser window and discover whom to contact to obtain the data themselves (or how to immediately download the data via the Web).

Metacat's user-friendly Registry application allows data providers to enter data-set documentation into Metacat using a Web form. When the form is submitted, Metacat compiles the provided documentation into the required format and saves it. Information providers need never work directly with the XML format in which the data are stored or with the database records themselves. In addition, the Metacat application can easily be extended to provide a customized data-entry interface that suits the particular requirements of each project. Metacat users can also choose to enter metadata using the Morpho application, which provides data-entry wizards that guide information providers through the process of documenting each data set.

The metadata stored in Metacat includes all of the information you and others need to understand what the described data are and how to use them: a descriptive data set title; an abstract; the temporal, spatial, and taxonomic coverage of the data; the data collection methods; distribution information; and contact information. Each information provider decides who has access to this information (the public, or just specified users), and whether or not to upload the data set itself with the data documentation. Information providers can also edit the metadata or delete it from the repository, again using Metacat's straightforward Web interface.

Metacat is a Java servlet application that runs on Window or Linux platforms in conjunction with a database, such as PostgreSQL (or Oracle 8i), and a Web server. The Metacat application stores data in an XML format using Ecological Metadata Language (EML) or another ecological metadata standard. For more information about Metacat or for examples of projects currently using Metacat, please see <http://knb.ecoinformatics.org>.

1.1 What's in this Guide

The Administrator guide includes information for installing, configuring, managing and extending Metacat for both Linux and Windows systems. Chapter Two contains instructions for downloading and installing Metacat and the applications required to run the software on Linux and Microsoft platforms. Chapter Three covers how to configure Metacat, both for new and upgraded installations. Chapter Four details the ways in which you can customize the Metacat interface so users can access and submit information easily: using Metacat's generic web-interface (the Registry), creating your own HTML forms, and creating your own desktop client (like Morpho). Chapter Five discusses how to work with Metacat's Geoserver. Chapter Six describes how to set up the Metacat's replication service, which permits Metacat servers to share data with each other, effectively backing up metadata and data files. Chapter Seven looks at the Metacat Harvester, a program that automates the retrieval of EML documents from one or more sites and their subsequent upload (insert or update) to Metacat. Chapter Eight discusses logging, Chapter Nine contains instructions for creating a site map, which makes individual metadata entries available via Web searches. Metacat's Java API is included as an appendix at the end of the guide.

1.2 Metacat Features

Metacat is a repository for metadata (data about data), which help scientists find, understand and effectively use the data sets they manage or that have been created by others. Specifically,

- Metacat is a Java servlet application, which can run on both Windows and Linux systems
- Metadata submitted to Metacat is broken into modules, which are stored to optimize rapid information retrieval
- Metacat's Web interface facilitates the input and retrieval of data (Figure 1.1)
- Metacat's optional mapping functionality enables you to query and visualize the geographic coverage of stored documents
- Metacat's replication feature ensures that all Metacat data and metadata is stored safely on multiple Metacat servers
- The Metacat interface can be easily extended and customized via Web forms, skins, and/or user-developed Java clients
- The Metacat harvester automates the process of retrieving and storing EML documents from one or more sites
- Metacat can be customized to use Life Sciences Identifiers (LSIDs), uniquely identifying every data record
- Metacat has a built-in logging system for tracking events such as document insertions, updates, deletes, and reads
- The appearance of Metacat's Web interface can be customized via skins.

Data Catalog Search

Home

search for data



You are *NOT* logged in ([Login](#)). You may search the data catalog without being logged into your account, but will have access only to "public" data (see "login & registration")

Enter a search phrase (e.g. biodiversity) to search for data sets in the data catalog, or simply browse by category using the links below.

» advanced search «

Search Title, Abstract, Keywords, Personnel (Quicker)

Search all fields (Slower)

Taxonomy

Plant, Invertebrate, Mammal, Bird, Reptile, Amphibian, Fungi, Microbe, Virus

Habitat

Alpine, Aquatic, Beach, Benthic, Desert, Estuary, Forest, Grassland, Marine, Montane, Oceanic, Savanna, Shrubland, Terrestrial, Tundra, Urban, Wetland

Data Catalog Map




Map Layers

- Dataset Points
- Dataset Bounds
- Satellite image
- World Map
- Countries

lon: lat:

Select a Location



[Download Google Earth KML](#)

login & registration

Logging into your account enables you to search any additional, non-public data for which you may have access privileges:

You are *NOT* logged in

username:

organization:

password:

[create a new account](#)

[forgot your password?](#)

[change your password](#)

Figure 1.1: Metacat's default home page. Users can customize the appearance using skins.

2 Downloading and Installing Metacat

Instructions for both Linux and Windows systems are included in this section.

2.1 System Requirements

In addition to meeting the recommended system requirements, the server on which you wish to install Metacat must have the following software installed and running correctly:

- PostgreSQL (or another SQL92-compliant RDBMS like [Oracle 8i](#))
- [Apache Jakarta-Ant](#) (if building from source)
- [Apache Jakarta-Tomcat](#)
- Apache Web server (recommended*)
- Java 6 (Note: Java 5 is deprecated)

System requirements for running Metacat:

- a server running an SQL92-compliant database (PostgreSQL recommended)
- at least 128MB RAM
- a Pentium III processor (or higher)
- 11 MB disk space (Note: The amount of disk space required depends on the size of your RDBMS tablespace (which should be at least 10 MB; however, Metacat itself requires only about 140 MB of free space after installation).

* In order to use the Metacat Registry (and for a more robust Web-serving environment in general), the Apache Web server should be installed with Tomcat and the two should be integrated. See the [installing Apache](#) for more information.

2.2 Installing on Linux

Section 2.2 contains instructions for downloading and installing Metacat on Linux systems.

2.2.1 Quick Start Overview

For the impatient or those who have already installed Metacat and know what they are doing, here are the steps needed to install Metacat. Detailed instructions for each step are in the next section.

1. Download and install prerequisites (Java 6, Tomcat 5, PostgreSQL, Apache), including the tomcat5.5 init.d script

2. Create a database in PostgreSQL named 'metacat' and authorize access to it in `pb_hba.conf` for the user 'metacat'
3. Log in to PostgreSQL and create the 'metacat' user
4. Download Metacat from the [KNB Software Download Page](#) and extract the archive
5. `sudo mkdir /var/metacat; sudo chown -R <tomcat_user> /var/metacat`
6. `sudo cp <metacat_package_dir>/knb.war <tomcat_app_dir>`
7. `sudo /etc/init.d/tomcat5.5 restart`
8. Configure Metacat through the Web interface

2.2.2 Downloading Metacat

Before installing Metacat, please ensure that all required software is installed and running correctly. To obtain a Metacat WAR file, which is needed for installation, download one of the following:

- the [Metacat installer](#), which has a pre-built WAR file,
- the [Metacat source distribution](#), which must be built in order to create a WAR file,
- the [Metacat source code from SVN](#). You must build the source code in order to create a WAR file.

Instructions for all three options are discussed below. Note that downloading the installer (described in the next section) is the simplest way to get started.

2.2.2.1 Download the Metacat Installer

Downloading the Metacat Installer is the simplest way to get started with the application. To download the installer:

- 1) Browse to the [KNB Software Download Page](http://knb.ecoinformatics.org/software/download.html) (<http://knb.ecoinformatics.org/software/download.html>). In the Metacat section, select the link to the "GZIP file" (the link should look like: `metacat-bin-X.X.X.tar.gz`, where X.X.X is the latest version of Metacat e.g., 1.9.0)
- 2) Save the file locally.
- 3) Extract the Metacat package files by typing:

```
tar -xvzf metacat-bin-X.X.X.tar.gz
```

You should see a WAR file and several supporting files (Table 2.1). The extraction location will be referred to as the `<metacat_package_dir>` for the remainder of this documentation.

File	Description
knb.war	The Metacat Web archive file (WAR)
knb	The Web definition file used by Apache on Ubuntu/Debian Linux systems.
Knb.ssl	The SSL definition file used by Apache on Ubuntu/Debian Linux systems.
jk.conf	The JkMount configuration file used by Apache on Ubuntu/Debian Linux systems.
workers.properties	The workers definition file used by Apache on Ubuntu/Debian Linux systems.
tomcat5.5	The Tomcat startup script for Tomcat 5.5 installed with apt-get on Ubuntu/Debian Linux systems.
authority	The optional LSID Server application WAR

Table 2.1: Files extracted from the Metacat GZip file.

2.2.2.2 Download Metacat Source Code

To get the Metacat source distribution:

- 1) Browse to <http://knb.ecoinformatics.org/software/download.html>. In the Metacat section, select the link to the Metacat Source code (it will look something like this: metacat-src-X.X.X.tar.gz, where X.X.X is the latest version of Metacat, e.g., 1.9.0).
- 2) Save the file locally.
- 3) Extract the Metacat package files by typing (replace X.X.X with the current version number):

```
tar -xvzf metacat-src-X.X.X.tar.gz
```

- 4) Rename the metacat-X.X.X directory to metacat.

Note that you *do not* need to create the WAR file directly because the Ant build-file has an "install" target that will build and deploy the WAR for you.

2.2.2.3 Check Out Metacat Source Code from SVN (for Developers)

If you wish to work with the most recent Metacat code, or you'd like to extend the Metacat code yourself, you may wish to check out the Metacat source code from SVN.

You will need a Subversion (SVN) client installed and configured on your system (see the end of this section for information about obtaining an SVN client).

To check out the code from SVN, go to the directory where you would like the code to live and type:

```
svn co https://code.ecoinformatics.org/code/metacat/tags/METACAT_<rev> metacat
```

Where <rev> is the version of the code you want to check out (like 1_9).

To check out the head, type:

```
svn co https://code.ecoinformatics.org/code/metacat/trunk metacat
```

You should see a list of files as they check out.

Note that you *do not* need to create the WAR file directly because the Ant build-file has an "install" target that will build and deploy the WAR for you.

Installing an SVN Client:

If you have not already installed Subversion and you are running Ubuntu/Debian, you can get the SVN client by typing:

```
sudo apt-get install subversion
```

Otherwise, you can get the SVN client from [The Subversion homepage](http://subversion.tigris.org/) (<http://subversion.tigris.org/>).

2.2.3 Installing and Configuring Required Software

Before you can install and run Metacat, you must ensure that a recent Java SDK, PostgreSQL (or another SQL92-compliant RDBMS like Oracle 8i), [Ant](#) (if installing from source), and [Tomcat](#) are installed and running correctly. We also highly recommend that you install Apache Web server, as it provides a more robust Web-serving environment and is required by some Metacat functionality.

- Java 6
- [Apache Jakarta-Tomcat](#)
- [Apache Web Server](#) (Highly Recommended)
- [PostgreSQL Database](#) (or Oracle 8i)
- [Apache Jakarta-Ant](#) (if building from Source)

2.2.3.1 Java 6

To run Metacat, you should use Java 6 (Java 5 is deprecated and will not be supported after Metacat version 1.9.2). Make sure that the `JAVA_HOME` environment variable is properly set and that both `java` and `javac` are on your `PATH`.

To install Java if you are running Ubuntu/Debian, type:

```
sudo apt-get install sun-java6-jdk
```

Click "ok" then "yes" for license agreement.

If you are not using Ubuntu/Debian, you can get Java from the [Sun website](http://www.sun.com) (<http://www.sun.com>).

2.2.3.2 Apache Jakarta-Tomcat

We recommend that you install Tomcat 5.5 into the directory of your choice. Included with the Metacat download is a Tomcat-friendly start-up script that should be installed as well.

Note: we will refer to the Tomcat installation directory as `<tomcat_home>` for the remainder of the documentation.

If you are running Ubuntu/Debian, get Tomcat by typing:

```
sudo apt-get install tomcat5.5
```

Otherwise, get Tomcat from [the Apache Tomcat page](#).

Install the Metacat-friendly Tomcat start-up script by typing:

```
sudo /etc/init.d/tomcat5.5 stop
sudo mv /etc/init.d/tomcat5.5 /etc/init.d/tomcat5.5.bak
sudo cp <metacat_package_dir>/debian/tomcat5.5 /etc/init.d/tomcat5.5
sudo chmod +x /etc/init.d/tomcat5.5
```

2.2.3.3 Apache Web Server (Highly Recommended)

Although you have the option of running Metacat with only the Tomcat server, we highly recommend that you run it behind the Apache Web server for several reasons: Running Tomcat with the Apache server provides a more robust Web serving environment, and the Apache Web server is required if you wish to install and run the [Metacat Registry](#).

This section contains instructions for installing and configuring the Apache Web server for Metacat on an Ubuntu/Debian system. Instructions for configuring Apache running on other Linux systems are included at the end of this section.

- 1) Install the Apache and Mod JK packages (Mod JK is the module Apache uses to talk to Tomcat applications) by typing:

```
sudo apt-get install apache2 libapache2-mod-jk
```

If you are installing the Apache server on an Ubuntu/Debian system, and you installed Apache using apt-get as described above, the Metacat code will have helper files that can be dropped into directories to configure Apache. Depending on whether you are installing from binary distribution or source, these helper files will be in one of two locations:

- the directory in which you extracted the distribution (for binary distribution)
- `<metacat_code_dir>/src/scripts` (for both the source distribution and source code checked out from SVN)

We will refer to the directory with the helper scripts as `<metacat_helper_dir>` and the directory where Apache is installed (e.g., `/etc/apache2/`) as `<apache_install_dir>`.

- 2) Set up Mod JK apache configuration by typing:

```
sudo cp <metacat_helper_dir>/jk.conf <apache_install_dir>/mods-available
sudo cp <metacat_helper_dir>/workers.properties <apache_install_dir>
```

- 3) Disable and re-enable the Apache Mod JK module to pick up the new changes:

```
sudo a2dismod jk
sudo a2enmod jk
```

- 4) Apache needs to know about the Metacat site. The helper file named "knb" has rules that tell Apache which traffic to route to Metacat. Set up the knb (Metacat) site by dropping the knb file into the `sites-available` directory and running `a2ensite` to enable the site:

```
sudo cp <metacat_helper_dir>/knb <apache_install_dir>/sites-available
sudo a2ensite knb
```

- 5) Restart Apache to bring in changes by typing:

```
sudo /etc/init.d/apache2 restart
```

Configuring Apache on an OS other than Ubuntu/Debian

If you are running on an O/S other than Ubuntu/Debian (e.g., Fedora Core or RedHat Linux) or if you installed the Apache source or binary, you must manually edit the Apache configuration file, where <apache_install_dir> is the directory in which Apache is installed:

```
<apache_install_dir>/conf/httpd.conf
```

- 1) Configure the log location and level for Mod JK. If your configuration file does not already have the following section, add it and set the log location to any place you'd like:

```
<IfModule mod_jk.c>
JkLogFile "/var/log/tomcat/mod_jk.log"
JkLogLevel info
</IfModule>
```

- 2) Configure apache to route traffic to the Metacat application. ServerName should be set to the DNS name of the Metacat server. ScriptAlias and the following Directory section should both point to the cgi-bin directory inside your Metacat installation.

```
<VirtualHost XXX.XXX.XXX.XXX:80>
DocumentRoot /var/www
ServerName dev.nceas.ucsb.edu
ErrorLog /var/log/httpd/error_log
CustomLog /var/log/httpd/access_log common
ScriptAlias /cgi-bin/ "/var/www/cgi-knb/"
<Directory /var/www/cgi-knb/>
    AllowOverride None
    Options ExecCGI
    Order allow,deny
    Allow from all
</Directory>
ScriptAlias /knb/cgi-bin/ "/var/www/webapps/knb/cgi-bin/"
<Directory "/var/www/webapps/knb/cgi-bin/">
    AllowOverride None
    Options ExecCGI
    Order allow,deny
    Allow from all
</Directory>
JkMount /knb ajp13
JkMount /knb/* ajp13
JkMount /knb/metacat ajp13
JkUnMount /knb/cgi-bin/* ajp13
JkMount /*.jsp ajp13
JkMount /metacat ajp13
JkMount /metacat/* ajp13
</VirtualHost>
```

- 3) Copy the "workers.properties" file provided by Metacat into your Apache configuration directory (<apache_install_dir>/conf/). Depending on whether you are installing from binary distribution or source, the workers.properties file will be in one of two locations:
 - the directory in which you extracted the Metacat distribution (for binary distribution)
 - <metacat_code_dir>/src/scripts/workers.properties (for both the source distribution and source code checked out from SVN)
- 4) Edit the workers.properties file and make sure the following properties are set correctly:
 - workers.tomcat_home - set to the Tomcat install directory on your system.
 - workers.java_home - set to the Java install directory on your system.
- 5) Restart Apache to bring in changes by typing:


```
sudo /etc/init.d/apache2 restart
```

2.2.3.4 PostgreSQL Database (or Oracle 8i)

Metacat has been most widely tested with PostgreSQL and we recommend using it. Instructions for installing and configuring Oracle 8i are also included at the end of this section.

To install and configure PostgreSQL:

- 1) If you are running Ubuntu/Debian, get PostgreSQL by typing:

```
sudo apt-get install postgresql
```

On other systems, install the rpms for postgres.

- 2) Start the database by running:

```
/etc/init.d/postgresql start
```

- 3) Change to postgres user:

```
sudo su - postgres
```

- 4) Set up an empty Metacat database instance by editing the postgresSQL configuration file:

```
gedit /etc/postgresql/8.3/main/pg_hba.conf
```

Add the following line to the configuration file:

```
host metacat metacat 127.0.0.1 255.255.255.255 password
```

Save the file and then create the Metacat instance:

```
createdb metacat
```

5) Log in to PostgreSQL by typing:

```
psql metacat
```

6) At the psql prompt, create the Metacat user by typing:

```
CREATE USER metacat WITH UNENCRYPTED PASSWORD 'your_password';
```

where 'your_password' is whatever password you would like for the Metacat user.

7) Exit PostgreSQL by typing

```
\q
```

8) Restart the PostgreSQL database to bring in changes:

```
/etc/init.d/postgresql-8.3 restart
```

9) Log out of the postgres user account by typing:

```
logout
```

10) Test the installation and Metacat account by typing:

```
psql -U metacat -W -h localhost metacat
```

11) Log out of PostgreSQL:

```
\q
```

The Metacat servlet automatically creates the required database schema. For more information about configuring the database, please see [Database Configuration](#).

Installing and Configuring Oracle 8i

To use Oracle 8i with Metacat, the Oracle RDBMS must be installed and running as a daemon on the system. In addition the JDBC listener must be enabled. Enable it by logging in as an Oracle user and typing:

```
lsnrctl start
```

Your instance should have a table space of at least 5 MB (10 MB or higher recommended). You must also create and enable a username specific to Metacat. The Metacat user must have most normal permissions including: CREATE SESSION, CREATE TABLE, CREATE INDEX, CREATE TRIGGER, EXECUTE PROCEDURE, EXECUTE TYPE, etc. If an action is unexplainably rejected by Metacat, the user permissions are (most likely) not correctly set.

The Metacat servlet automatically creates the required database schema. For more information, please see [Database Configuration](#).

2.2.3.5 Apache Jakarta-Ant (if building from Source)

If you are building Metacat from a source distribution or from source code checked out from SVN, Ant is required. (Users installing Metacat from the binary distribution do not require it.) Ant is a Java-based build application similar to Make on UNIX systems. It takes build instructions from a file named "build.xml", which is found in the root installation directory. Metacat source code comes with a default "build.xml" file that may require some modification upon installation.

If you are running Ubuntu/Debian, get Ant by typing:

```
sudo apt-get install ant
```

Otherwise, get Ant from [The Apache Ant homepage \(http://ant.apache.org/\)](http://ant.apache.org/).

Ant should be installed on your system and the "ant" executable shell script should be available in the user's path. The latest Metacat release was tested with Ant 1.6.5.

2.2.4 Installing Metacat

Instructions for a [new install](#), an [upgrade](#), and a [source install](#) are included below.

2.2.4.1 New Install

Before installing Metacat, please ensure that [all required applications](#) are installed, configured to run with Metacat, and running correctly. If you are upgrading an existing Metacat servlet, please skip to [Upgrade](#). For information about installing from source, skip to [Source Install and Upgrade](#).

To install a new Metacat servlet:

- 1) Create the Metacat directory. Metacat uses a base directory to store data, metadata, temporary files, and configuration backups. This directory should be outside of the Tomcat application directory so that it will not get wiped out during an upgrade. Typically, the directory is '/var/metacat', as shown in the instructions. If you choose a different location, remember it. You will be asked to configure Metacat to point to the base directory at startup.

Create the Metacat directory by typing:

```
sudo mkdir /var/metacat
```

- 2) Change the ownership of the directory to the user that will start Tomcat by typing:

```
sudo chown -R <tomcat_user> /var/metacat
```

Note: If you are starting Tomcat as the root user, you do not need to run the chown command.

- 3) Install the Metacat WAR in the Tomcat web-application directory. For instructions on downloading the Metacat WAR, please see [Downloading Metacat](#). Typically, Tomcat will look for its application files (WAR files) in the <tomcat_home>/webapps directory (e.g., /usr/share/tomcat5.5/webapps). Your instance of Tomcat may be configured to look in a different directory. We will refer to the Tomcat application directory as <tomcat_app_dir>. To install the Metacat WAR:

```
sudo cp <metacat_package_dir>/knb.war <tomcat_app_dir>
```


NOTE: The name of the WAR file (e.g., knb.war) provides the application context, which appears in the URL of the Metacat (e.g., <http://yourserver.com/knb/>). To change the context, simply change the name of the WAR file to the desired name.

- Restart Tomcat. Log in as the user that runs your Tomcat server (often "tomcat") and type:

```
/etc/init.d/tomcat5.5 restart
```

Congratulations! You have now installed Metacat. If everything is installed correctly, you should see the Authentication Configuration screen (Figure 2.1) when you type <http://yourserver.com/yourcontext/> (e.g., <http://knb.ecoinformatics.org/knb>) into a browser. For more information about configuring Metacat, please see [Section 3](#).

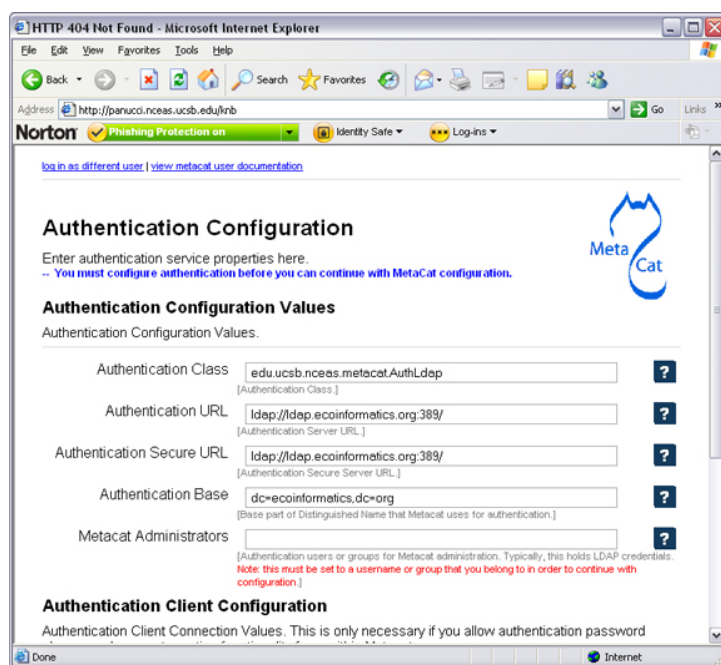


Figure 2.1: The Authentication Configuration screen appears the first time you open a new installation of Metacat. For more information about these settings, please see [Authentication Configuration](#).

2.2.4.2 Upgrade Metacat

To upgrade an existing binary Metacat installation follow the steps in this section. The steps for upgrading Metacat from source are the same as the instructions for [installing from source](#):

- Download and extract the new version of Metacat. For more information about downloading and extracting Metacat, please see [Downloading Metacat](#).

- 2) Stop running Metacat. To stop Metacat, log in as the user that runs your Tomcat server (often "tomcat") and type:

```
/etc/init.d/tomcat5.5 stop
```

- 3) Back up the existing Metacat installation. Although not required, we highly recommend that you back up your existing Metacat to a backup directory (<backup_dir>) before installing a new one. You can do so by typing:

```
cp <web_app_dir>/knb <backup_dir>/knb.<yyyymmdd>
cp <web_app_dir>/knb.war <backup_dir>/knb.war.<yyyymmdd>
```

Warning: Do *not* backup the files in the <web_app_dir> directory. Tomcat will try to run the backup copy as a service.

- 4) Copy the new Metacat WAR file in to Tomcat applications directory:

```
sudo cp <metacat_package_dir>/knb.war <tomcat_app_dir>
```

Note: Typically, Tomcat will look for its application files (WAR files) in the <tomcat_home>/webapps directory. Your instance of Tomcat may be configured to look in a different directory.

- 5) If you have been (or would like to start) running an LSID server, copy the new authority.war file to the Tomcat applications directory. For more information about the LSID server, please see [Optional Installation Options](#).

```
sudo cp <metacat_package_dir>/authority.war <tomcat_app_dir>
```

- 6) Restart Tomcat (and Apache if you have Tomcat integrated with it). Log in as the user that runs your Tomcat server (often "tomcat"), and type:

```
/etc/init.d/tomcat5.5 restart
```

- 7) Run your new Metacat servlet. Go to a Web browser and type:

```
http://yourserver.yourdomain.com/yourcontext/
```

You should substitute your context name for "yourcontext" in the URL above (your context will be "knb" unless you change the name of the knb.war file to something else). If everything is working correctly, you should be presented with Metacat's [Authorization Configuration](#) screen. Note that if you do not have Tomcat integrated with Apache you will probably have to type `http://yourserver.yourdomain.com:8080/yourcontext/`

2.2.4.3 Source Install and Upgrade

Whether you are building Metacat from the source distribution or source code checked out from SVN, you will need Apache Ant to do the build (see [Installing and Configuring Required Software](#) for more information about Ant).

To install Metacat from source:

- 1) Edit the `build.properties` file found in the directory in which you downloaded Metacat. Note: Throughout the instructions, we will refer to this directory as `<metacat_src_dir>`.
 - a) Set the `build.tomcat.dir` property to your Tomcat installation directory. Metacat will use some of the native Tomcat libraries during the build. For instance: `build.tomcat.dir=/usr/local/tomcat`
 - b) Set the `app.deploy.dir` property to your application deployment directory. For instance: `app.deploy.dir=/usr/local/tomcat/webapps`
- 2) In the `<metacat_src_dir>`, run:

```
sudo ant clean install
```

You will see the individual modules get built. You should see a "BUILD SUCCESSFUL" message at the end.

You should see a new file named `knb.war` in your application deployment directory.

To run your new Metacat servlet, open a Web browser and type:

<http://yourserver.yourdomain.com/yourcontext/> (e.g. <http://knb.ecoinformatics.org/knb/>)

Your context will be "knb" unless you changed the name of the `knb.war` file to something else. The servlet may require a few seconds to start up, but once it is running, you will be presented with the [Authorization Configuration](#) screen.

2.2.5 Optional Installation Options (LSID Server)

Metacat's optional LSID server allows Metacat to use a standardized syntax for identifying data sets, in addition to Metacat's internal, custom scheme for identifiers. LSID's were designed to identify complex biological entities with short identifiers (much

like DOIs in publishing) that are both computer and human readable. LSID identifiers are URIs and are therefore usable in many Internet applications, but they also cleanly separate the identity of a data set (i.e., its permanent identifier) from its current location (e.g., the list of URLs from which it might be retrieved). LSIDs accomplish this by using a level of indirection; the identifier represents simply a name without location, but an associated resolver service can be used to locate the current location of the data and metadata for the data set. This is accomplished by establishing a well-known location for the resolution service for each authority using an infrequently used feature of the domain name system called SRV records. At its most basic, resolution of an identifier is performed when a client looks up the SRV record for an LSID by querying DNS, which returns the current host and port of the authority web service, which is in turn used to locate the data and metadata.

Using LSIDs to identify data records is being debated among members of the Taxonomic Databases Working Group (TDWG). There are several alternate technologies that are under consideration (e.g., DOI, plain http URIs), and so at this time the support for LSIDs in Metacat has been created on an experimental basis only. If the LSID approach is ratified by the broader community, we will expand support for LSIDs in Metacat, but until then it is an optional and experimental feature.

The format of an LSID is:

```
urn:lsid:<Authority>:<Namespace>:<ObjectID>[:<Version>]
e.g., urn:lsid:ecoinformatics.org:tao:12039:1
```

When you enable the Metacat LSID support, you can use LSID clients (such as LSID Launchpad) and LSID notation to query Metacat for data and metadata. LSID notation can be used directly in Metacat HTTP queries as well. For example, a data package with an ID `tao.12039.1` that is stored in a Metacat available at: `http://example.com:9999` can be accessed by the following HTTP Metacat queries:

```
http://example.com:9999/authority/data?lsid=urn:lsid:ecoinformatics.org:tao:12039:1
```

(To return the data)

```
http://example.com:9999/authority/metadata?lsid=urn:lsid:ecoinformatics.org:tao:12039:1
```

(To return the metadata)

Notice that in the HTTP query strings, the periods in the data package ID have been replaced with colons. The authority (`ecoinformatics.org`) must be properly configured by the Metacat administrator. Note: In order to configure the authority, you must have access to the DNS server for the Metacat domain. Further instructions are provided below.

To install and configure the LSID Server with Metacat:

To install the LSID server using the binary installation:

- 1) Copy the authority.war file to Tomcat

```
sudo cp <metacat_package_directory>/authority.war  
/usr/share/tomcat5.5/webapps
```

- 2) Set up the LSID server by dropping the authority file into Apache's sites-available directory and running a2ensite to enable the site:

```
sudo cp <metacat_helper_dir>/authority /etc/apache2/sites-available  
sudo a2ensite authority
```

- 3) Restart Tomcat. Log in as the user that runs your Tomcat server (often "tomcat") and type:

```
/etc/init.d/tomcat5.5 restart
```

- 4) Restart Apache to bring in changes by typing:

```
sudo /etc/init.d/apache2 restart
```

- 5) See notes beneath LSID server source installation for instructions for modifying the SRV record(s)

To install the LSID server from a source installation:

- 1) In the build.properties file found in the directory into which you extracted the Metacat source code, set the authority and config.lsidauthority properties. For example:

```
authority.context=authority  
config.lsidauthority=ecoinformatics.org
```

- 2) In the <metacat-src-directory> create the authority.war by running:

```
sudo ant war-lsid
```

- 6) Copy the LSID WAR file into the Tomcat application directory.

```
sudo cp <metacat_package_dir>/dist/authority.war  
<tomcat_app_dir>
```

- 7) Restart Tomcat. Log in as the user that runs your Tomcat server (often "tomcat") and type:

```
/etc/init.d/tomcat5.5 restart
```

- 8) If you are running Tomcat behind the Apache server (the recommended configuration), set up and enable the authority service site configurations by typing:

```
sudo cp <metacat_helper_dir>/authority <apache_install_dir>/sites-available
sudo a2ensite authority
```

Where <metacat_helper_dir> can be found in <metacat_code_dir>/src/scripts

- 9) Restart Apache to bring in changes by typing:

```
sudo /etc/init.d/apache2 restart
```

Once the authority.war is installed, you must also modify the SRV record(s) on the DNS server for the domain hosting the Metacat. The record should be added to the master zone file for the metacat's DNS server:

```
_lsid._tcp      IN      SRV     1       0       8080
<metacat.edu>.
```

Where <metacat.edu> is the name of the machine that will serve as the physical location of the AuthorityService.

For example, the value of <metacat.edu> for the below example URL would be example.com:

```
http://example.com:9999/authority/data?lsid=urn:lsid:ecoinformatics.org:tao:12039:1
```

For more information, please see <http://www.ibm.com/developerworks/opensource/library/os-lsid/>

2.2.6 Troubleshooting

We keep and update a list of common problems and their solutions on the KNB website. See <http://knb.ecoinformatics.org/software/metacat/troubleshooting.html> for more information.

2.3 Installing on Windows

Metacat can be installed on Windows. Please follow the instructions in this section for [downloading Metacat](#), [installing the required software](#), and [installing Metacat](#). Note that Registry and Data Upload functionality has not been tested on Windows.

2.3.1 Download Metacat

To obtain a Metacat WAR file, which is used when installing the Metacat servlet:

1. Browse to the [KNB Software Download Page](#). In the Metacat section, select the link that looks like: `metacat-bin-X.X.X.zip`, where X.X.X is the latest version of Metacat (e.g., 1.9.0).
2. Choose to download and Save the file locally.
3. Extract the Metacat package files using your Windows zip utility. You should see a WAR file and several supporting files (we will only use the WAR file when installing Metacat).

Note: The location where these files were extracted will be referred to as the `<metacat_package_dir>` for the remainder of this documentation.

Note: Before installing Metacat, please ensure that [all required software](#) is installed and running correctly.

2.3.2 Install Required Software

Before you can install and run Metacat, you must ensure that a recent Java SDK, PostgreSQL and [Tomcat](#) are installed, configured, and running correctly.

- [Java 6](#)
- [Tomcat](#)
- [PostgreSQL Database](#)

2.3.2.1 Java 6

To run Metacat, you must have Java 6. (Java 5 is deprecated). Make sure that the `JAVA_HOME` environment variable is properly set and that both `java` and `javac` are on your `PATH`.

To download and install Java:

1. Browse to: <http://java.sun.com/javase/downloads/widget/jdk6.jsp> and follow the instructions to download JDK 6.

2. Run the downloaded installer to install Java.
3. Set the JAVA_HOME environment variable: In "My Computer" properties, go to "advanced settings > environment variables". Add:
System Variable: JAVA_HOME C:\Program Files\Java\jdk1.6.0_18 (or whichever version you downloaded)

2.3.2.2 Tomcat

We recommend that you install Tomcat version 5.5.

To download and install Tomcat:

- 1) Browse to: <http://tomcat.apache.org/download-55.cgi>
- 2) Download the Tomcat core zip file
- 3) Extract Tomcat files to C:\Program Files\tomcat using the windows zip utility.

2.3.2.3 PostgreSQL Database

Metacat can be run with any SQL92-compliant RDBMS, but it has been most widely tested with PostgreSQL. Instructions for installing and configuring PostgreSQL for use with Metacat are included in this section.

To download and install PostgreSQL:

- 1) Browse to <http://www.postgresql.org/download/windows> and download the one-click installer
- 2) Run the installer
- 3) Edit C:\Program Files\PostgreSQL\8.3\data and add:

```
host metacat metacat 127.0.0.1 255.255.255.255 password
```

- 4) Create a super user. At the command line, go to C:\Program Files\PostgreSQL\8.3\bin and run:

```
createdb -U postgres metacat (enter super user password)
```

- 5) Log in to PostgreSQL:

```
psql -U postgres metacat (enter super user password)
```


- 6) Create a Metacat user:

```
CREATE USER metacat WITH UNENCRYPTED PASSWORD 'your_password'
```

- 7) Exit PostgreSQL:

```
\q
```

- 8) Restart PostgreSQL from the start menu by selecting:

```
run start/All Programs/PostgreSQL 8.3/Stop Server  
run start/All Programs/PostgreSQL 8.3/Start Server
```

- 9) Test the installation by logging in as the metacat user:

```
Psql -U metacat -W -h localhost metacat
```

- 10) Exit PostgreSQL:

```
\q
```

The Metacat servlet automatically creates the required database schema. For more information, please see [Database Configuration](#).

2.3.3 Installing Metacat

Instructions for a [new install](#) and for an [upgrade](#) are included below.

2.3.3.1 New Install

Before installing Metacat, please ensure that [all required applications](#) are installed, configured to run with Metacat, and running correctly. If you are upgrading an existing Metacat servlet, please skip to [Upgrade](#).

To install a new Metacat servlet:

- 1) Create the Metacat base directory at:

```
C:/Program Files/metacat
```

- 2) Copy the Metacat WAR file to Tomcat (for information about obtaining a Metacat WAR file, see [Download Metacat](#)):

```
copy <metacat_package_dir>\knb.war C:\Program
Files\tomcat\webapps
```

- 3) Restart Tomcat:

```
C:\Program Files\tomcat\shutdown.bat
C:\Program Files\tomcat\startup.bat
```

Congratulations! You are now ready to configure Metacat. Please see Section 3, [Configuring Metacat](#) for more information.

2.3.3.2 Upgrade

To upgrade an existing Metacat installation:

- 1) Download and extract the new version of Metacat. For more information about downloading and extracting Metacat, please see [Download Metacat](#).
- 2) Back up the existing Metacat installation. Although not required, we highly recommend that you back up your existing Metacat to a backup directory (<backup_dir>) before installing a new version. You can do so by copying:

```
<web_app_dir>/knb to <backup_dir>/knb.<yyyymmdd>
```

```
<web_app_dir>/knb.war to <backup_dir>/knb.war.<yyyymmdd>
```

Warning: Do *not* backup the knb directory in the <web_app_dir> directory. Tomcat will try to run the backup copy as a service.

- 3) Copy the new Metacat WAR file in to Tomcat applications directory:

```
copy knb.war C:\Program Files\tomcat\webapps
```

- 4) Restart Tomcat:

```
C:\Program Files\tomcat\shutdown.bat
C:\Program Files\tomcat\startup.bat
```

Congratulations! You are now ready to configure Metacat. Please see [Configuring Metacat](#) for more information.

3 Configuring Metacat

When Metacat (Tomcat) is started, the Metacat servlet checks to see if it is configured. If not, Metacat will automatically send you to the configuration pages.

If the installation is new, or the previous version is before 1.9.0, pay close attention to the configuration values. If you have upgraded Metacat, and the previous version is 1.9.0 or later, Metacat will pull existing configuration settings from a backup location. You should still verify that the values are correct.

To access your Metacat, open a Web browser and type:

```
http://<your_context_url>
```

Where <your_context_url> is the URL of the server hosting the Metacat followed by the name of the WAR file (i.e., the application context) that you installed. For instance, the context URL for the KNB Metacat is: <http://knb.ecoinformatics.org/knb>

You can always open the configuration screen from within Metacat by typing:

```
http://<your_context_url>/admin
```

3.1 Initial Configurations

Before you can log in to the Metacat and configure it, you are required to confirm Metacat's [back-up location](#) and [authentication configuration](#) (if not already configured). Metacat will automatically attempt to locate an existing back-up directory, but you may need to correct the value or specify a directory (if the installation is new, or if Metacat was unable to determine the location of an existing back-up directory). The authentication configuration is required for logging in to the Metacat and for defining administrative accounts. Instructions for [changing the authentication configuration without authentication](#) are included at the end of this section.

3.1.1 Back-up Configuration

To preserve its configuration settings, Metacat backs up all configurations to a directory outside the application directories. Because a new installation/upgrade does not know where this external directory is, Metacat uses a discovery algorithm to locate it. If Metacat cannot identify a backup directory, you will see the Backup Directory Configuration screen (Figure 3.1).

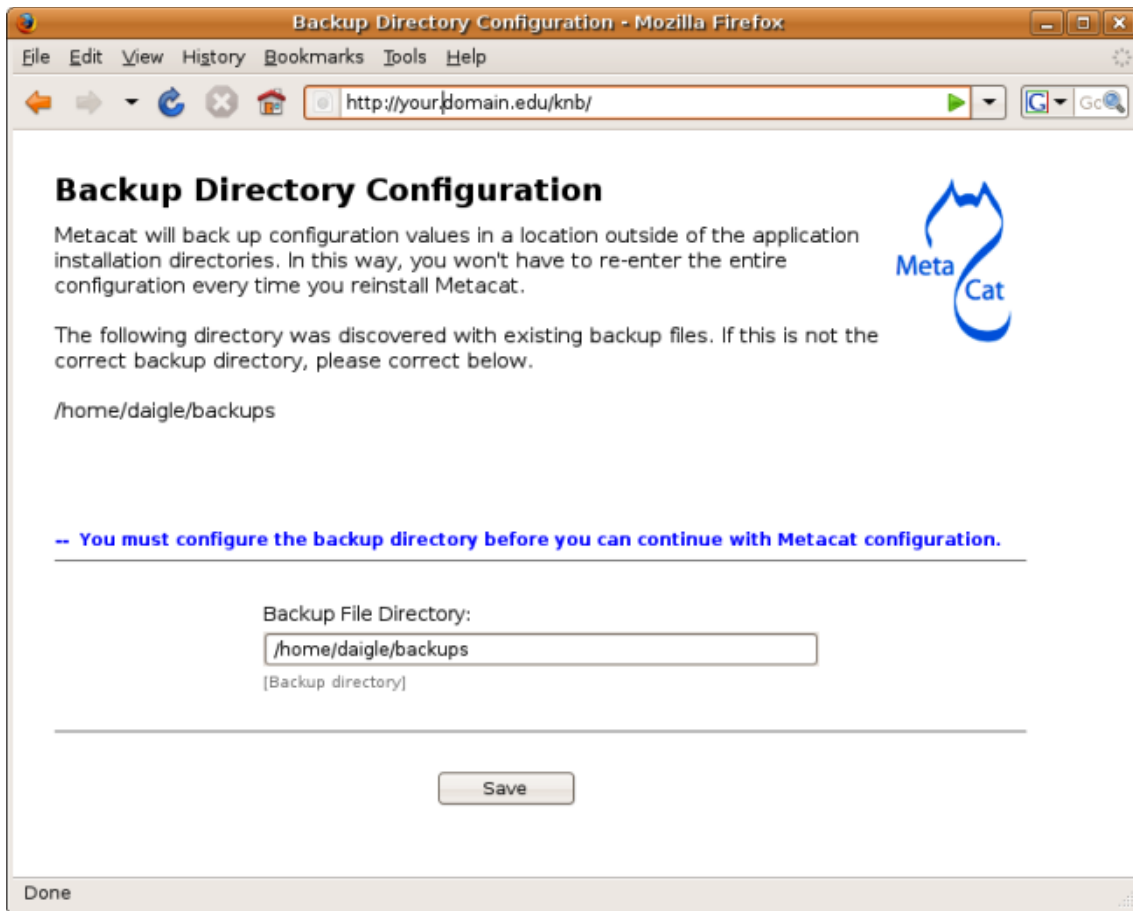


Figure 3.1: Configuring the Backup Directory.

3.1.2 Authentication Configuration

Whether you are installing or upgrading the Metacat servlet, you will automatically be sent to the Authentication Configuration page. You can also reach the Authentication Configuration page from a running Metacat by typing: `http://<your_context_url>/admin`

Metacat uses LDAP as its primary authentication mechanism, but you can define your own authentication mechanism by [creating a Java class that implements AuthInterface](#). Required configuration values are: Authentication Class, Authentication URL, Authentication Secure URL, and Metacat Administrators (Figure 3.2). Make sure that your user account information is entered into the Metacat Administrators field (e.g., `uid=daigle,o=nceas,dc=ecoinformatics,dc=org`). You will not be allowed to continue with configuration if this is missing.

NOTE: To create an LDAP account on the KNB LDAP server (specified as the default LDAP server), go to <http://knb.ecoinformatics.org> and select the "create a new user account" link.

If you make changes to the authentication settings, you must restart Tomcat to put them into effect.

Authentication Configuration - Mozilla Firefox


File Edit View History Bookmarks Tools Help

http://your.domain.edu/knb/

[log in as different user](#) | [view metacat user documentation](#)

Authentication Configuration

Enter authentication service properties here.



-- You must configure authentication before you can continue with MetaCat configuration.

Authentication Configuration Values

Authentication Class ?
[Authentication Class.]

Authentication URL ?
[Authentication Server URL.]

Authentication Secure URL ?
[Authentication Secure Server URL.]

MetaCat Administrators ?
[Authentication users or groups for Metacat administration. Typically, this holds LDAP credentials. **Note: this must be set to a username or group that you belong to in order to continue with configuration.**]

You are logged in as: test-user
[log in as different user](#) | [view metacat user documentation](#)

Done

Figure 3.2: Configuring Authentication Values.

3.1.3 Changing Authentication Configuration without Authentication

If you need to change or add authentication information and cannot authenticate using the existing authentication settings (e.g., the existing Metacat administrator is no longer available or you forgot the administrator password), you must edit the Metacat configuration file by hand. This ensures that only a person who has access to the Metacat server and the configuration files on that server will be able to change the administrator accounts.

To edit the authentication configuration file:

- 1) Stop Tomcat and edit the Metacat properties (metacat.properties) file in the Metacat context directory inside the Tomcat application directory. The Metacat context directory is the name of the application (usually knb):

```
<tomcat_app_dir>/<context_dir>/WEB-INF/metacat.properties
```

- 2) Change the following properties appropriately:
 - auth.administrators - a colon separated list of administrators
 - auth.url - the authentication server URL
 - auth.surl - the authentication secure server URL
- 3) Save the metacat.properties file and start Tomcat.

3.2 Logging in to Metacat

In order to configure Metacat, you must log in with an administrative account that has been configured in the [Authentication Configuration](#) settings. If you did not set up the correct administrative user there, you must change the [authentication configuration by hand](#) before you can log in.

In the log-in screen (Figure 3.3) enter your user name and password and click the "Login" button.

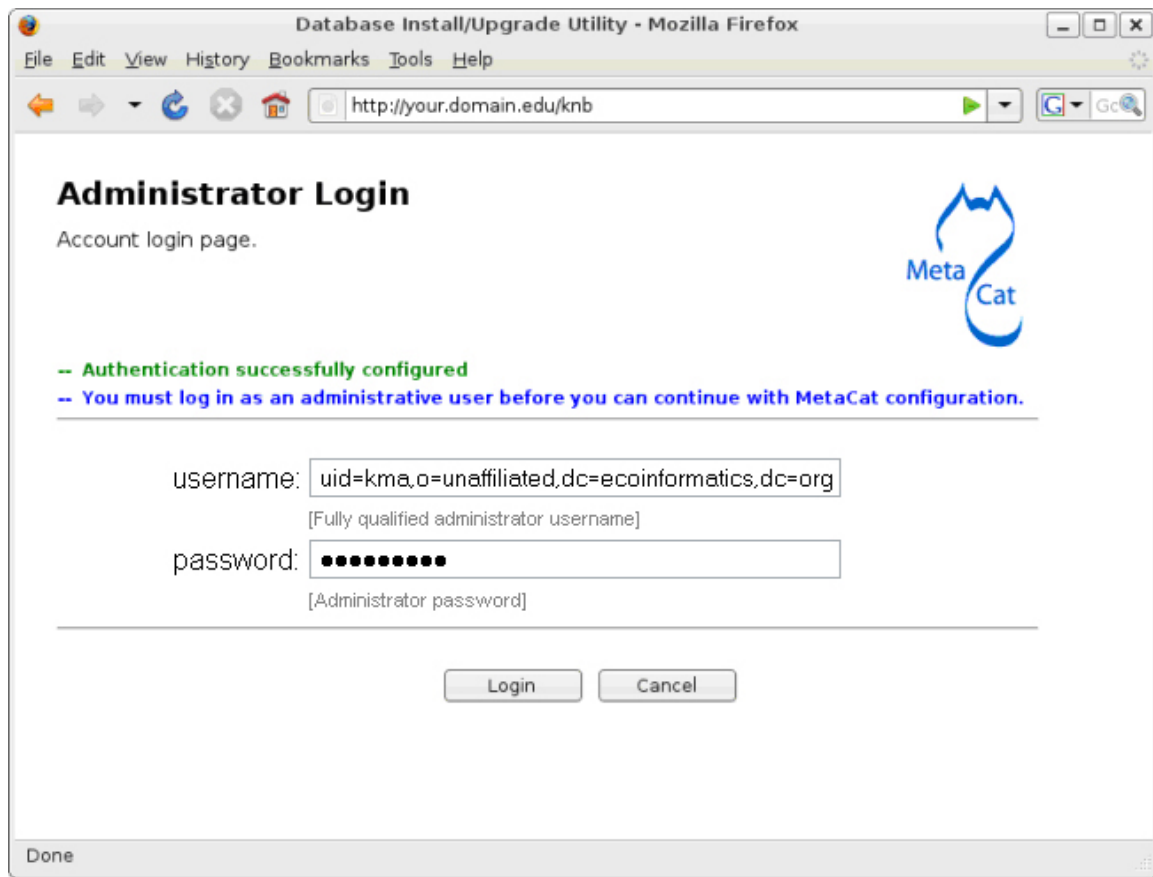


Figure 3.3: Logging in to Metacat

3.3 Required Configurations

All required Metacat settings can be accessed from the Metacat Configuration utility (Figure 3.4), which becomes available after the **initial configurations** have been specified and an authorized administrator logs in.

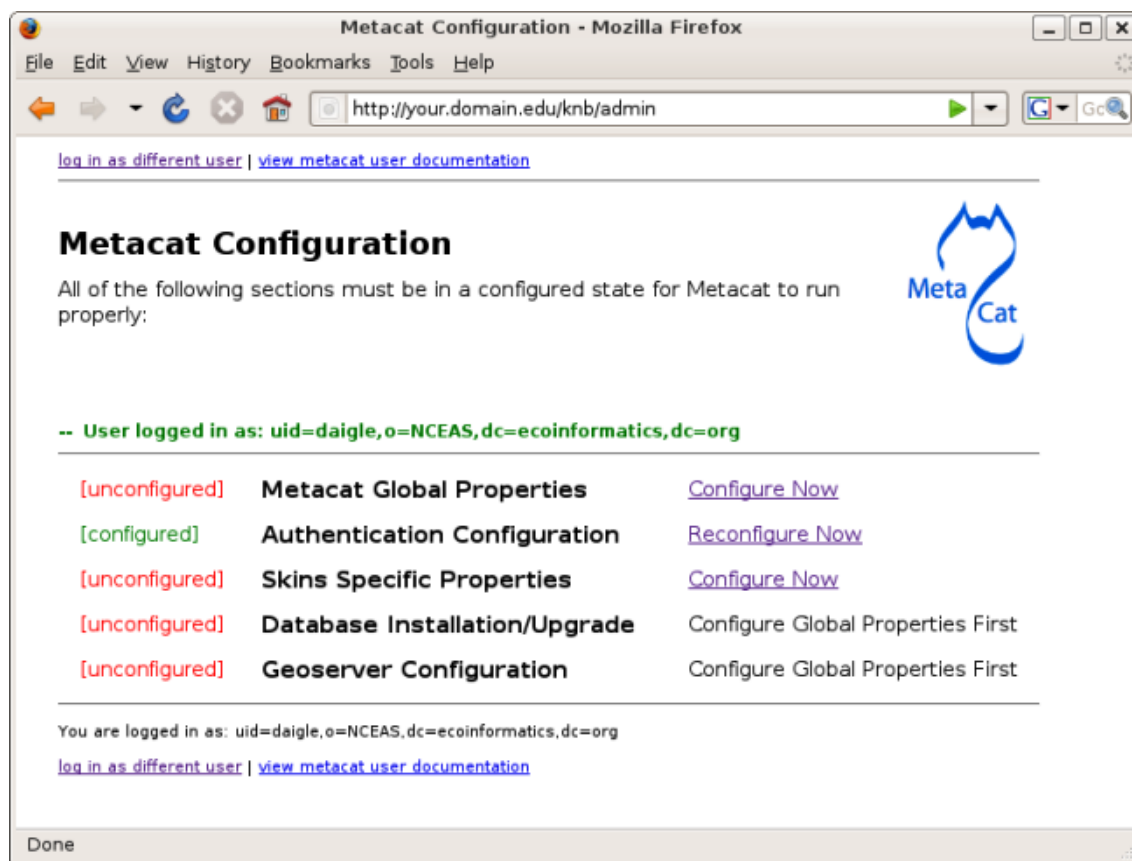


Figure 3.4: Required Metacat configurations.

The configuration settings are grouped into five sections (Metacat Global Properties, Authentication Configuration, Skins Specific Properties, Database Installation/Upgrade, Geoserver Configuration), each of which is listed with its current status (Table 3.1)

Status	Description
[unconfigured]	The section has yet to be configured
[configured]	The section has been configured.
[bypassed]	Used only for the Geoserver configurations. The administrator can choose not to configure the Geoserver user/password.

Table 3.1: Possible configuration statuses.

To the right of each configuration section is one of the following options: Configure Now, Reconfigure Now, Configure Global Properties First, or Version:X.X.X. If the option is linked (e.g., Configure Now or Reconfigure Now), you can select the link to open the associated configuration settings and add or edit them, respectively. If the option

is not linked (e.g., Configure Global Properties First), the settings cannot be specified until the global properties are set. Once the global properties are configured, the option to configure this section becomes available. The Version:X.X.X option is used only for the Database Installation/Upgrade section. If the database schema version detected by Metacat matches the application version (eg, 1.9.0), then no further database configuration is required.

All settings must be in a configured or bypassed state in order to run Metacat (Figure 3.5). For new installations or upgrades, click the "go to metacat" link that appears after configuration is complete to go directly to Metacat. Note that Metacat indexes at start-up time, so the initial start-up may take some time depending on the amount of data in your database. If you are reconfiguring a running version of Metacat, you must restart the Tomcat server for the changes to take effect.

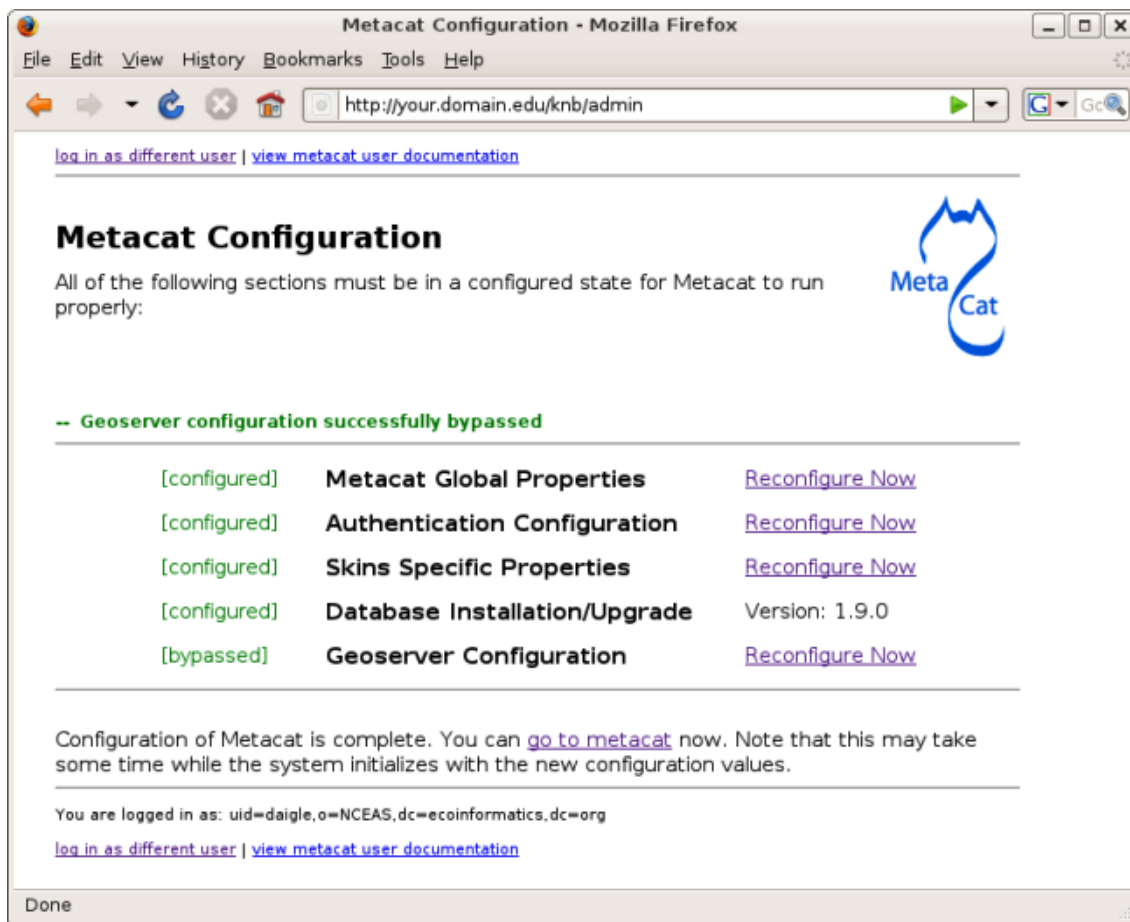


Figure 3.5: The Metacat settings as they appear after configured.

3.3.1 Global Properties (server, ports, etc)

The Metacat configurations included under Global Properties represent the bulk of the settings required to run Metacat (Figure 3.6). Click a blue question-mark icon beside any

setting for detailed instructions. More information about each property is also included in the [Metacat Properties Appendix](#).

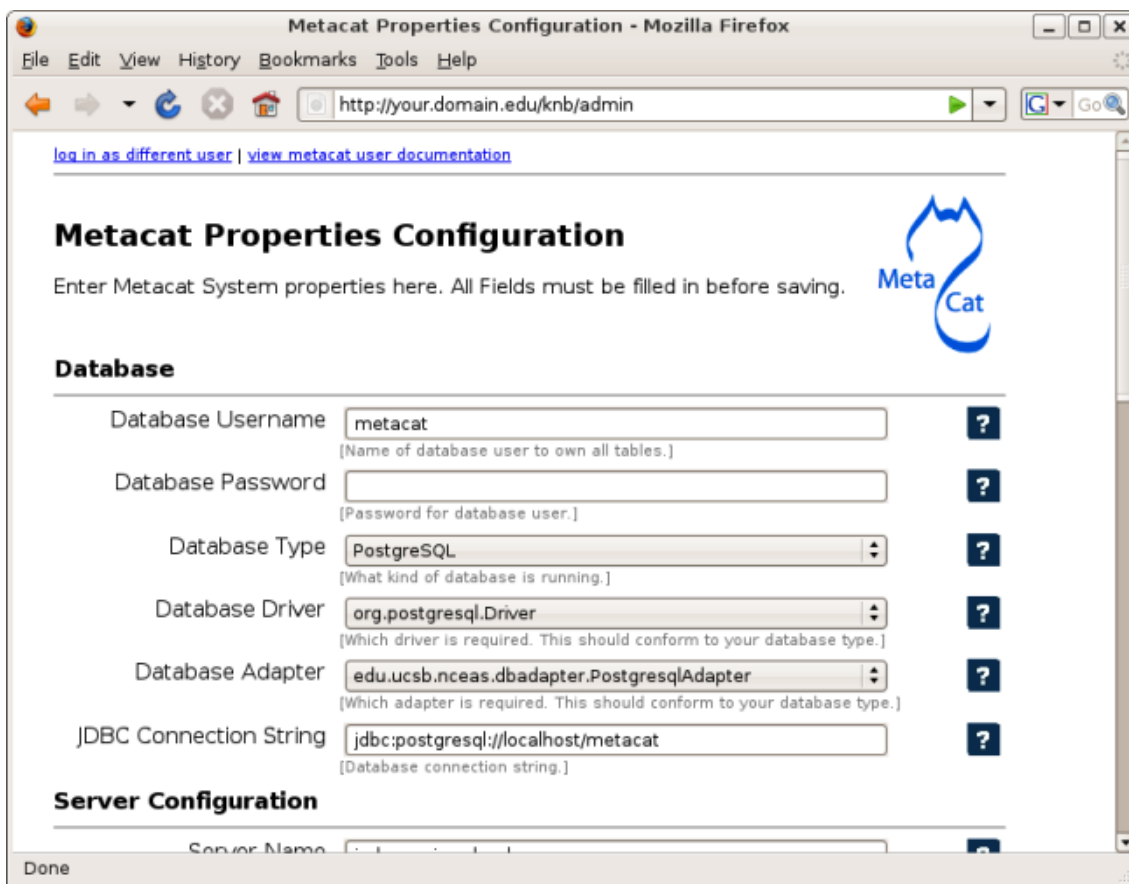


Figure 3.6: Setting Metacat's Global Properties.

When you save global properties, Metacat also saves a back-up file that is located in `/var/metacat/.metacat` (on Linux) or `C:\Program Files\metacat\metacat` (on Windows). When you update Metacat, the system automatically locates the back-up file so you do not have to re-enter the configuration settings.

The first time you install Metacat, the system attempts to automatically detect the values for a number of settings (Table 3.2). It is important to ensure that these values are correct.

Property	Description
Metacat Context	The name of the deployed Metacat WAR file (minus the .war extension). E.g., "knb"
Server Name	The DNS name of the server hosting Metacat, not including port numbers or the "http://" header.
HTTP Port	The non-secure port where Metacat will be available.
HTTP SSL Port	The secure port where Metacat will be available.
Deploy Location	The directory where the application is deployed.

Table3.2: Configuration settings that Metacat will automatically detect.

3.3.2 Authentication Configuration

Because you must specify the Authentication settings before you can access the main configuration page, the settings will always be configured when you view them in the admin interface. If you wish to change the authentication settings, you must restart Metacat to put the changes into effect. For more information about the Authentication configurations, please see [Initial Configurations](#).

3.3.3 Skins Configuration (look & feel)

Customizing the look and feel of Metacat's Web interface is done via skins, which are applied in the Skins Configuration section. If you have installed the optional [Registry](#), which provides a Web interface for creating, editing, and submitting content to the Metacat, you can also choose which form fields appear in that interface and which are required. Note that if you do not have a custom skin AND you are not using the Registry, you can simply save the default configuration.

If your Metacat has a customized skin, it will appear as a choice in the Skins Configuration settings (Figure 3.7). You can create your own skins as well. For more information about creating skins, please see Section 3.4.3, [Creating a Custom Skin](#).

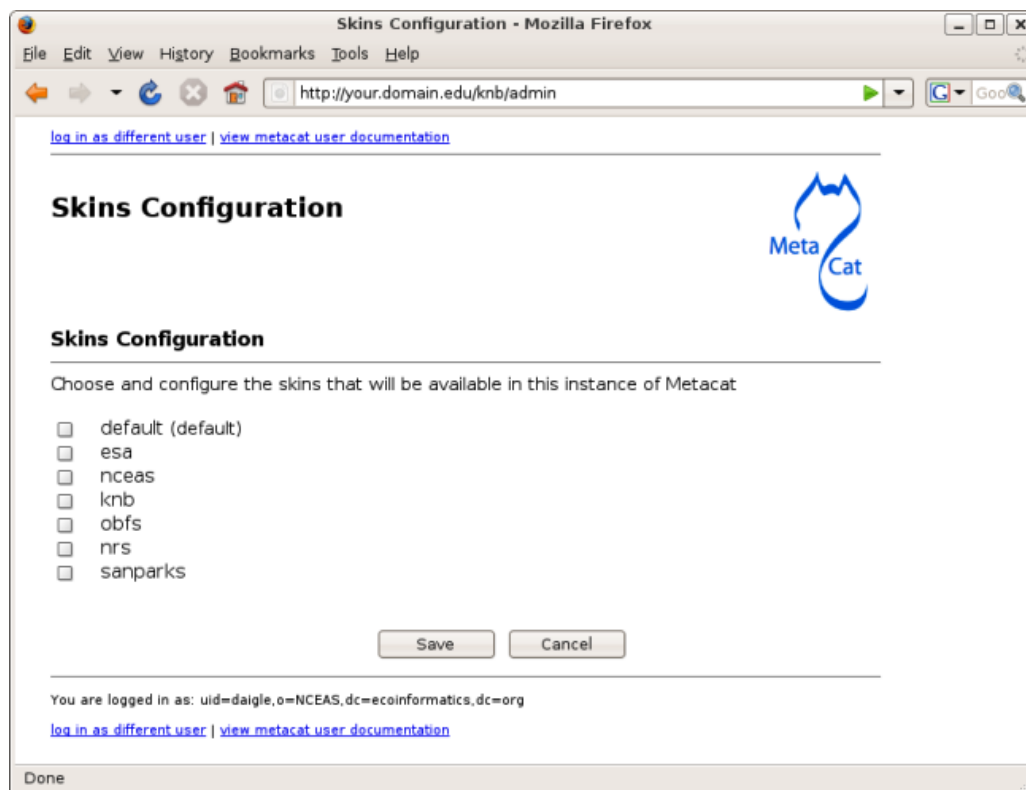


Figure 3.7: Configuring Metacat skins.

Select the checkbox next to your customized skin and click the 'Make <skin_name> default' radio button. If you do not have a custom skin, select the "default" skin. Once you have selected a skin, Metacat will open a list of options that apply to the [Registry](#) interface (Figure 3.8).

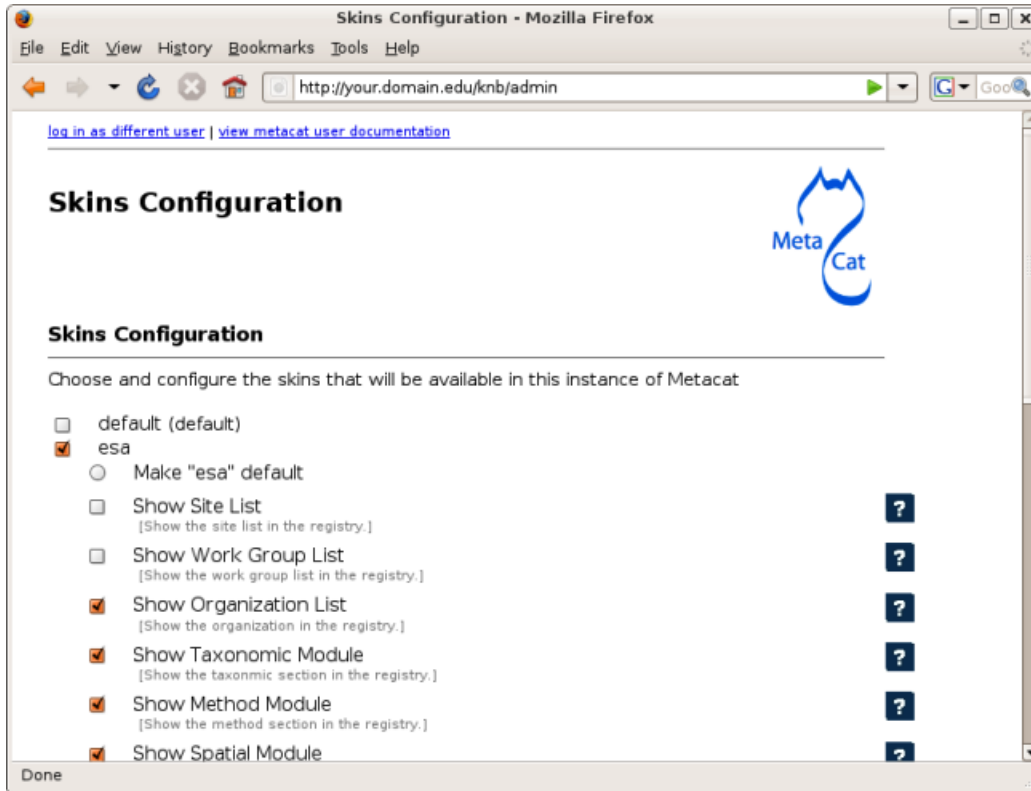


Figure 3.8: Configuring the lists and modules that will be displayed by the Registry.

Select the lists and modules that you would like to appear in the Registry form-interface by checking the box beside each. When you save the configuration, the customized interface will appear to site visitors.

3.3.4 Database Configuration

Because the Database Configuration is dependent on values specified in the Global Configuration section, the link to these settings does not become active until after the global settings have been saved. Once the global settings have been saved, Metacat automatically detects the database schema version and upgrades it if necessary (and with your permission).

- [New Installation](#)
- [Upgrade](#)

3.3.4.1 New Installation

If Metacat determines that your database is new, the Database Install/Upgrade utility lists the SQL scripts that will run in order to create a database schema for the new version of Metacat (Figure 3.9).

If the database is not new, or if you have any questions about whether it is new or not, choose Cancel and contact support at knb-help@nceas.ucsb.edu.

When you choose Continue, Metacat runs the listed scripts and create the database schema.

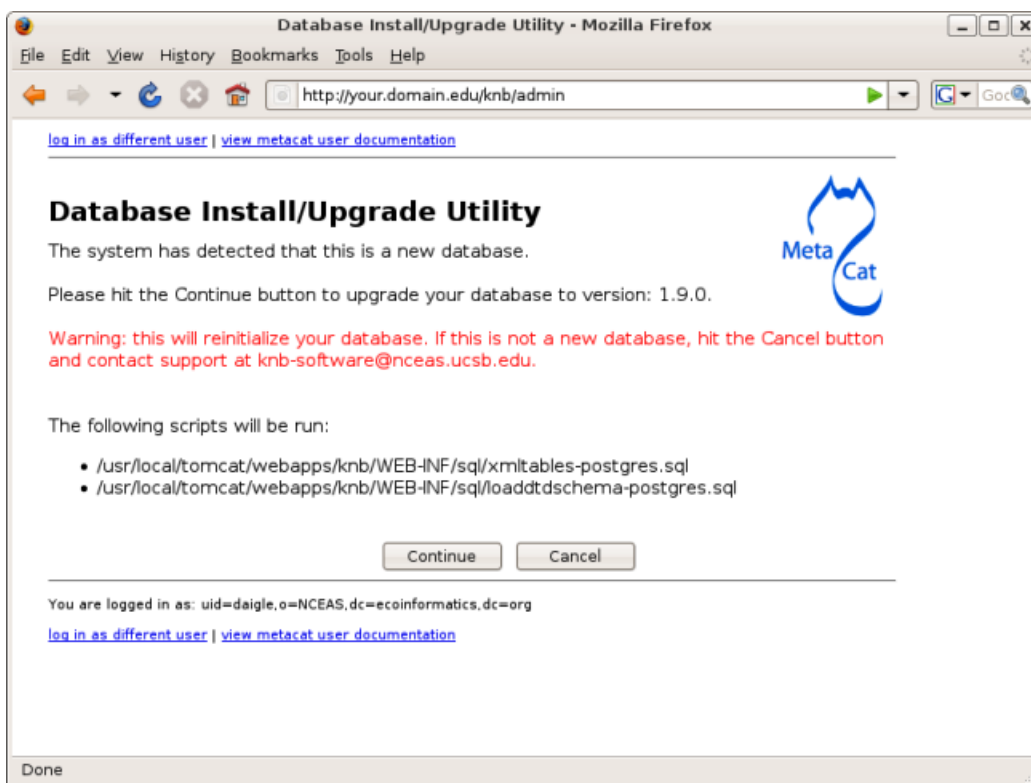


Figure 3.9: Database Install/Upgrade for new databases.

3.3.4.2 Upgrade

If Metacat identifies a previous database schema, the Database Install/Upgrade utility notes the existing version and lists the SQL scripts that will run in order to update the schema for the new version of Metacat (Figure 3.10).

If the detected schema version is incorrect, or if you have any questions about whether it is correct or not, click the Cancel button and contact support at knb-help@nceas.ucsb.edu. When you choose to continue, Metacat runs the listed scripts and updates the database schema.

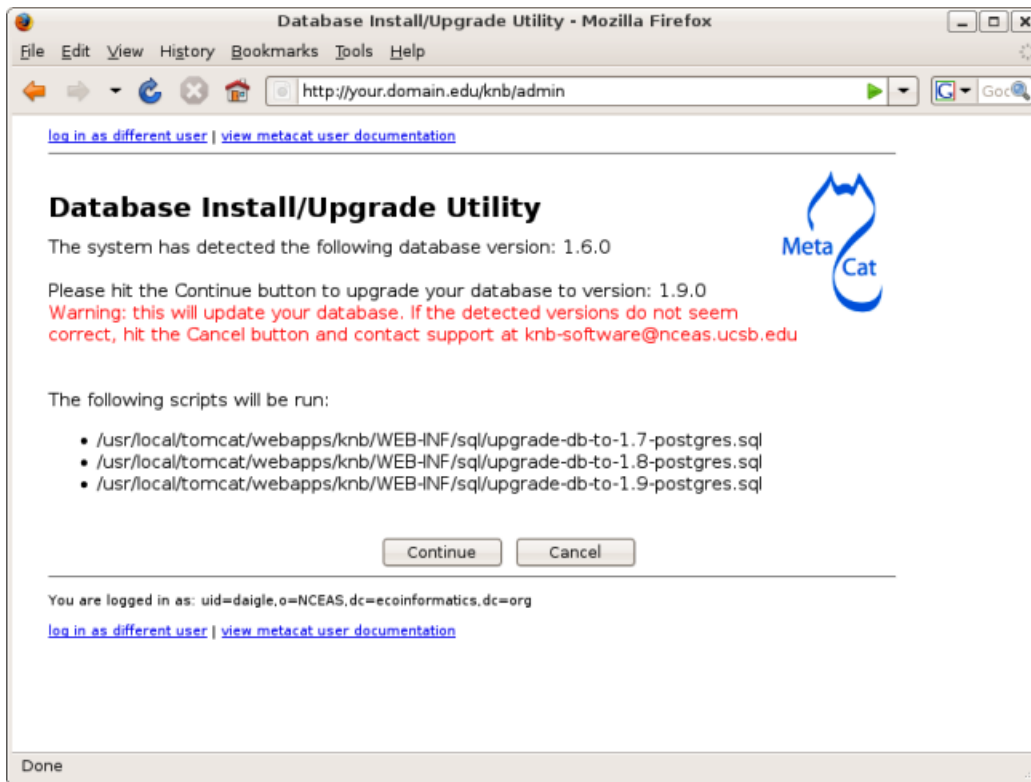


Figure 3.10: Upgrading an existing database.

3.3.5 Geoserver Configuration (Highly Recommended)

Metacat comes bundled with a Web Mapping Service called Geoserver, which converts spatial data into Web-deliverable map images. Geoserver installs with a default administrative username and password. *We highly recommend that you change the default credentials so that only local administrators can make changes to your Geoserver.* For more information about Geoserver, see [Metacat's use of Geoserver](#).

When you choose the Geoserver Configuration link from the main configuration screen, Metacat will prompt you for a few important details about your Geoserver installation. The data directory and context settings allow Geoserver and Metacat to share the same spatial data store and render maps within Metacat skins. The security configuration prompts for a new admin password (Figure 3.11). After you enter the new settings, Metacat writes the information to the Geoserver deployment.

The default settings are typically appropriate for most Metacat deployments, but if you wish to skip the Geoserver configuration, click the Bypass button. Geoserver (if deployed) will remain with a default configuration and the main Metacat configuration screen will display the "bypassed" status beside the Geoserver settings. You will be able to run Metacat, but maps will not be rendered.

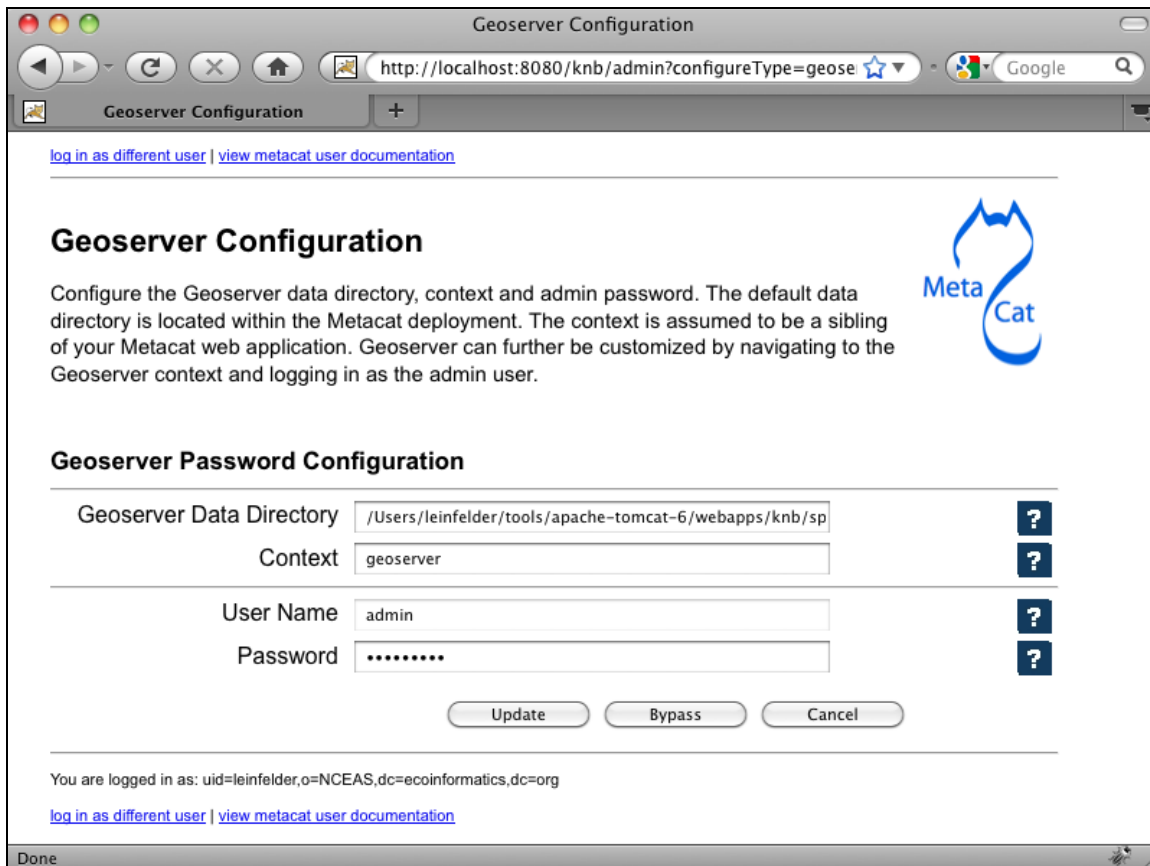


Figure 3.11: Configuring Geoserver.

Manual Geoserver Update

You can change the Geoserver username and password by directly logging in to the Geoserver. To configure the credentials manually:

- 1) Go to the Geoserver admin page: `http://<your_context_url>/geoserver/`
- 2) Log in using the default username and password (admin / geoserver)
- 3) Navigate to the Password Change Page. Enter a new user and password and click Submit.
- 4) Click Apply then Save to save your new password.

3.4 Additional Configuration

The most dynamic Metacat properties are managed and modified with the form-based [Metacat Configuration utility](#). These configuration properties can also be accessed directly (along with additional static properties) via Metacat's property files:

`metacat.properties` (which contains global properties, e.g., authorization and database values) and `<SKIN_NAME>.properties` (which contains skin-specific properties). Each of these property files is discussed in more depth in this section.

3.4.1 The `metacat.properties` file

Metacat's `metacat.properties` file contains all of Metacat's global properties, both the dynamic properties, which are managed with the [Configuration utility](#), as well as the more static properties, which can only be modified manually in this file. The `metacat.properties` file also contains optional properties that are only relevant when optional Metacat features (such as the [harvester](#) or [replication](#)) are enabled.

The `metacat.properties` file is found here:

```
<CONTEXT_DIR>/WEB_INF/metacat.properties
```

Where `<CONTEXT_DIR>` is the directory in which the Metacat application code lives (e.g., `/usr/share/tomcat5.5/webapps/knb`). The path is a combination of the Web application directory (e.g., `/usr/share/tomcat5.5/webapps/`) and the Metacat context directory (e.g., `knb`). Both values depend upon how your system was set up during installation.

For information about each property and default or example settings, please see the [Metacat Properties Appendix](#). Properties that can only be edited manually in the `metacat.properties` file are highlighted.

3.4.2 `<SKIN_NAME>.properties`

The `<SKIN_NAME>.properties` file contains skin-specific properties (e.g., template information). For each skin, the skin-specific properties are found here:

```
<CONTEXT_DIR>/style/skins/<SKIN_NAME>/<SKIN_NAME>.properties
```

Where `<CONTEXT_DIR>` is the directory in which the Metacat application code lives (described above) and `<SKIN_NAME>` is the name of the skin (e.g., `default` or `nceas`)

3.4.3 Creating a Custom Skin

To create and customize your own Metacat skin, you must first create a skin directory. This is most easily accomplished by copying one of the existing skin directories. Step-by-step directions for creating and installing a custom skin are included below:

- 1) Copy an existing skin directory. We recommend using the "default" directory.

```
sudo cp -r <CONTEXT_DIR>/style/skins/default/  
<CONTEXT_DIR>/style/skins/[yourSkin]/
```

Where `<CONTEXT_DIR>` is the directory in which the Metacat application code lives and `[yourSkin]` is the name you wish to apply to your skin.

- 2) In `[yourSkin]` directory, change all files named "default.xxx" to "yourSkin.xxx". The following files should be changed:

```
default.css  
default.js  
default.properties  
default.properties.metadata.xml  
default.xml
```

- 3) In the `metacat.properties` file(`<CONTEXT_DIR>/WEB_INF/metacat.properties`), add `[yourSkin]` to the value of the `skin.names` property.
- 4) Restart Tomcat. Log in as the user that runs your Tomcat server (often "tomcat") and type:

```
/etc/init.d/tomcat5.5 restart
```

Navigate to [Metacat's Configuration utility](#) and select the Configure Skins option. Your custom skin should appear as a choice in the skins list. Change the layout and style by modifying the header, footer, css, and other files in your new skin directory.

It is important to note that all customized skins will be overwritten when Metacat is reinstalled or upgraded. Please remember to back up your skins before reinstalling Metacat.

4 Accessing and Submitting Metadata and Data

The Metacat repository can be accessed and updated using a number of tools, including:

- the Registry, Metacat's optional Web interface
- user-created HTML forms
- Metacat's EarthGrid API
- existing clients, such as KNB's Morpho application, designed to help scientists create, edit, and manage metadata
- user-created desktop clients that take advantage of Metacat's Java API.

In this section, we will look at how to take advantage of these tools to customize Metacat for your user-base.

4.1 A Brief Note about How Information is Stored

Metacat stores XML files as a hierarchy of nodes, where each node is stored as records in database tables. Because many XML data schemas are broken up into multiple DTDs requiring multiple XML files that are related but stored separately in the system, the system uses "packages" to link related but separate documents. Packaged documents contain information that shows how they are related to each other, essentially stating that file A has a relationship to file B, etc. A package file also allows users to link metadata files to the data files they describe. For more information about the structure of data packages and how XML documents and data are stored in Metacat, please see the developer's documentation.

4.2 Using the Registry

Metacat's optional Registry provides a simple Web-based interface for creating, editing, and submitting metadata to the Metacat repository (Figure 4.1). The interface includes help documentation, and can be customized using Metacat's configuration settings. The Registry also includes an administrative interface for managing LDAP user accounts, which is useful if you are using LDAP as your Metacat authentication system. Note that you must be running your own LDAP server if you wish to use the LDAP Web interface. If you do not have your own LDAP server, you can create and manage new accounts on the KNB website (<http://knb.ecoinformatics.org/>). Please note that at this time, the Registry interface has only been tested on Linux systems.

NAME OF SUBMITTER (What's this?)		Hide
'First Name	<input type="text"/>	
'Last Name	<input type="text"/>	
BASIC INFORMATION (What's this?)		Show
PRINCIPAL DATA SET OWNER (What's this?)		Show
ASSOCIATED PARTIES (What's this?)		Show
DATA SET ABSTRACT (What's this?)		Hide
'Data Set Abstract (max. 350 words)	<input type="text"/>	
KEYWORD INFORMATION (What's this?)		Hide
For samples, see NASA Global Change Master Directory (GCMD) .		
Keyword	<input type="text"/>	
Keyword Type	None <input type="button" value="v"/>	
Keyword Thesaurus	None <input type="button" value="v"/>	
	<input type="button" value="Add Keyword"/>	
TEMPORAL COVERAGE OF DATA (What's this?)		Hide
Start Date		Stop Date
'Year (yyyy)	<input type="text"/>	Year (yyyy)
Month	MM <input type="button" value="v"/>	Month
Day	DD <input type="button" value="v"/>	Day
Note: Leave "Stop Date" blank if your data set is open-ended.		
SPATIAL COVERAGE OF DATA (What's this?)		Show
TAXONOMIC COVERAGE OF DATA (What's this?)		Show
DATA COLLECTION METHODS (What's this?)		Show
DATA SET CONTACT (What's this?)		Show
DISTRIBUTION INFORMATION (What's this?)		Show
UPLOAD DATA (What's this?)		Hide
Upload Data File:	<input type="text"/>	<input type="button" value="Browse..."/>
Attached Files: <i>(None currently attached)</i>		
<input type="button" value="Submit Data Set Description"/>		

Figure 4.1: An example installation of the Register's web interface. Customize the displayed and required modules with the Skins Configuration settings.

You can customize which modules (e.g., "Name of Submitter" or "Temporal Coverage of Data") are displayed and which are required using the [Skins Configuration](#) settings.

4.2.1 Installing the Registry

In order to install and run the Registry, you must have Metacat installed and Tomcat must be running behind an Apache Web server (see [Section 2.2.2.3](#) for information about installing and configuring Apache to run with Tomcat).

To install and run the Registry:

- 1) Build the Metacat Perl client library:

```
cd $METACAT/src/perl/Metacat
perl Makefile.PL
sudo make
sudo make install
```

2) Install the required system libraries using Ubuntu/Debian (instructions Red Hat included beneath Ubuntu/Debian instructions)

a) Install the libraries

```
sudo apt-get install ant libappconfig-perl libxml-libxml-perl libxml-libxslt-perl libtemplate-perl
libcgi-session-perl libdigest-sha1-perl libnet-ldap-perl libterm-readkey-perl libxml-dom-perl libsoap-
lite-perl -y
```

b) Install two more package using cpan

```
sudo cpan -i Config::Properties
sudo cpan -i Scalar::Util
```

Instructions for Red Hat (Step 3)

a) Install the libraries

```
sudo yum install gcc libxml2-devel libxslt-devel ant -y
```

b) Install CPAN, which allows us to install the Perl dependencies for the registry and account management parts of Metacat. If asked to manually configure cpan, type 'no' and CPAN will be setup with its default values.

```
sudo yum install perl-CPAN
sudo cpan
```

c) You should now see a prompt which looks like:

```
cpan>
```

d) The rest of the commands assume you're inside of CPAN. Let's get the most recent version of the CPAN software. Just press return after any prompts you receive during this process.

```
install Bundle::CPAN
reload cpan
```

- e) Install the required modules. Here we're installing an old LibXSLT, as the current one requires a newer libxslt than is available on Redhat 4 & 5. Again, just answer 'yes' to any questions.

```
install AutoLoader
install CGI
install CGI::Session
install LWP::UserAgent
install Net::LDAP
install Template
install URI
install MSERGEANT/XML-LibXSLT-1.58.tar.gz
```

- 3) Double-check that Metacat's temporary folder, `application.tempDir`, is writable by the apache user, usually `www-data` or `apache`.
- 4) Make sure that the following scripts (found in `<tomcat-home>/webapps/knb/cgi-bin`) are executable: `register-dataset.cgi` and `ldapweb.cgi`.

```
sudo chmod +x <tomcat-home>/webapps/knb/cgi-bin/*.cgi
```

- 5) Restart Apache.

```
sudo /etc/init.d/apache2 restart
```

- 6) Visit the resulting URL: `http://<your_context_url>/cgi-bin/register-dataset.cgi?cfg=default`
Where `<your_context_url>` is the URL of the server hosting the Metacat followed by the name of the WAR file (i.e., the application context) that you installed. For instance, the context URL for the KNB Metacat is: <http://knb.ecoinformatics.org/knb>.

If everything worked correctly, the registry home page will open (Figure 4.2).

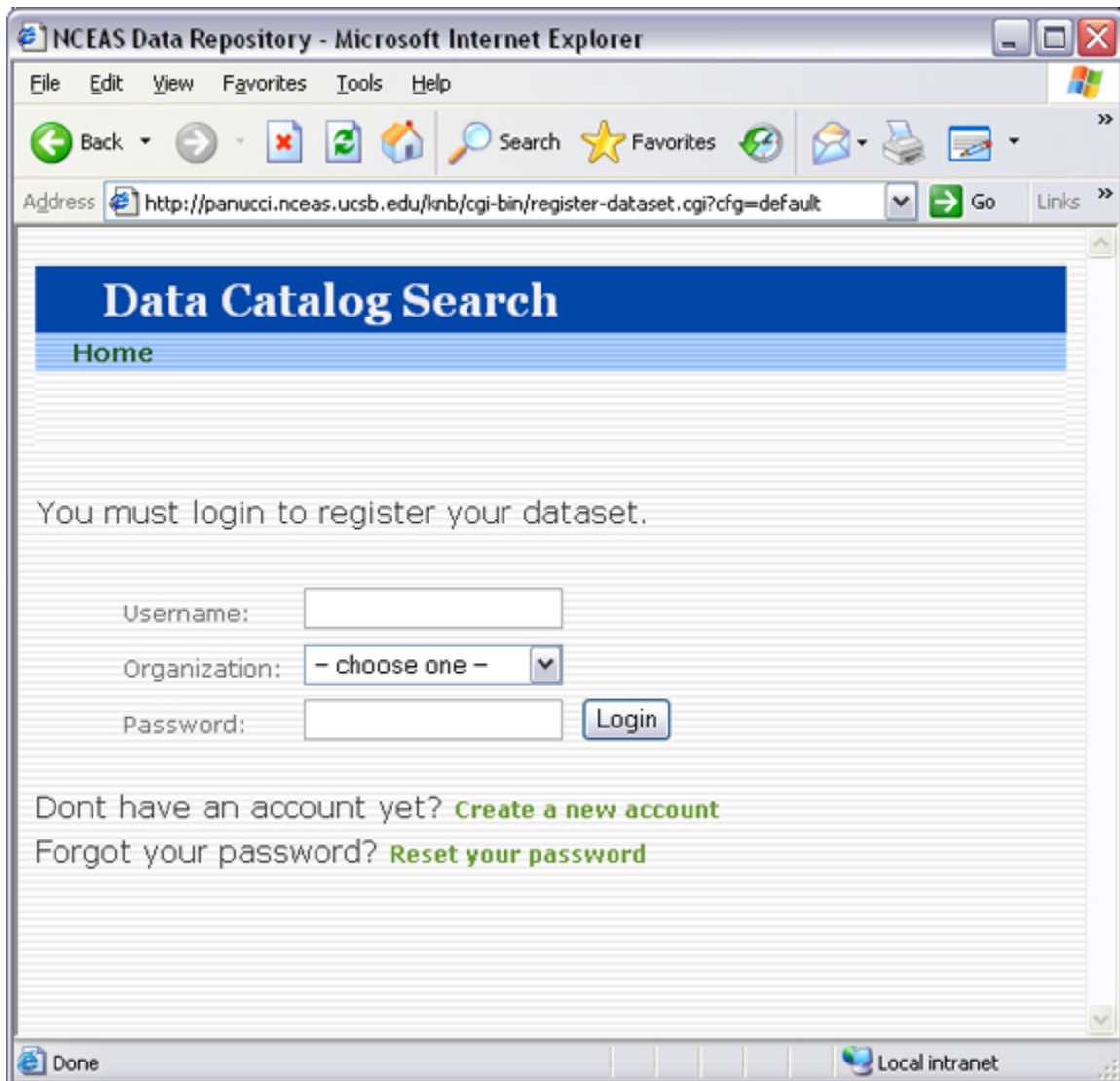


Figure 4.2: An example of the Registry home page (with the default skin).

4.2.2 Customizing the Registry

Before using the registry, you may wish to customize the interface using the [Skins Configuration settings](#). If you are using the default skin, you must disable the 'show site list' setting before you can submit the form without errors. You may also wish to remove (or modify) the list of NCEAS-specific projects that appear in the default registry. To remove these form fields, open Metacat's administrative interface (<http://<your.context.url>/knb/admin>) and select the Skins Specific Properties Configuration option. On the skins configuration page, uncheck the boxes beside any form elements that you do not wish to appear in the registry (Figure 4.3).

Once you have saved your changes, you must restart Tomcat for them to come into effect. To restart Tomcat, log in as the user that runs your Tomcat server (often "tomcat") and type: `/etc/init.d/tomcat5.5 restart` or an equivalent command appropriate to your operating system.

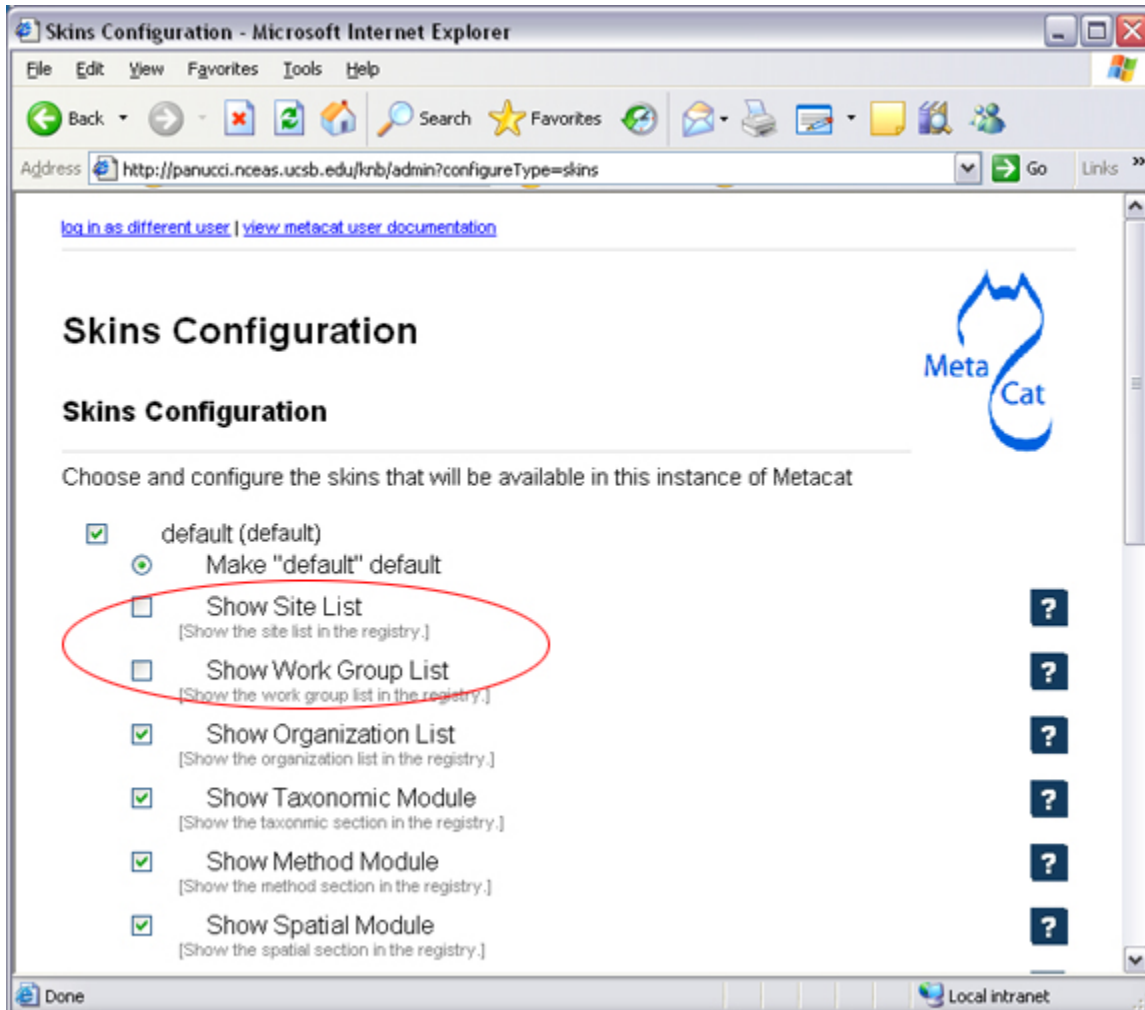


Figure 4.3: Uncheck the box beside any setting to remove it from the Registry form. In the example, the "Show Site List" and "Show Work Group" form fields, corresponding to the "Station Name" and "NCEAS Project" drop-down lists in the registry form, have been removed.

4.3 Using HTML Forms (the HTTP Interface)

Metacat's HTTP interface supports Get and Post requests and a variety of actions (Table 4.1) that facilitate information retrieval and storage. HTTP requests can be sent from any client application that communicates using the Web's HTTP protocol.

- Supported Actions (API)
- Logging in
- Inserting, Updating, and Deleting XML and Data Documents

- [Searching Metacat](#)
- [Paged Query Return](#)
- [Reading Data and Metadata](#)

4.3.1 Supported Actions

Metacat supports get and post requests as well as actions for writing, querying, and reading stored XML. In addition, the HTTP interface includes functions for validating and transforming XML documents (Table 4.1).

Note that if Replication is enabled, Metacat recognizes several additional actions, included in Table 4.2. For more information about replication, please see the [Replication](#) section.

Actions Supported by the Metacat Servlet

Action	Description and Parameters
<code>delete</code>	Delete the specified document from the database. For an example, please see Inserting, Updating, and Deleting XML and Data Documents . <code>docid</code> - the docid of the document to delete
<code>export</code>	Export a data package in a zip file. <code>docid</code> - the docid of the package you wish to export
<code>getaccesscontrol</code>	Get the access control list (ACL) for the specified document. <code>docid</code> - the docid of the document for which to get the ACL
<code>getalldocids</code>	Retrieve a list of all docids registered with the system. <code>scope</code> - a string used to match a range of docids in a SQL LIKE statement.
<code>getdataguide</code> DEPRECATED Use <code>getdttdschema</code> instead.	Read a data guide for the specified document type. <code>doctype</code> - the doctype for which to get the data guide
<code>getdoctypes</code>	Get all doctypes currently available in the Metacat Catalog System. No parameters
<code>getdttdschema</code>	Read the DTD or XMLSchema file for the specified doctype. <code>doctype</code> - the doctype for which DTD or XMLSchema files to read.

getlastdocid	Get the latest docid with revision number used by user. username - the user's log-in name
getlog	Print a report from the Metacat event log. ipaddress - the internet protocol address for the event principal - the principal for the event (a username, etc) docid - the identifier of the document to which the event applies event - the string code for the event start - beginning of date-range for query end - end of date-range for query
getloggedinuserinfo	Get user info for the currently logged in user. No parameters.
getprincipals	Get all users and groups in the current authentication schema. No parameters.
getrevisionanddoctype	Return the revision and doctype of a document. The output is String that looks like "rev;doctype" docid - the docid of the document.
getversion - get Metacat version.	Return the current version of Metacat as XML. No parameters.
insert	Insert an XML document into the database. For an example, please see Inserting, Updating, and Deleting XML and Data Documents docid - the user-defined docid to assign to the new XML document. doctext - the text of the XML document to insert
insertmultipart	Insert an XML document using multipart encoding into the database. docid - the user-defined docid to assign to the new XML document. doctext - the text of the XML document to insert
isregistered	Check if an individual document exists in either the <code>xml_documents</code> or <code>xml_revisions</code> tables. For more information about Metacat's database schema, please see the developer documentation. docid - the docid of the document.

login	<p>Log the user in. You must log in using this action before you can perform many of the actions. For an example of the login action, see Logging In.</p> <p>username - the user's login name</p> <p>password - the user's password</p>
logout	<p>Log the current user out and destroy the associated session. No parameters</p>
query	<p>Perform a free text query. For an example, please see Searching Metacat</p> <p>returndoctype - the doctype to use for your Package View. For more information about packages, see http://knb.ecoinformatics.org/software/metacat/packages.html</p> <p>qformat - the format of the returned result set. Possible values are html or xml or the name of your servlet's Metacat skin.</p> <p>querytitle - OPTIONAL - the title of the query</p> <p>doctype - OPTIONAL - if doctype is specified, the search is limited only to the specified doctype(s). (e.g., eml://ecoinformatics.org/eml-2.0.1 and/or eml://ecoinformatics.org/eml-2.0.0) If no doctype element is specified, all document types are returned</p> <p>returnfield - a custom field to be returned by any hit document.</p> <p>operator - the Boolean operator to apply to the query. Possible values are: union or intersect</p> <p>searchmode - the type of search to be performed. Possible values are: contains, starts-with, ends-with, equals, isnot-equal, greater-than, less-than, greater-than-equals, less-than-equals.</p> <p>anyfield - a free-text search variable. The value placed in this parameter will be searched for in any document in any node.</p> <p>pagesize - the number of search results to display on each search results page (e.g., 10). Used with pagestart. See section 4.3.4 for an example.</p> <p>pagestart - the displayed search results page (e.g, 1). Used with pagesize. See section 4.3.4 for an example.</p>
read	<p>Get a document from the database and return it in the specified format. See Searching Metacat for an example.</p> <p>docid - the docid of the document to return</p> <p>qformat - the format to return the document in. Possible values are: html, xml, or, if your Metacat uses a skin, the name of the skin.</p>

readinlinedata	<p>Read inline data only.</p> <p>inlinedataid - the id of the inline data to read</p>
setaccess	<p>Change access permissions for a user on a specified document.</p> <p>docid - the docid of the document to be modified.</p> <p>principal - the user or group whose permissions will be modified</p> <p>permission - the permission to set (read, write, all)</p> <p>permType - the type of permission to set (allow, deny)</p> <p>permOrder - the order in which to apply the permission (allowFirst, denyFirst)</p>
spatial_query	<p>Perform a spatial query. These queries may include any of the queries supported by the WFS / WMS standards. For more information, see Spatial Queries.</p> <p>xmax - max x spatial coordinate ymax - max y spatial coordinate xmin - min x spatial coordinate ymin - min y spatial coordinate</p>
squery	<p>Perform a structured query. For an example, please see Searching Metacat.</p> <p>query - the text of the pathquery document sent to the server</p>
update	<p>Overwrite an XML document with a new one and give the new one the same docid but with the next revision number. For an example, please see Inserting, Updating, and Deleting XML and Data Documents.</p> <p>docid - the docid of the document to update doctext - the text with which to update the XML document</p>
upload	<p>Upload (insert or update) a data file into Metacat. Data files are stored on Metacat and may be in any format (binary or text), but they are all treated as if they were binary.</p> <p>docid - the docid of the data file to upload datafile - the data file to upload</p>
validate	<p>Validate a specified document against its DTD.</p> <p>docid - the docid of the document to validate valtext - the DTD by which to validate this document</p>

Table 4.1 Supported actions and associated parameters.

Metacat Replication Parameters

Action	Description and Parameters
forcereplicate	Force the local server to get the specified document from the remote host. server - The server to which this document is being sent docid - The docid of the document to send dbaction - The action to perform on the document: <code>insert</code> or <code>update</code> (the default)
getall	Force the local server to check all known servers for updated documents. No parameters.
getcatalog	Send the contents of the <code>xml_catalog</code> table encoded in XML. No parameters.
getlock	Request a lock on the specified document. docid - the docid of the document updaterev - the revision number of docid
gettime	Return the local time on this server. No parameters.
servercontrol	Perform the specified replication control on the Replication daemon. add - add a new server to the replication list delete - remove a server from the replication list list - list all of the servers currently in the server list replicate - a Boolean flag (1 or 0) which determines if this server should copy files from the newly added server. server - the server to add/delete
read	Sends docid to the remote host. docid - the docid of the document to read
start	Start the Replication daemon with a time interval of deltaT. rate - The rate (in seconds) at which you want the replication daemon to check for updated documents. The value cannot be less than 30. The default is 1000
stop	Stop the Replication daemon. No parameters.
update	Send a list of all documents on the local server along with their revision numbers. No parameters.

Table 4.2 Supported actions when Replication is enabled.

4.3.2 Logging In

To log in to Metacat, use the `login` action.

The following is an example of a Web form (Figure 4.4) that logs a user into Metacat. Example HTML code is included below the screenshot (Figure 4.5).

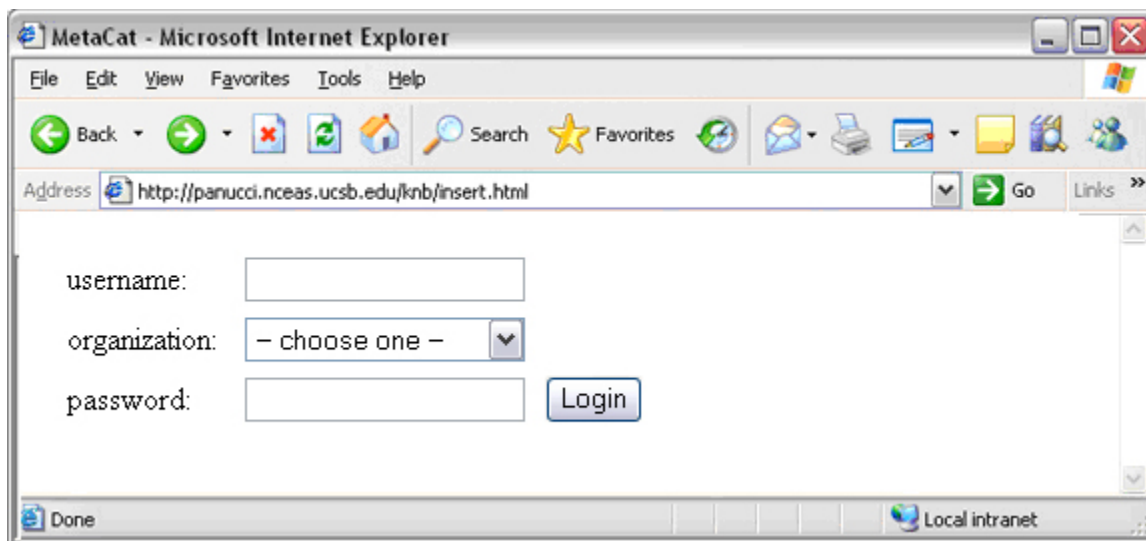


Figure 4.4: Logging in to Metacat using an HTML form.

```
<body>
  <form name="loginform"
method="post"action="http://yourserver.com/yourcontext/servlet/metacat"
target="_top" onsubmit="return submitform(this);" id="loginform">
  <input type="hidden" name="action" value="login"> <input type=
"hidden" name="username" value=""> <input type="hidden" name=
"qformat" value="xml"> <input type="hidden" name=
"enableediting" value="false">

  <table>
    <tr valign="middle">
      <td align="left" valign="middle" class="text_plain">
        username:</td>

      <td width="173" align="left" class="text_plain" style=
"padding-top: 2px; padding-bottom: 2px;"><input name="uid"
type="text" style="width: 140px;" value=""></td>
    </tr>

    <tr valign="middle">
      <td height="28" align="left" valign="middle" class=
"text_plain">organization:</td>

      <td align="left" class="text_plain" style=
```

```

"padding-top: 2px; padding-bottom: 2px;"><select name=
"organization" style="width:140px;">
  <option value="" selected>&#8212; choose one
&#8212;</option>
  <option value="NCEAS">NCEAS</option>
  <option value="LTER">LTER</option>
  <option value="UCNRS">UCNRS</option>
  <option value="PISCO">PISCO</option>
  <option value="OBFS">OBFS</option>
  <option value="OSUBS">OSUBS</option>
  <option value="SAEON">SAEON</option>
  <option value="SANParks">SANParks</option>
  <option value="SDSC">SDSC</option>
  <option value="KU">KU</option>
  <option value="unaffiliated">unaffiliated</option>
</select></td>
</tr>

<tr valign="middle">
  <td width="85" align="left" valign="middle" class=
  "text_plain">password:</td>

  <td colspan="2" align="left" class="text_plain" style=
  "padding-top: 2px; padding-bottom: 2px;">
    <table width="100%" border="0" cellpadding="0"
    cellspacing="0">
      <tr>
        <td width="150" align="left"><input name="password"
        type="password" maxlength="50" style="width:140px;"
        value=""></td>

        <td align="center" class="buttonBG_login">
          <input type="submit" name="loginAction" value="Login"
          class="button_login"></td>

        <td align="left">&nbsp;</td>
      </tr>
    </table>
  </td>
</tr>
</table>
</form>
</body>
</html>

```

Figure 4.5: HTML code used to generate the log-in form in Figure 4.4

4.3.3 Inserting, Updating, and Deleting XML and Data Documents

Adding, editing, and deleting XML documents in Metacat can be accomplished using the *insert*, *update*, and *delete* actions, respectively. Before you can insert, delete, or update documents, you must log in to Metacat using the login action. See [Logging in](#) for an example.

- insert:** Insert a new XML or data document into Metacat. You must specify a document ID.
- update:** Update an existing Metacat document. The original document is archived, then overwritten.
- delete:** Archive a document and move the pointer in xml_documents to xml_revisions, effectively "deleting" the document from public view, but preserving the revision for the revision history.

The following is an example of a Web form (Figure 4.6) that can perform all three tasks. Example HTML code is included below the screenshot (Figure 4.7).

MetaCat XML Loader

Upload, Change, or Delete an XML document using this form.

1. Choose an action: Insert Update Delete

2. Provide a Document ID (optional for Insert) Public Document

3. Provide XML text (not needed for Delete)

4. Provide DTD text for upload (optional; not needed for Delete)

Figure 4.6: An example of a Web form used to insert, delete, or update XML documents in Metacat.

```

<html>
  <head>
    <title>MetaCat</title>
  </head>
  <body class="emlbody">
    <b>MetaCat XML Loader</b>
    <p>
      Upload, Change, or Delete an XML document using this form.
    </p>
    <form action="http://yourserver.com/yourcontext/servlet/metacat"
method="POST">
      <strong>1. Choose an action: </strong>
      <input type="radio" name="action" value="insert" checked> Insert
      <input type="radio" name="action" value="update"> Update
      <input type="radio" name="action" value="delete"> Delete
      <input type="submit" value="Process Action">
      <br />
      <strong>2. Provide a Document ID </strong>
      <input type="text" name="docid"> (optional for Insert)
      <input type="checkbox" name="public" value="yes"
checked><strong>Public Document</strong>
      <br />
      <strong>3. Provide XML text </strong> (not needed for
Delete)<br/>
      <textarea name="doctext" cols="65" rows="15"></textarea><br/>
      <strong>4. Provide DTD text for upload </strong> (optional; not
needed for Delete)
      <textarea name="dtdtext" cols="65" rows="15"></textarea>
    </form>
  </body>
</html>

```

Figure 4.7: HTML code used to generate the form in Figure 4.6.

4.3.4 Searching Metacat

To search Metacat use the `query` or `squery` actions.

`query`: Perform a free text query. Specify the `returndoctype`, `qformat`, `returnfield`, `operator`, `searchmode`, `anyfield`, and (optionally) a `querytitle` and `doctype`.

`squery`: Perform a structured query by submitting an XML `pathquery` document to the Metacat server.

When Metacat receives a `query` via HTTP (Figure 4.8), the server creates a "pathquery" document, which is an XML document populated with the specified search criteria (Figure 4.10). The pathquery document is then translated into SQL statements that are executed against the data base. Results are translated into an XML "resultset" document, which can be returned as XML or transformed into HTML and returned (specify which you would prefer with the `returnfield` parameter). You can also opt to submit a pathquery document directly, using an `squery` action.

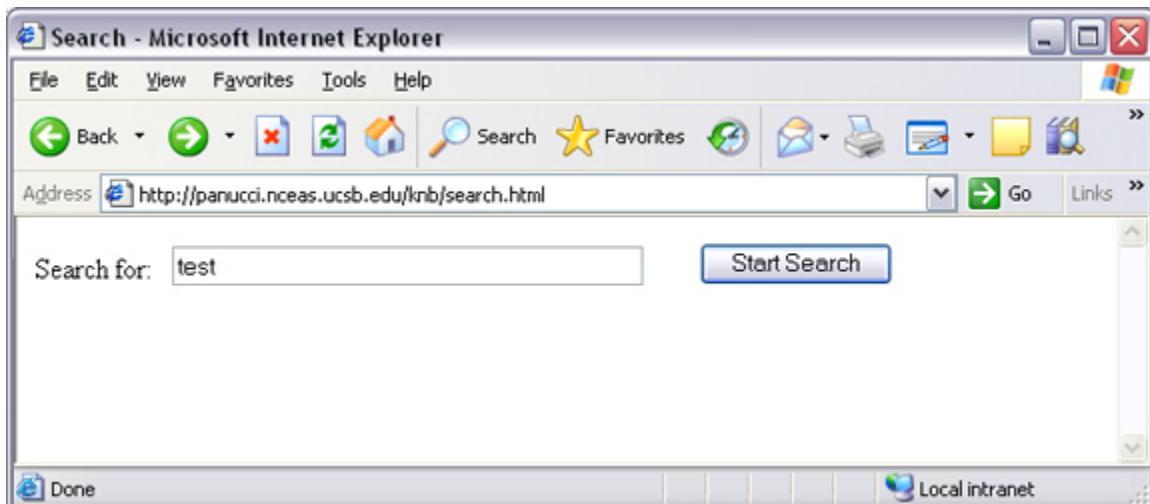


Figure 4.8: Example of a basic search form using a query action. The HTML code used to create the form is displayed in Figure 4.9.

```

<html>
<head>
<title>Search</title>
</head>
<body>
<form method="POST" action="http://panucci.nceas.ucsb.edu/knb/servlet/metacat">

Search for:

  <input name="action" value="query" type="hidden">

  <input name="operator" value="INTERSECT" type="hidden">

  <input name="anyfield" type="text" value=" " size="40">

<input name="qformat" value="html" type="hidden">

<input name="returnfield" value="creator/individualName/surName" type="hidden">

<input name="returnfield" value="creator/individualName/givenName"
type="hidden">

  <input name="returnfield" value="creator/organizationName" type="hidden">

  <input name="returnfield" value="dataset/title" type="hidden">

  <input name="returnfield" value="keyword" type="hidden">

  <input name="returndoctype" value="eml://ecoinformatics.org/eml-2.0.1"
type="hidden">

  <input value="Start Search" type="submit">

</form>
</body>
</html>

```

Figure 4.9: HTML form code that generates the search form displayed in Figure 4.8.

Metacat's pathquery document can query specific fields of any XML document. The pathquery can also be used to specify which fields from each hit are returned and displayed in the search result set (Figure 4.10)

```
<pathquery version="1.0">
  <meta_file_id>unspecified</meta_file_id>
  <querytitle>unspecified</querytitle>
  <returnfield>dataset/title</returnfield>
  <returnfield>keyword</returnfield>
  <returnfield>originator/individualName/surName</returnfield>
  <returndoctype>eml://ecoinformatics.org/eml-2.0.1</returndoctype>
  <returndoctype>eml://ecoinformatics.org/eml-2.0.0</returndoctype>
  <querygroup operator="UNION">
    <queryterm casesensitive="false" searchmode="contains">
      <value>Plant</value>
      <pathexpr>dataset/title</pathexpr>
    </queryterm>
    <queryterm casesensitive="false" searchmode="contains">
      <value>plant</value>
      <pathexpr>keyword</pathexpr>
    </queryterm>
  </querygroup>
</pathquery>
```

Figure 4.10: An example of a pathquery document

Each `<returnfield>` parameter specifies a field that the data base will return (in addition to the fields Metacat returns by default) for each search result.

The `<returndoctype>` field limits the type of returned documents

(eg, `eml://ecoinformatics.org/eml-2.0.1` and/or `eml://ecoinformatics.org/eml-2.0.0`).

If no `returndoctype` element is specified, all document types are returned.

A `<querygroup>` creates an AND or an OR statement that applies to the nested `<queryterm>` tags. The `querygroup` operator can be UNION or INTERSECT. A `<queryterm>` defines the actual field (contained in `<pathexpr>` tags) against which the query (contained in the `<value>` tags) is being performed.

The `<pathexpr>` can also contain a document type keyword contained in `<returndoc>` tags. The specified document type applies only to documents that are packaged together (e.g., a data set and its corresponding metadata file). If Metacat identifies the search term in a packaged document, the servlet will check to see if that document's type matches the specified one. If not, Metacat will check if one of the other documents in the package matches. If so, Metacat will return the matching document. For more information about packages, please see the developer documentation.

After Metacat has processed a Pathquery document, it returns a resultset document (Figure 4.11)

```

<resultset>
  <query>
    <pathquery version="1.0">
      <meta_file_id>unspecified</meta_file_id>
      <querytitle>unspecified</querytitle>
      <returnfield>dataset/title</returnfield>
      <returnfield>keyword</returnfield>
    </pathquery>
  </query>
  <returnfield>originator/individualName/surName</returnfield>
  <returndoctype>eml://ecoinformatics.org/eml-
2.0.1</returndoctype>
  <returndoctype>eml://ecoinformatics.org/eml-
2.0.0</returndoctype>
  <querygroup operator="UNION">
    <queryterm casesensitive="false"
searchmode="contains">
      <value>Datos</value>
      <pathexpr>dataset/title</pathexpr>
    </queryterm>
    <queryterm casesensitive="false"
searchmode="contains">
      <value>plant</value>
      <pathexpr>keyword</pathexpr>
    </queryterm>
  </querygroup>
</pathquery>
</query>

  <document>
    <docid>nceas.44.1</docid>
    <docname>resource</docname>
    <doctype>eml://ecoinformatics.org/eml-2.0.1</doctype>
    <createdate>2001-01-12 16:12:06.0</createdate>
    <updatedate>2001-01-12 16:12:06.0</updatedate>
    <param name="dataset/title">Datos Meteorologicos</param>
    <param name="keyword">intertidal</param>
    <param name="originator/individualName/surName">Smith</param>
  </document>

  <document>
    <docid>nceas.42.1</docid>
    <docname>resource</docname>
    <doctype>eml://ecoinformatics.org/eml-2.0.1</doctype>
    <createdate>2001-01-12 16:11:31.0</createdate>
    <updatedate>2001-01-12 16:11:31.0</updatedate>
    <param name="dataset/title">Ocean Surface Temperature</param>
    <param name="keyword">Plant</param>
    <param name="originator/individualName/surName">Henry</param>
  </document>
  .....
</resultset>

```

Figure 4.11: An example of a resultset document.

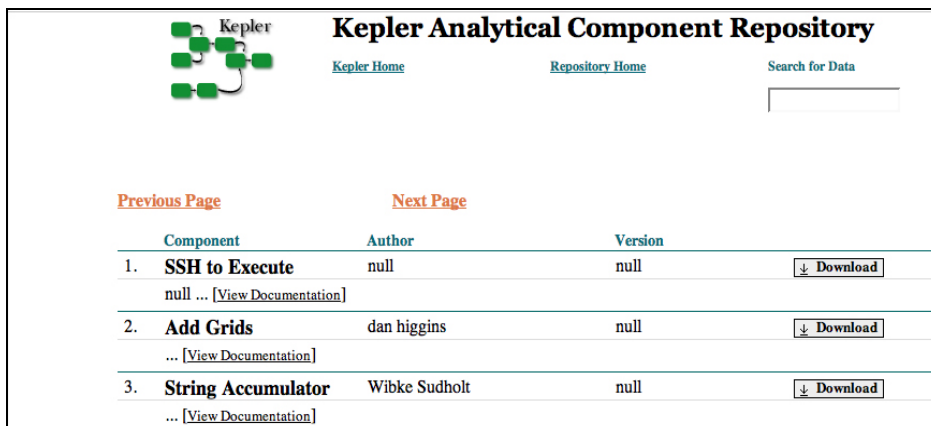
When Metacat returns a resultset document, the servlet always includes the pathquery used to create it. The pathquery XML is contained in the <query> tag, the first element in the resultset.

Each XML document returned by the query is represented by a <document> tag. By default, Metacat will return the docid, docname, doctype, doctitle, createdate and updatedate for each search result. If the user specified additional return fields in the pathquery using <returnfield> tags (e.g., dataset/title to return the document title), the additional fields are returned in <param> tags.

Metacat can return the XML resultset to your client as either XML or HTML.

4.3.5 Paged Query Return

Dividing large search result sets over a number of pages speeds load-time and makes the result sets more readable to users (Figure 4.12). To break your search results into pages, use the query action's optional pagestart and pagesize parameters. The pagesize parameter indicates how many results should be returned for a given page. The pagestart parameter indicates which page you are currently viewing.



Component	Author	Version	
1. SSH to Execute	null	null	Download
null ... View Documentation			
2. Add Grids	dan higgins	null	Download
... View Documentation			
3. String Accumulator	Wibke Sudholt	null	Download
... View Documentation			

Figure 4.12: An example of paged search results.

When a paged query is performed, the query's resultset contains four extra fields: pagestart, pagesize, nextpage, and previouspage (Figure 4.13). The nextpage and previouspage fields help Metacat generate navigational links in the rendered resultset using XSLT to transform the XML to HTML.

```

<resultset>
  <pagestart>1</pagestart>
  <pagesize>10</pagesize>
  <nextpage>2</nextpage>
  <previouspage>0</previouspage>
  <query> ...</query>
  <document>...</document>
  <document>...</document>
</resultset>

```

Figure 4.13: An example of an XML resultset that include support for page breaks. The pagestart parameter will always indicate the page you are currently viewing.

The HTML search results displayed in Figure 4.12 were rendered using Kepler's XSLT, which can be found in `lib/style/skins/kepler`. Kepler's XSLT uses the four extra resultset fields to render the "Next" and "Previous" links (Figure 4.14).

```
<a href="metacat?action=query&operator=INTERSECT&enableediting=false&anyfield=actor&qformat=kepler&pagestart=0&pagesize=10">Previous Page</a>

<a
href="metacat?action=query&operator=INTERSECT&enableediting=false&anyfield=actor&qformat=kepler&pagestart=2&pagesize=10">Next Page</a>
```

Figure 4.14: The HTML Next and Previous links generated by Metacat using Kepler's XSLT.

In the example in Figure 4.14, the current page is 1, and the previous page (page 0) and next page (page 2) pages are indicated by the values of the `pagestart` parameters.

4.3.6 Reading Data and Metadata

To read data or metadata from Metacat, use the `read` action. The `read` action takes two parameters: `docid`, which specifies the document ID of the document to return, and `qformat`, which specifies the return format for the document (`html` or `xml` or the name of a configured style-set, e.g., `default`). If `qformat` is set to `xml`, Metacat will return the XML document untransformed. If the return format is set to `html`, Metacat will transform the XML document into HTML using the default XSLT style sheet (specified in the Metacat configurations). If the name of a style-set is specified, Metacat will use the XSLT styles specified in the set to transform the XML (Figure 4.15).

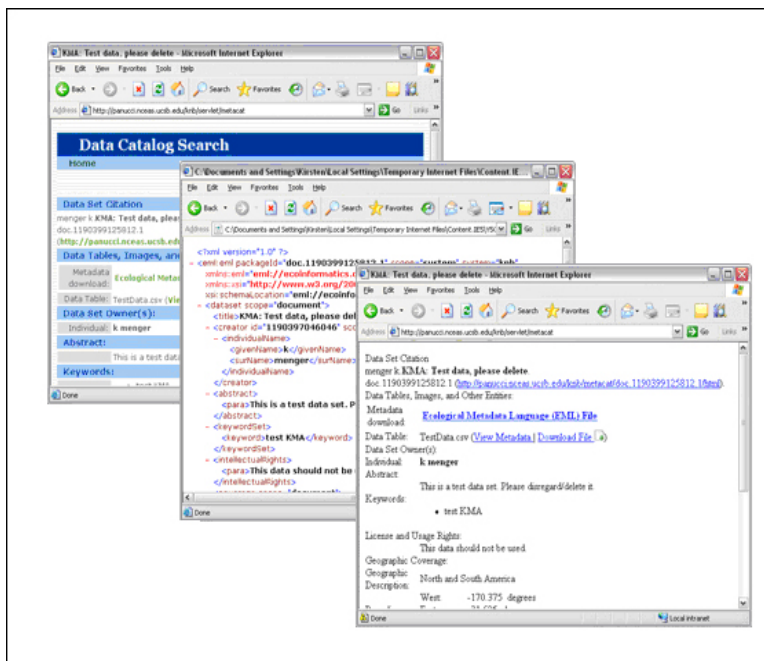


Figure 4.15: The same document displayed using different `qformat` parameters (from left to right: the `default` style-set, XML, and HTML)

Note that the `read` action can be used to read both data files and metadata files. To read a data file, you could use the following request:

```
http://yourserver.com/yourcontext/metacat?action=read&docid=nceas.55&qformat=default
```

Where `nceas.55` is the `docid` of the data file stored in the Metacat and `default` is the name of the style (you could also use `"html"` or `"xml"` or the name of a customized skin).

```
<html>
<head>
<title>Read Document</title>
</head>
<body>
<form method="POST" action="http://your.server/your.context/servlet/metacat">
<input name="action" value="read" type="hidden">
<input name="docid" type="text" value="" size="40">
<input name="qformat" value="default" type="hidden">
  <input value="Read" type="submit">
</form>
</body>
</html>
```

4.4 Using the EarthGrid API

The EarthGrid provides access to disparate data on different networks (e.g., KNB, GBIF, GEON) and storage systems (e.g., Metacat and SRB), allowing scientists access to a wide variety of data and analytic resources (e.g., data, metadata, analytic workflows and processors) networked at different sites and at different organizations via the internet.

Because Metacat supports the EarthGrid API (Table 4.3), it can query the distributed EarthGrid, retrieve metadata and data results, and write new and updated metadata back to the grid nodes.

For more information about each EarthGrid service and its WSDL file, navigate to the "services" page on your Metacat server (e.g., <http://knb.ecoinformatics.org/knb/services>). Note that the AdminService and Version service that appear on this page are not part of EarthGrid.

The EarthGrid API

Service	Description
AuthenticationQueryService	Search for and retrieve protected metadata and data from the EarthGrid as an authenticated user. Methods: <code>query</code> , <code>get</code>
AuthenticationService	Log in and out of the EarthGrid Methods: <code>login</code> , <code>logout</code>

IdentifierService	List, lookup, validate, and add Life Science Identifiers (LSIDs) to the EarthGrid Methods: <code>isRegistered</code> , <code>addLSID</code> , <code>getNextRevision</code> , <code>getNextObject</code> , <code>getAllIds</code>
PutService	Write metadata to the EarthGrid Methods: <code>put</code>
QueryService	Search for and retrieve metadata from the EarthGrid Methods: <code>query</code> , <code>get</code>
RegistryService	Add, update, remove, and search for registered EarthGrid services. Note: The WSDL for this service is found under http://ecogrid.ecoinformatics.org/registry/services Methods: <code>add</code> , <code>update</code> , <code>remove</code> , <code>list</code> , <code>query</code>

Table 4.3: The EarthGrid API

4.5 Using Morpho

Morpho is a desktop tool created for ecologists to facilitate the creation, storage, and retrieval of metadata. Morpho interfaces with any Metacat server, allowing users to upload, download, store, query and view relevant metadata and data using the network. Users can authorize the public or only selected colleagues to view their data files.

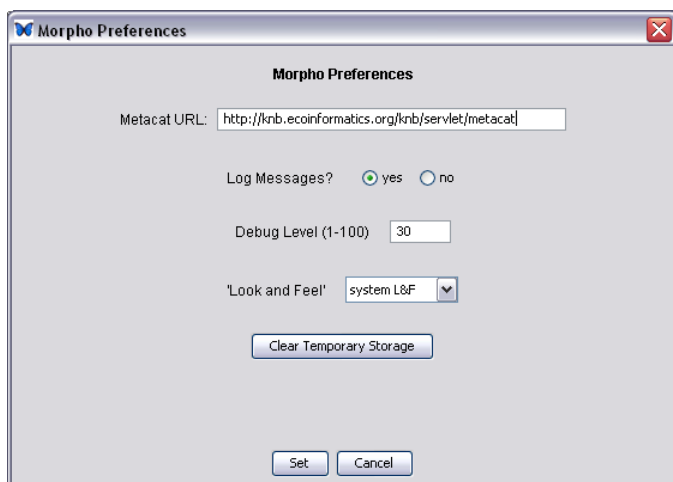


Figure 4.16: Set the Metacat URL in the Morpho preferences to point to your Metacat.

Morpho is part of the Knowledge Network for Biocomplexity (KNB), a national network intended to facilitate ecological and environmental research on biocomplexity. To use Morpho with your Metacat, set the Metacat URL in the Morpho Preferences to point to your Metacat server (Figure 4.16)

For more information about Morpho, please see: <http://knb.ecoinformatics.org/morphportal.jsp>

4.6 Creating Your Own Client

Metacat's client API is available in Java and Perl (the Java interface is described in this section and further detailed in the appendix). Some of the API is also available in Python and Ruby. The API allows client applications to easily authenticate users and perform basic Metacat operations such as reading metadata and data files; inserting, updating, and deleting files; and searching for packages based on metadata matches.

The Client API is defined by the interface `edu.ucsb.nceas.metacat.client.Metacat`, and all operations are fully defined in the [javadoc](#) documentation. To use the client API, include the `metacat-client.jar`, `utilities.jar`, `commons-io-2.0.jar`, and `httpclient.jar` in your classpath. After including these classes, you can begin using the API methods (Table 4.4).

Figure 4.17 displays a typical session for reading a document from Metacat.

```
String metacatUrl = "http://foo.com/context/metacat";
String username = "uid=jones,o=NCEAS,dc=ecoinformatics,dc=org";
String password = "neverHarcodEAPasswOrdInCode";
try {
    Metacat m =
MetacatFactory.createMetacatConnection(metacatUrl);
    m.login(username, password);
    Reader r = m.read("testdocument.1.1");
    // Do whatever you want with Reader r
} catch (MetacatAuthException mae) {
    handleError("Authorization failed:\n" + mae.getMessage());
} catch (MetacatInaccessibleException mie) {
    handleError("Metacat Inaccessible:\n" + mie.getMessage());
} catch (Exception e) {
    handleError("General exception:\n" + e.getMessage());
}
```

Figure 4.17: A typical session for reading a document from Metacat.

Operations provided by Client API (metacat.java class)

Method	Parameters and Throws	Description
delete	<code>public String delete(String docid) throws InsufficientKarmaException, MetacatException, MetacatInaccessibleException;</code>	Delete an XML document in the repository.
getAllDocids	<code>public Vector getAllDocids(String scope) throws MetacatException;</code>	Return a list of all docids that match a given scope. If scope is null, return all docids registered in the system.

getLastDocid	public String getLastDocid(String scope) throws MetacatException;	Return the highest document ID for a given scope. Used by clients to determine the next free identifier in a sequence for a given scope.
getloggeduserinfo	public String getloggeduserinfo() throws MetacatInaccessibleException;	Return the logged in user for this session.
getNewestDocRevision	public int getNewestDocRevision (String docId) throws MetacatException;	Return the latest revision of specified the document from Metacat
getSessionId	public String getSessionId ();	Return the session identifier for this session.
insert	public String insert (String docid, Reader xmlDocument, Reader schema) throws InsufficientKarmaException, MetacatException, IOException, MetacatInaccessibleException;	Insert an XML document into the repository.
isRegistered	public boolean isRegistered(String docid) throws MetacatException;	Return true if given docid is registered; false if not.
login	public String login (String username, String password) throws MetacatAuthException, MetacatInaccessibleException;	Log in to a Metacat server.
logout	public String logout () throws MetacatInaccessibleException, MetacatException;	Log out of a Metacat server.
query	public Reader query (Reader xmlQuery) throws MetacatInaccessibleException, IOException;	Query the Metacat repository and return the result set as a Reader.
query	public Reader query (Reader xmlQuery, String qformat) throws MetacatInaccessibleException, IOException;	Query the Metacat repository with the given metacat-compatible query format and return the result set as a Reader.
read	public Reader read (String docid) throws InsufficientKarmaException, MetacatInaccessibleException, DocumentNotFoundException, MetacatException;	Read an XML document from the Metacat server.

readInlineData	public Reader readInlineData (String inlinedataid) throws InsufficientKarmaException, MetacatInaccessibleException, MetacatException;	Read inline data from the Metacat server session.
setAccess	public String setAccess (String _docid, String _principal, String _permission, String _permType, String _permOrder); throws InsufficientKarmaException, MetacatException, MetacatInaccessibleException;	Set permissions for an XML document in the Metacat repository.
setMetacatUrl	public void setMetacatUrl (String metacatUrl);	Set the MetacatUrl to which connections should be made.
setSessionId	public void setSessionId (String sessionId);	Set the session identifier for this session.
update	public String update (String docid, Reader xmlDocument, Reader schema) throws InsufficientKarmaException, MetacatException, IOException, MetacatInaccessibleException;	Update an XML document in the repository by providing a new version of the XML document.
upload	public String upload (String docid, File file) throws InsufficientKarmaException, MetacatException, IOException, MetacatInaccessibleException;	Upload a data document into the repository.
upload	public String upload (String docid, String fileName, InputStream fileData, int size) throws InsufficientKarmaException, MetacatException, IOException, MetacatInaccessibleException;	Upload a data document into the repository.

Table 4.4: Operations provided by client API. For more information, please see either the javadocs.

5 Metacat's Use of GeoServer

GeoServer 2.0.2, an open source Web Mapping Service (WMS) written in Java, is bundled with Metacat and can be used to render spatial data as web-deliverable maps (Figure 5.1). Metacat uses OpenLayers (<http://openlayers.org/>) to provide a web-based user interface for interacting with the generated maps. You can use any WMS-compatible client (e.g., ArcGIS, QGIS, JUMP, UDig, OpenLayers, Mapbender, Map Builder).

IMPORTANT: Regardless of whether you plan on using the mapping functionality you should, for security purposes, configure GeoServer so that it doesn't use the default password. For instructions, please see [Geoserver Configuration](#).

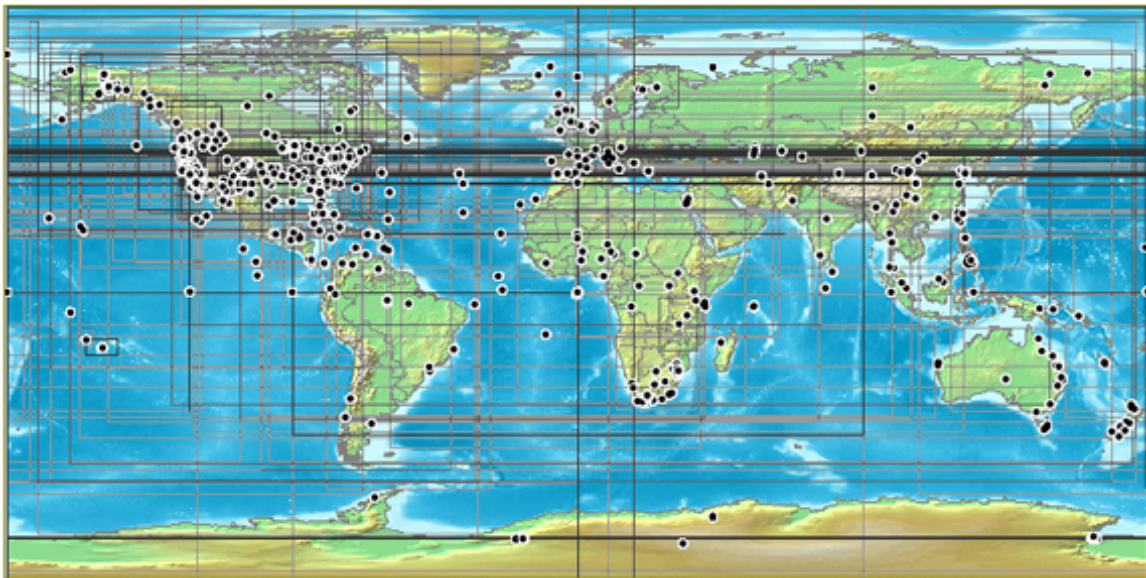


Figure 5.1: A map generated by Metacat's GeoServer. Points and "bounding boxes" represent the geographic extent of datasets stored in the KNB Metacat repository.

GeoServer supports a wide variety of vector GIS data sources, which can be styled using Styled Layer Descriptors (SLDs) and output as images (the default) or raw vector data (GML or KML).

Currently, GeoServer can be used with the following limitations:

- GeoServer will only map documents that are publicly available. This is because the mapping server's support for permissions control is not as fine-grained as Metacat's.

Metacat developers plan to continue extending and improving Metacat's mapping capabilities. If you are interested in contributing to those efforts, or if you are interested in learning more about the architecture and future plans for the mapping software, please contact the Metacat development team (metacat-dev@ecoinformatics.org).

5.1 Installing and Configuring

The GeoServer webapp should be installed as a sibling of Metacat. If you do NOT wish to run GeoServer, the deployment can be skipped, but any skins that use maps will not render correctly.

Metacat comes with a pre-configured data directory to be used by GeoServer. This includes a world-countries base layer and a default configuration that is already aware of Metacat's spatial cache. The Metacat configuration interface is used to configure GeoServer to use this shared data directory. To further configure GeoServer, use the Web-based configuration utility (Figure 5.2), which is available at:

http://your.server.com/<geoserver_context> (e.g., <http://knb.ecoinformatics.org/geoserver>).

Common configuration tasks include:

- Adding a Map to a Web Page or Skin
- Configuring the Size and Initial Extent of the Map
- Configuring the Layout of the HTML Mapping Interface
- Configuring the "Select Location" Drop-down Menu
- Configuring the Visual Portrayal of Geospatial Data (e.g., symbology and color)
- Adding Other Spatial Datasets to the Web Map

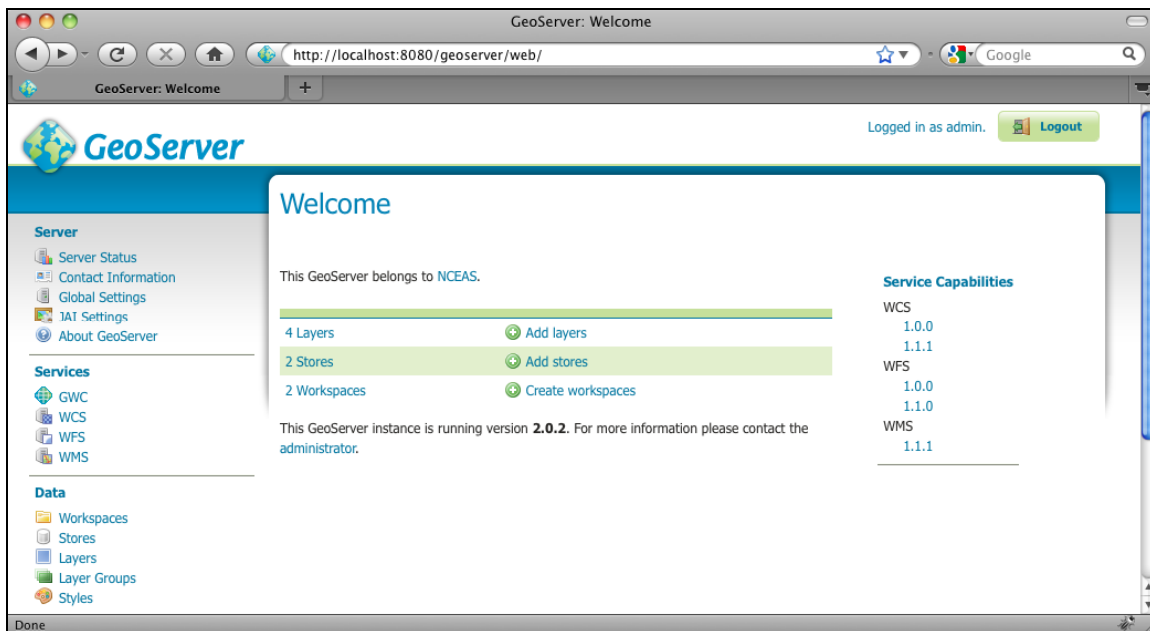


Figure 5.2: GeoServer's Web-based administrative interface.

OpenLayers, which Metacat uses as the front-end for GeoServer's WMS service, provides interface components or "widgets" (e.g., the map, a box zoom, layer list, "Select Location" drop-down menu, scale bar, lat/long coordinates, and a query form) that make it easy to deploy web-based mapping applications with minimal coding.

OpenLayers has three main configuration files used to customize the map interface (Table 5.1). Default configurations are in:

```
$METACAT/lib/style/common/spatial/
```

Document	Location	Description
The named location file	locations.jsp	The list of pre-defined locations (name and lat/lon bounds)
Main map rendering functions	map.js	Defines the map, widgets and their behavior
The rendered map and page layout	map.jsp	Loads the map and controls the HTML layout of the widgets.

Table 5.1: The main configuration files used to configure the mapping user interface. The default configuration files are located in \$METACAT/lib/style/common/spatial/

NOTE: By default, the first time Metacat is restarted, it generates a "spatial cache" containing geographic information about documents in its repository. This default behavior is specified in `lib/metacat.properties`, where the `regenerateCacheOnRestart` parameter is set to `true`. The information in the spatial cache is stored in a GIS-compatible format (the ESRI Shapefile) and consists of the document name and its geographic coverage. When documents are inserted, deleted, and updated in the Metacat repository, Metacat automatically syncs the spatial cache to reflect the changes. Because generating the cache can take a considerable amount of time (several minutes in the case of a few thousand documents), Metacat resets the `regenerateCacheOnRestart` property to `false` after the spatial cache has been generated. Note that if you upgrade or reinstall Metacat, the spatial cache will be regenerated again.

5.1.1 Adding a Map to a Web Page or Skin

To add a map to a Web page, simply include the map interface using an `iframe`:

```
<iframe scrolling="no" frameborder="0" width="780" height="420"
    src="/knb/style/common/spatial/map.jsp">
</iframe>
```

The map URL, `/knb/style/common/spatial/map.jsp`, is the default map interface. If you plan to customize the map interface, copy the `map.jsp` file into your skin's directory (either the `default` or `customized` skin directory).

```
cp -r style/common/spatial/map.jsp /style/skins/<myskin>/spatial
```

You can access the customized map with the URL:
`/knb/style/skins/<myskin>/spatial/map.jsp`

5.1.2 Configuring the Size and Initial Extent of the Map

Before you configure the size and initial extent of the map, make sure that you have copied the map layout page into your skin's directory (See section 5.1.1 for directions). Once the file has been copied, you can modify the map's initial extent in: `${skin.dir}/spatial/map.jsp`.

To change the map's initial extent, edit the bounding box. The default is to show the entire globe. The `initMap()` function should also be given the skin name so that spatial search results can be correctly styled.

```
<script type="text/javascript">
  function init() {
    var bounds = new OpenLayers.Bounds(-180,-90,180,90);
    // make the map for this skin
    initMap("<%=GEOSERVER_URL%>", "<%=SERVLET_URL%>",
"default", bounds);
  }
</script>
```

The size (height/width) of the map can be controlled by the `#map` CSS entry included in the `map.jsp` page.

5.1.3 Configuring the Layout of the HTML Mapping Interface

The size and initial extent of the map can be edited in: `${skin.dir}/spatial/map.jsp`.

The `map.jsp` is a simple container that can be included in other more complex pages if desired. It contains the map, widgets and location dropdown list.

5.1.4 Configuring the "Select Location" Drop-down Menu

The locations that appear in the "Select Location" drop-down menu are specified in the `locations.jsp` file. The `locations.jsp` can be copied from the common spatial template into your skin directory. Each location is defined as an HTML `<option/>` tag. Edit the `value` and `label` to edit or add new locations.

```
<option value="-149.725,68.475 -149.3254,68.725">
  Arctic LTER (ARC)
</option>
```

5.1.5 Configuring the Visual Portrayal of Geospatial Data (e.g., symbology and color)

Geospatial data sets are styled through the use of Styled Layer Descriptors (SLD). The default SLDs used for the data points and data bounding boxes are in

`/lib/spatial/geoserver/data/styles/` and are named `data_points_style.sld` and `data_bounds_style.sld`, respectively.

You can find a more detailed tutorial on using SLD with GeoServer in the GeoServer documentation: <http://docs.geoserver.org/>.

5.1.6 Adding Other Spatial Datasets to the Web Map

If you have vector GIS data sets, such as weather or topographical information, on your server that you'd like to include in the interactive map, you must first register the data set with GeoServer. After the data set has been registered, you can add the layer to the map. You can also add spatial layers that have been made publically available through WMS (There are hundreds of spatial data sets available. Check out wms-sites.com for good catalog). Instructions for adding publically available layers are included at the end of this section.

To register the data set and add it to the map:

- 1) Point your browser to <http://your.server/geoserver>, log in to GeoServer, and navigate to the "Data Stores" configuration page under Data > Stores.
- 2) Create a new vector data source from a Shapefile in the "metacat" workspace (Figure 5.3).

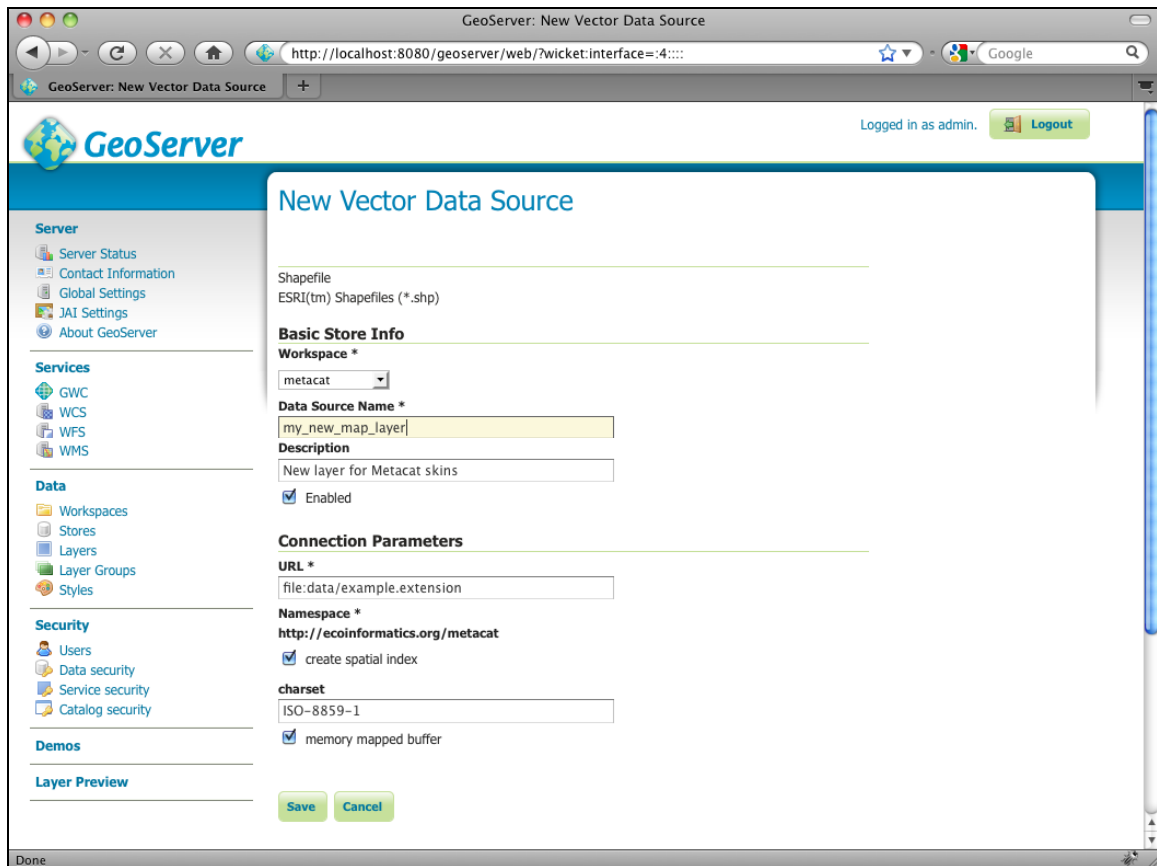


Figure 5.3: Creating a new shapefile using GeoServers web-based administrative interface.

- 3) Point to the GIS data file on the file system. Use an absolute path or a relative to the GeoServer data directory that is configured in Metacat.

The Description, if specified, is mostly used internally to provide other administrators with information about the Data Store. Click Submit.

- 4) Navigate to the "Layers" configuration page under Data > Layers. Add a new Layer from your new data source.
- 5) You should also define a spatial reference system (SRS) number for the new layer. Most lat/long data is "4326". If your data is in another projection, determine its spatial reference system using the help links provided (Figure 5.4).

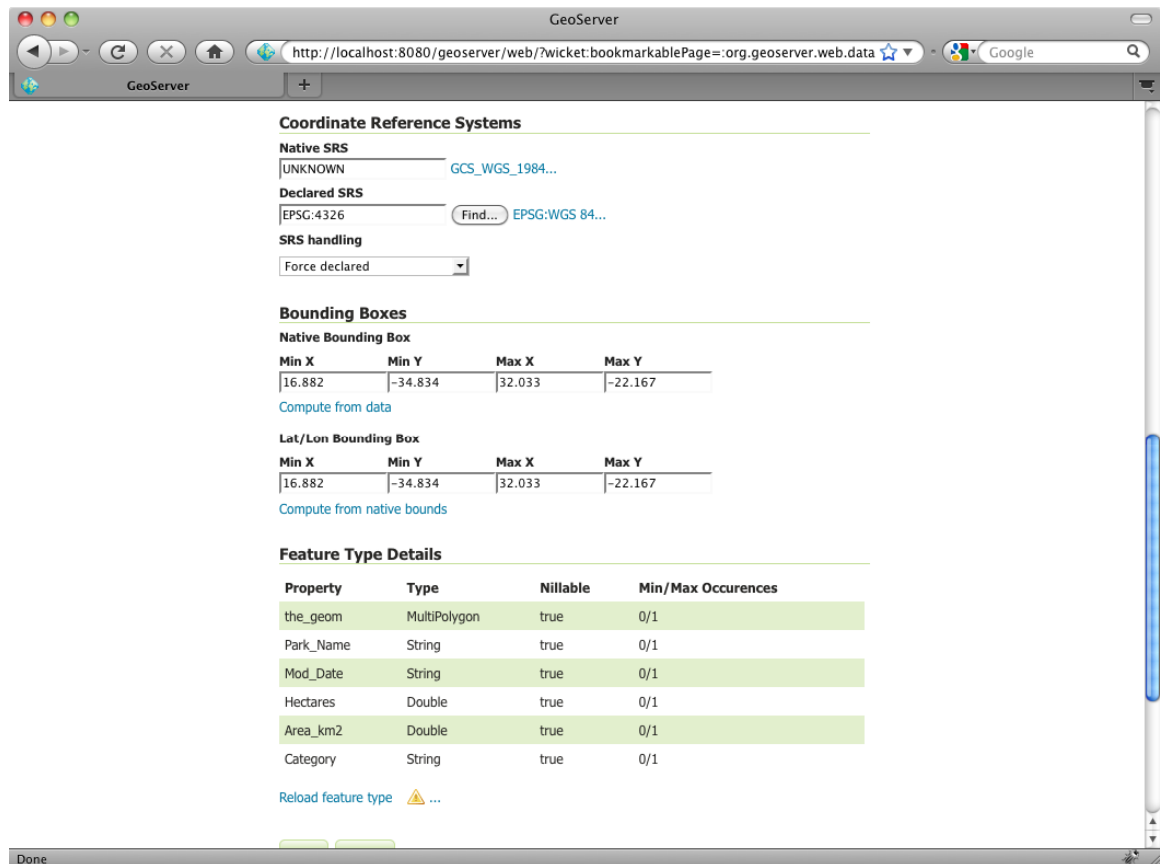


Figure 5.4: GeoServer's FeatureType configuration. The SRS settings discussed in step 5 are highlighted.

- 6) Style the layer using a style from the drop-down menu on the Publishing tab, or create a new SLD to create a new style object and corresponding SLD (this option provides more control over the style).
- 7) Try out the styled data set as a WMS layer using a the Layer Preview (Figure 5.5)

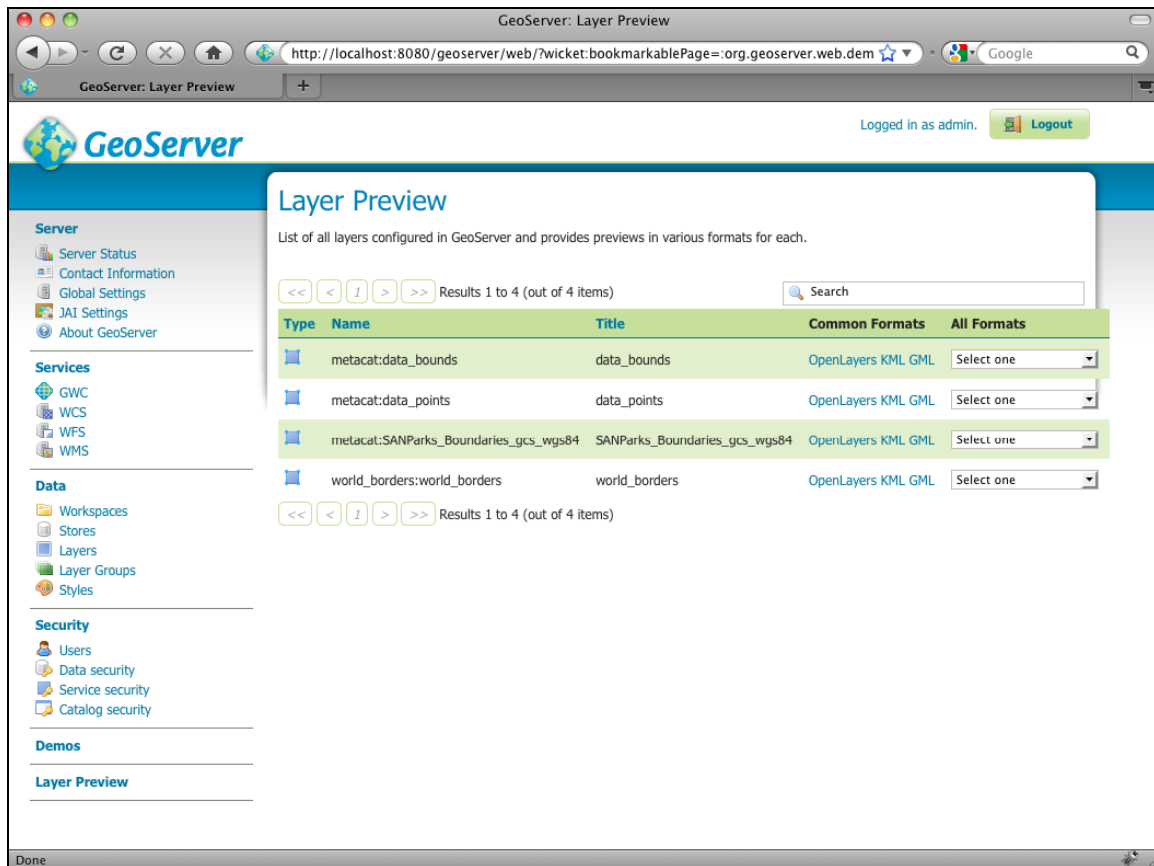


Figure 5.5: GeoServer's Layer Preview allows you to see an OpenLayer rendering of the new layer.

- 8) Copy the default map.js file that assembles the map in OpenLayers (`style/common/spatial/map.js`) to your skin's spatial directory.
- 9) Edit the `init()` method to include your new layer in the map – either as an overlay or as a base layer.
- 10) Point your browser to the map interface. Your new layer should appear with the existing ones.

Adding External Spatial Data Made Publicly Available through WMS

There are hundreds of sources of spatial data made publically available through WMS. (check out wms-sites.com for good catalog). To add these data sources to your map, add the layers in your skin's `spatial/map.js` file.

5.2 Spatial Queries

To find out which documents in the Metacat repository lie in a specified geographic region, query the spatial cache using Metacat's `spatial_query` action. Metacat can perform any query supported by the WFS/WMS standards.

An example of a spatial query string is:

```
http://localhost/knb/metacat?action=spatial_query&xmin=-117.5&xmax=-64&ymin=3&ymin=46&skin=default
```

Where `xmin`, `xmax`, `ymin` and `ymax` represent the western, eastern, southern and northern bounding coordinates (the "bounding box"), respectively. The spatial query action returns all documents that overlap or that are contained inside the specified spatial coordinates. The result set is returned as HTML using the style of the specified skin (in this example, `default`).

6 Replication

Metacat has a built-in replication feature that allows different Metacat servers to share data (both XML documents and data files) between each other. Metacat can replicate not only its home server's original documents, but also those that were replicated from partner Metacat servers. When changes are made to one server in a replication network, the changes are automatically propagated to the network, even if the network is down.

Replication allows users to manage their data locally and (by replicating them to a shared Metacat repository) to make those data available to the greater scientific community via a centralized search. In other words, your Metacat can be part of a broader network, but you retain control over the local repository and how it is managed.

For example, the KNB Network (Figure 6.1), which currently consists of ten different Metacat servers from around the world, uses replication to "join" the disparate servers to form a single robust and searchable data repository--facilitating data discovery, while leaving the data ownership and management with the local administrators.

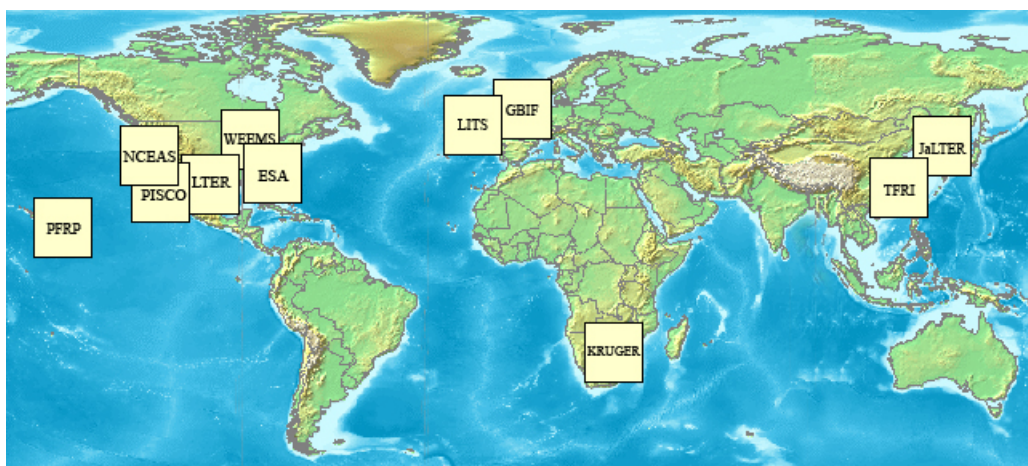


Figure 6.1: A map of the KNB Metacat network.

When properly **configured**, Metacat's replication mechanism can be triggered by several types of events that occur on either the home or partner server: a document insertion, an update, or an automatic replication (i.e., Delta-T monitoring), which is set at a user-specified time interval (Table 6.1).

Replication Triggers	Description
Insert	Whenever a document is inserted into Metacat, the server notifies each server in its replication list that it has a new file available.
Update	Whenever a document is updated, the server notifies each server in its replication list of the update.
Delta-T monitoring	At a user-specified time interval, Metacat checks each of the servers in its replication list for updated documents.

Table 6.1: Events that can trigger Metacat's replication mechanism.

6.1 Configuring Replication

To configure replication, you must configure both the home and partner servers:

- 1) Create a list of partner servers on your home server using the [Replication Control Panel](#)
- 2) [Create certificate](#) files for the home server
- 3) [Create certificate](#) files for the partner server
- 4) [Import partner certificate files](#) to the home server
- 5) [Import home certificate](#) to the partner server
- 6) [Update your Metacat database](#)

Each step is discussed in more detail in the following sections.

6.1.1 Using the Replication Control Panel

To add, remove, or alter servers on your home server's Replication list, or to activate and customize the Delta-T handler, use the Replication control panel (Figure 6.2), which is accessed at the following URL:

<http://somehost.somelocation.edu/context/style/skins/dev/replControl.html>

"<http://somehost.somelocation.edu/context>" should be replaced with the name of your Metacat server and context (e.g., <http://knb.ecoinformatics.org/knb/>). You must be logged in to Metacat as an administrator.

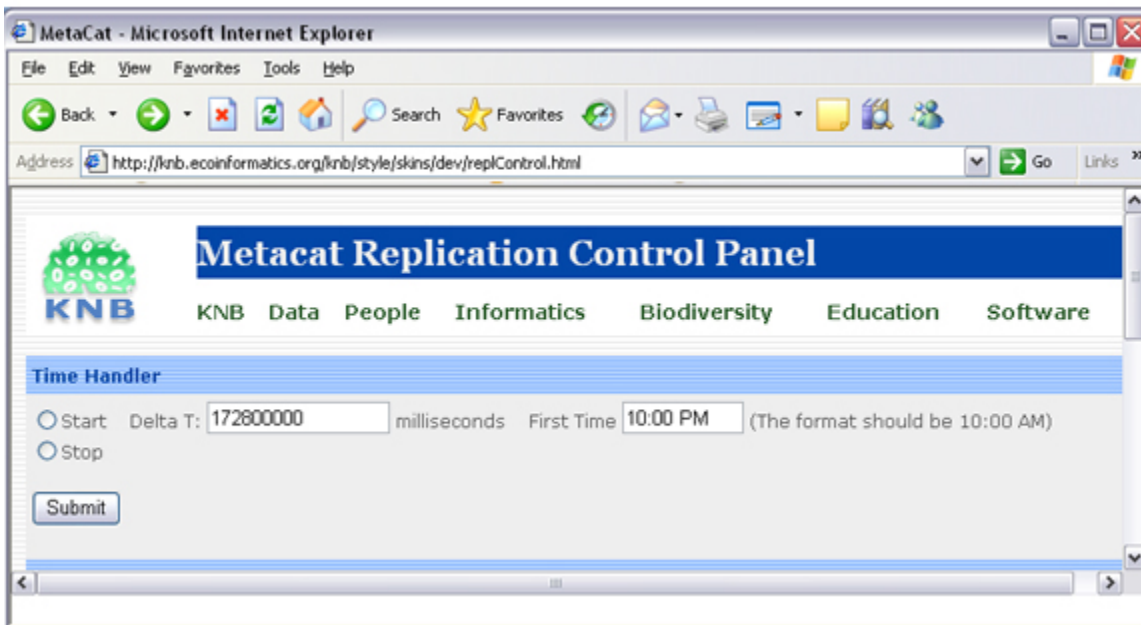


Figure 6.2: Replication control panel.

Note that currently, you cannot use the Replication Control Panel to remove a server after a replication has occurred. At this point in time, the only way to remove a replication server after replication has occurred is to remove the certificates.

Also note that you must SCP partner certificates to your machine; you cannot use the "Download Certificate from" option on the Control Panel. For more information about creating and installing certificates, please see [Generating and Exchanging Security Certificates](#).

6.1.2 Generating and Exchanging Security Certificates

Before you can take advantage of Metacat's replication feature, you must generate security certificates on both the replication partner and home servers. The certificates will be exchanged so that each machine understands that the other has replication access.

The process for generating certificates is different for Metacat servlets running under Tomcat and those under Tomcat/Apache (the recommended configuration). For instructions on generating and exchanging certificates on systems running only Tomcat 5 (and Java 6), see [Generating a Certificate for Tomcat standalone \(no Apache\)](#).

Generate Certificates for Metacat running under Apache/Tomcat

Note: Instructions are for Ubuntu/Debian systems.

- 1) Generate a certificate key using openssl. The key will be named `<hostname>-apache.key`, where `<hostname>` is the name of your Metacat server. Example values for the individual key fields are included in Table 6.2.

```
openssl req -new -out REQ.pem -keyout <hostname>-apache.key
```

Key Field	Description and Example Value
Country Name	Two letter country code (e.g., US)
State or Province Name	The name of your state or province spelled in full (e.g., California)
Locality Name	The name of your city (e.g., Santa Barbara)
Organization Name	The company or organization name (e.g., UCSB)
Organizational Unit Name	The department or section name (e.g., NCEAS)
Common Name	The host server name without port numbers (e.g., myserver.mydomain.edu)
Email Address	Administrator's contact email (e.g., administrator@mydomain.edu)
A challenge password	--leave this field blank--
An optional company name	--leave this field blank--

Table 6.2: Example values for certificate fields.

- 2) Create the local certificate file by running the command:

```
openssl req -x509 -days 800 -in REQ.pem -key <hostname>-  
apache.key -out <hostname>-apache.crt
```

Use the same <hostname> you used when you generated the key.

A file named <hostname>-apache.crt will be created in the directory from which you ran the `openssl` command. Note: You can name the certificate file anything you'd like, but keep in mind that the file will be sent to the partner machine used for replication. The certificate name should have enough meaning that someone who sees it on that machine can figure out where it came from.

- 3) Enter the certificate into Apache's security configuration. You must register the certificate in the local Apache instance. Note that the security files may be in a different directory from the one used in the instructions depending on how you installed Apache.

Copy the certificate and key file using the following commands

```
sudo cp <hostname>-apache.crt /etc/ssl/certs  
sudo cp <hostname>-apache.key /etc/ssl/private
```

- 4) Apache needs to know about Metacat SSL. The helper file named "knb-ssl" has rules that tell Apache which traffic to route to the Metacat SSL port. Set up SSL by dropping the knb-ssl file into the `sites-available` directory and running `a2ensite` to enable the site:

```
sudo cp <metacat_helper_dir>/knb-ssl <apache_install_dir>/sites-  
available  
sudo a2ensite knb-ssl
```

- 5) Restart Apache to bring in changes by typing:

```
sudo /etc/init.d/apache2 restart
```

- 6) SCP <hostname>-apache.crt to the replication partner machine.

Generating a Certificate for Tomcat standalone (no Apache)

If you are running Metacat under Tomcat (no Apache), generate keys in the Java default key store. The generated key is placed into the binary certificate's file located at `/etc/java-1.5.0-sun/security/cacerts`.

- 1) Generate the key by running the command. Note that you must be logged in as the root user to use the keytool:

```
keytool -genkey -alias <aliasname> -keyalg RSA -
  validity 800 -keystore /etc/java-1.6.0-
  sun/security/cacerts
```

`<aliasname>` is a unique name that you choose for this key. Something like "`<hostname-tomcat>`" might be appropriate, where `<hostname-tomcat>` is the name of the Metacat host.

- 2) The Password-keytool will ask for a password. If writing to a pre-existing keystore, you must know the password. If you are creating a new keystore, the password you enter will become the keystore password.

Sample values when creating certificate:

What is your first and last name? `myserver.nceas.ucsb.edu` (note: use the host name without port number)

What is the name of your organizational unit? `NCEAS`

What is the name of your organizational unit? `UCSB`

What is the name of your City or Locality? `Santa Barbara`

What is the name of your State or Province? `California` (note: this is spelled in full)

What is the two-letter country code for this unit? `US`

- 3) Create a certificate by running the command:

```
keytool -export -alias <aliasname> -file
  <outputfile>.cert -keystore /etc/java-1.6.0-
  sun/security/cacerts
```

`<aliasname>` is the same name you used when you created the key file.

A file named `<outputfile>.cert` will be created in the directory from which you ran the keytool command. You can name the output file anything you like, but keep in mind that it will be sent to the partner machine used for replication. The filename should have enough meaning that someone who sees it on that machine can figure out where it came from. Again, something like "`<hostname>-tomcat.cert`" will suffice.

- 4) Edit the Tomcat server file at `$TOMCAT_HOME/conf/server.xml` to enable SSL in Tomcat.

- a. Uncomment the section that starts with

```
"<Connector port="8443" ..."
```

(Note: Databased Information comments start with `<!--` and end with `-->`).

- b. Add two attribute to that section:

```
keystoreFile="/etc/java-1.6.0-sun/security/cacerts"
keystorePass="<keystore_password>"
```

where `<keystore_password>` is the password you used when you created or accessed the keystore.

- 5) SCP the certificate to the partner server.

After you have created and SCP'd a certificate file to each replication partner, and received a certificate file from each partner in return, both home and partner servers must import the respective partner certificates.

To import a certificate:

- 1) Log in as a root user (the keytool must run as a root user)

```
sudo su -
```

- 2) Import the remote certificate by running:

```
keytool -import -alias <remotehostalias> -file <remotehostfilename>.cert  
-keystore /etc/java-1.6.0-sun/security/cacerts
```

where the `<remotehostfilename>` is the name of the certificate file created on the remote partner machine and SCP'd to the home machine. The `<remotehostalias>` is the name the certificate will use in the keystore. The name should identify the remote host.

6.1.3 Update your Metacat database

The simplest way to update the Metacat database to use replication is to use the Replication Control Panel (Figure 6.3). You can also update the database using SQL. Instructions for both options are included in this section.

Figure 6.3: Using the Replication Control Panel to update the Metacat database.

To update your Metacat database to use replication, select the "Add this server" radio button from the Replication Control Panel, enter the partner server name, and specify how the replication should occur (whether to replicate xml, data, or use the local machine as a hub). Note that you cannot download certificates using this interface.

To update the database using SQL:

- 1) Log in to the database

```
psql -U metacat -W -h localhost metacat
```

- 2) Select all rows from the replication table

```
select * from xml_replication;
```

- 3) Insert the partner server.

```
INSERT INTO xml_replication  
(server,last_checked,replicate,datareplicate,hub) VALUES  
( '<partner.server/context>/servlet/replication',NULL,1,1,0);
```

Where `<partner.server/context>` is the name of the partner server and context. The values 'NULL, 1,1,0' indicate (respectively) the last time replication occurred, that XML docs should be replicated to the partner server, that data files should be replicated to the partner server, and that the local server should not act as a hub. Set a value of 'NULL,0,0,0' if your Metacat is only receiving documents from the partner site and not replicating to that site.

- 4) Exit the database
- 5) Restart Apache and Tomcat on both home and partner replication machines

7 Harvester and Harvest List Editor

Metacat's Harvester is an optional feature that can be used to automatically retrieve EML documents from one or more custom data management system (e.g., SRB or PostgreSQL) and to insert (or update) those documents to the home repository. The local sites control when they are harvested, and which documents are harvested.

For example, the [Long Term Ecological Research Network \(LTER\)](#) uses the Metacat Harvester to create a centralized repository of data stored on twenty-six different sites that store EML metadata, but that use different data management systems. Once the data have been harvested and placed into a centralized repository, they are [replicated](#) to the KNB network, exposing the information to an even larger scientific community.

Once the Harvester is properly configured, listed documents are retrieved and uploaded on a regularly scheduled basis. You must configure both the home Metacat and the remote sites (aka the "harvest sites") before using this feature. Local sites must also provide the Metacat server with a list of documents that should be harvested.

7.1 Configuring Harvester

Before you can use the Harvester to retrieve documents, you must configure the feature using the settings in the [metacat.properties](#) file. Note that you must also configure each site that the Harvester will connect to and retrieve documents from (see [section 7.2](#) for details).

The Harvester configuration information is managed in the [metacat.properties](#) file, which is located at:

```
<CONTEXT_DIR>/WEB_INF/metacat.properties
```

The Harvester properties are grouped together and begin after the comment line:

```
# Harvester properties
```

To configure Harvester, edit the `metacat.properties` and set appropriate values for the `harvesterAdministrator` and `smtpServer` property. You may also wish to customize the other Harvester parameters, each discussed in Table 7.1.

Harvester Properties and their Functions

Property	Description and Values
connectToMetacat	Determine whether Harvester should connect to Metacat to upload retrieved documents. Set to <code>true</code> (the default) under most circumstances. To test whether Harvester can retrieve documents from a site without actually connecting to Metacat to upload the documents, set the value to <code>false</code> . Values: <code>true/false</code>
delay	The number of hours that Harvester will wait before beginning its first harvest. For example, if Harvester is run at 1:00 p.m., and the delay is set to 12, Harvester will begin its first harvest at 1:00 a.m. Default: 0
harvesterAdministrator	The email address of the Harvester Administrator. Harvester will send email reports to this address after every harvest. Enter multiple email addresses by separating each address with a comma or semicolon (e.g., <code>name1@abc.edu,name2@abc.edu</code>). Values: An email address, or multiple email addresses separated by commas or semi-colons
logPeriod	The number of days to retain Harvester log entries. Harvester log entries record information such as which documents were harvested, from which sites, and whether any errors were encountered during the harvest. Log entries older than <code>logPeriod</code> number of days are purged from the database at the end of each harvest. Default: 90
maxHarvests	The maximum number of harvests that Harvester should execute before shutting down. If the value of <code>maxHarvests</code> is set to 0 or a negative number, Harvester will execute indefinitely. Default: 0
period	The number of hours between harvests. Harvester will run a new harvest every specified period of hours (either indefinitely or until the maximum number of harvests have run, depending on the value of <code>maxHarvests</code>). Default: 24
smtpServer	The SMTP server that Harvester uses for sending email messages to the Harvester Administrator and Site Contacts. (e.g., <code>somehost.institution.edu</code>). Note that the default value only works if the Harvester host machine is configured as a SMTP server. Default: <code>localhost</code>
Harvester Operation Properties (<code>GetDocError</code> , <code>GetDocSuccess</code> , etc.)	The Harvester Operation properties are used by Harvester to report information about performed operations for inclusion in log entries and email messages. Under most circumstances the values of these properties should not be modified.

Table 7.1: Harvester Properties found in the `metacat.properties` file.

7.2 Configuring a Harvest Site (Instructions for Site Contact)

After Metacat's Harvester has been configured, remote sites can register and send information about which files should be retrieved. Each remote site must have a site contact who is responsible for registering the site and creating a list of EML files to harvest (the "Harvest List"), as well as for reviewing harvest reports. The site contact can [unregister the site](#) from the Harvester at any time.

To use Harvester:

- 1) [Register with Harvester](#)
- 2) [Compose a Harvest List](#) (you will likely wish to use the Harvest List Editor)
- 3) [Prepare your EML Documents for Harvest](#)
- 4) [Review the Harvester Reports](#)

7.2.1 Register with Harvester

To register a remote site with Harvester, the Site Contact should log in to Metacat's Harvester Registration page and enter information about the site and how it should be harvested.

- 1) Using a Web browser, log in to Metacat's Harvester Registration page (Figure 7.1). The Harvester Registration page is inside the skins directory. For example, if the Metacat server that you wish to register with resides at the following URL:

```
http://somehost.somelocation.edu:8080/knb/index.jsp
```

then the Harvester Registration page would be accessed at:

```
http://somehost.somelocation.edu:8080/knb/style/skins/knb/harvesterRegistrationLogin.jsp
```

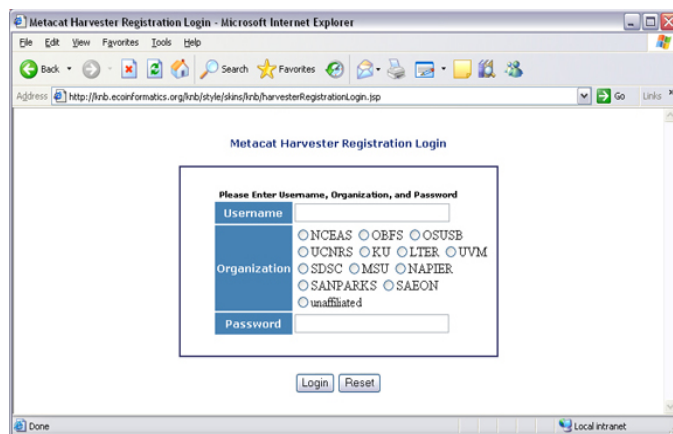


Figure 7.1: Metacat's Harvester Registration page.

- 2) Enter your Metacat account information and click Submit to log in to your Metacat from the Harvester Registration page.

Note: In some cases, you may need to log in to an anonymous "site" account rather than your personal account so that the registered data will not appear to have been registered by a single user. For example, an information manager (jones) who is registering data created by a team of scientists (jones, smith, and barney) from the Georgia Coastal Ecosystems site might log in to a dedicated account (named with the site's acronym, "GCE") to indicate that the registered data is from the entire site rather than "jones".

- 3) Enter information about your site and how often you want to schedule harvests and then click the Register button (Figure 7.2). The Harvest List URL should point to the location of the Harvest List, which is an XML file that lists the documents to harvest. If you do not yet have a Harvest List, please see Section 7.2.2 for more information about creating one.

Metacat Harvester Registration - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Refresh Home Search Favorites

Address <http://knb.ecoinformatics.org/knb/harvesterRegistration> Go Links

Metacat Harvester Registration

Fill out the form below to schedule regular harvests of EML documents from your site. To register or changes values, enter all values below and click **Register**. To unregister, simply click **Unregister**.

Email address:

Harvest List URL:

Harvest Frequency

Once every (1-99):

day(s) week(s) month(s)

Done Local intranet

Figure 7.2 Enter information about your site and how often you want to schedule harvests.

The example settings in Figure 7.2, instruct Harvester to harvest documents from the site once every two weeks. The Harvester will access the site's Harvest List at URL "http://somehost.institution.edu/~myname/harvestList.xml", and will send email reports to the Site Contact at email address "myname@institution.edu". Note that you can enter multiple email addresses by separating each address with a comma or a semi-colon. For example, "myname@institution.edu,anothername@institution.edu".

7.2.2 Compose a Harvest List (The Harvest List Editor)

The Harvest List is an XML file that contains a list of documents to be harvested. The list is created by the site contact and stored on the site contact's site at the location specified during the Harvester registration process (see Section 7.2.1 for details). The list can be generated by hand, or you can use Metacat's Harvest List Editor to automatically generate and structure the list to conform to the required XML schema (displayed in Figure 7.6 at the end of this section). In this section we will look at what information is required when building a Harvest List, and how to configure and use the Harvest List Editor. Note that you must have a source distribution of Metacat in order to use the Harvest List Editor.

The Harvest List contains information that helps Metacat identify and retrieve each specified EML file. Each document in the list must be described with a `docid`, `documentType`, and `documentURL` (Table 7.2).

Item	Description
<code>docid</code>	<p>The <code>docid</code> uniquely identifies each EML document. Each <code>docid</code> consists of three elements:</p> <p><code>scope</code> The document group to which the document belongs</p> <p><code>identifier</code> A number that uniquely identifies the document within the scope.</p> <p><code>revision</code> A number that indicates the current revision.</p> <p>For example, a valid <code>docid</code> could be: <code>demoDocument.1.5</code>, where <code>demoDocument</code> represents the scope, 1 the identifier, and 5 the revision number.</p>
<code>documentType</code>	<p>The <code>documentType</code> identifies the type of document as EML e.g., "<code>eml://ecoinformatics.org/eml-2.0.0</code>".</p>
<code>documentURL</code>	<p>The <code>documentURL</code> specifies a place where Harvester can locate and retrieve the document via HTTP. The Metacat Harvester must be given read access to the contents at this URL. e.g. "<code>http://www.lternet.edu/~dcosta/document1.xml</code>".</p>

Table 7.2: Information that must be included in the Harvest List about each EML file

The example Harvest List in Figure 7.3 contains two `<document>` elements that specify the information that Harvester needs to retrieve a pair of EML documents and upload them to Metacat.

Example Harvest List

```
<?xml version="1.0" encoding="UTF-8" ?>
<hrv:harvestList xmlns:hrv="eml://ecoinformatics.org/harvestList" >
  <document>
    <docid>
      <scope>demoDocument</scope>
      <identifier>1</identifier>
      <revision>5</revision>
    </docid>
    <documentType>eml://ecoinformatics.org/eml-2.0.0</documentType>
    <documentURL>http://www.lternet.edu/~dcosta/document1.xml</documentURL>
  </document>
  <document>
    <docid>
      <scope>demoDocument</scope>
      <identifier>2</identifier>
      <revision>1</revision>
    </docid>
    <documentType>eml://ecoinformatics.org/eml-2.0.0</documentType>
    <documentURL>http://www.lternet.edu/~dcosta/document2.xml</documentURL>
  </document>
</hrv:harvestList>
```

Figure 7.3: Example of a valid Harvest List

Rather than formatting the list by hand, you may wish to use Metacat's Harvest List Editor to compose and edit it. The Harvest List Editor displays a Harvest List as a table of rows and fields (Figure 7.4). Each table row corresponds to a single <document> element in the corresponding Harvest List file (i.e., one EML document). The row numbers are used only for visual reference and are not editable.

Screenshot of Harvest List Editor

Figure 7.4: The Harvest List Editor

To add a new document to the Harvest List, enter values for *all five editable fields* (all fields except the "Row #" field). Partially filled-in rows will cause errors that will result in an invalid Harvest List.

The buttons at the bottom of the Editor can be used to Cut, Copy, and Paste rows from one location to another. Select a row and click the desired button, or paste the default values (which are specified in the Editor's configuration file, discussed later in this section) into the currently selected row by clicking the Paste Defaults button. Note: Only one row can be selected at any given time: all cut, copy, and paste operations work on only a single row rather than on a range of rows.

To run the Harvest List Editor, from the terminal on which the Metacat source code is installed:

- 1) Open a system command window or terminal window.
- 2) Set the METACAT_HOME environment variable to the value of the Metacat installation directory. Some examples follow:

On Windows:

```
set METACAT_HOME=C:\somePath\knb
```

On Linux/Unix (bash shell):

```
export METACAT_HOME=/home/somePath/metacat
```

3) cd to the following directory:

On Windows:

```
cd %METACAT_HOME%\lib\harvester
```

On Linux/Unix:

```
cd $METACAT_HOME/lib/harvester
```

4) Run the appropriate Harvester shell script, as determined by the operating system:

On Windows:

```
runHarvestListEditor.bat
```

On Linux/Unix:

```
sh runHarvestListEditor.sh
```

The Harvest List Editor will open.

If you would like to customize the Harvest List Editor (e.g., specify a default list to open automatically whenever the editor is opened and/or default values), create a file called `.harvestListEditor` (note the leading dot character). Use a plain text editor to create the file and place the file in the Site Contact's home directory. To determine the home directory, open a system command window or terminal window and type the following:

On Windows:

```
echo %USERPROFILE%
```

On Linux/Unix:

```
echo $HOME
```


The configuration file contains a number of optional properties that can make using the Editor more convenient. A sample configure file is displayed in Figure 7.5, and more information about each configuration property is contained in Table 7.3.

A sample .harvestListEditor configuration file

```
defaultHarvestList=C:/temp/harvestList.xml
defaultScope=demo_document
defaultIdentifier=1
defaultRevision=1
defaultDocumentURL=http://www.lternet.edu/~dcosta/
defaultDocumentType=eml://ecoinformatics.org/eml-2.0.0
```

Figure 7.5: A sample .harvestListEditor configuration file.

Harvest List Editor Configuration Properties

Property	Description
defaultHarvestList	The location of a Harvest List file that the Editor will automatically open for editing on startup. Set this property to the path to the Harvest List file that you expect to edit most frequently. Examples: /home/jdoe/public_html/harvestList.xml C:/temp/harvestList.xml
defaultScope	The value pasted into the Editor's Scope field when the Paste Defaults button is clicked. The Scope field should contain a symbolic identifier that indicates the family of documents to which the EML document belongs. Example: xyz_dataset Default: dataset
defaultIdentifier	The value pasted into the Editor's Identifier field when the Paste Defaults button is clicked. The Scope field should contain a numeric value indicating the identifier for this particular EML document within the Scope.
defaultRevision	The value pasted into the Editor's Revision field when the Paste Defaults button is clicked. The Scope field should contain a numeric value indicating the revision number of this EML document within the Scope and Identifier. Example: 2 Default: 1
defaultDocumentType	The document type specification pasted into the Editor's DocumentType field when the Paste Defaults button is clicked. Default: eml://ecoinformatics.org/eml-2.0.0
defaultDocumentURL	The URL or partial URL pasted into the Editor's URL field when the Paste Defaults button is clicked. Typically, this value is set to the portion of the URL shared by all harvested EML documents. Example: http://somehost.institution.edu/somepath/ Default: http://

Table 7.3: The configurable properties of the Metacat Harvester.

XML Schema for Harvest Lists

```

<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 rel. 4 U (http://www.xmlspy.com) by Matt Jones
(NCEAS) -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:hrv="eml://ecoinformatics.org/harvestList"
xmlns="eml://ecoinformatics.org/harvestList"
targetNamespace="eml://ecoinformatics.org/harvestList"
elementFormDefault="unqualified" attributeFormDefault="unqualified">
  <xs:annotation>
    <xs:documentation>This module defines the required information for the
harvester to collect documents from the local site. The local system containing
this document must give the Metacat Harvester read access to this
document.</xs:documentation>
  </xs:annotation>
  <xs:annotation>
    <xs:appinfo>
      <tooltip/>
      <summary/>
      <description/>
    </xs:appinfo>
  </xs:annotation>
  <xs:element name="harvestList">
    <xs:annotation>
      <xs:documentation>This represents the local document information that is
used to inform the Harvester of the docid, document type, and location of the
document to be harvested.</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="document" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="docid">
                <xs:annotation>
                  <xs:documentation>The complete document identifier to be used
by metacat. The docid is a compound element that gives a scope for the
identifier, an integer local identifier that is unique within that scope, and a
revision. Each revision is assumed to specify a unique, non-changing document,
so once a particular revision is harvested, there is no need for it to be
harvested again. To trigger a harvest of a document that has been updated,
increment the revision number for that identifier.</xs:documentation>
                </xs:annotation>
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="scope" type="xs:string">
                      <xs:annotation>
                        <xs:documentation>The system prefix of a metacat docid
that defines the scope within which the identifier is
unique.</xs:documentation>
                      </xs:annotation>
                    </xs:element>
                    <xs:element name="identifier" type="xs:long">
                      <xs:annotation>
                        <xs:documentation>The local (site specific) portion of
the identifier (docid) that is unique within the context of the
scope.</xs:documentation>
                      </xs:annotation>
                    </xs:element>
                    <xs:element name="revision" type="xs:long">
                      <xs:annotation>

```

```

        <xs:documentation>The revision identifier for this
document, indicating a unique document version.</xs:documentation>
        </xs:annotation>
    </xs:element>
    </xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="documentType" type="xs:string">
    <xs:annotation>
        <xs:documentation>The type of document to be harvested,
indicated by a namespace string, formal public identifier, mime type, or other
type indicator. </xs:documentation>
    </xs:annotation>
</xs:element>
<xs:element name="documentURL" type="xs:anyURI">
    <xs:annotation>
        <xs:documentation>The documentURL field contains the URL of
the document to be harvested. The Metacat Harvester must be given read access
to the contents at this URL.</xs:documentation>
    </xs:annotation>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>

```

Figure 7.6: XML Schema to which all Harvest Lists must conform. Access this file at: /lib/harvester/harvestList.xsd

7.2.3 Prepare EML Documents for Harvest

To prepare a set of EML documents for harvest, ensure that the following is true for each document:

- The document contains valid EML
- The document is specified in a <document> element in the site's Harvest List (See section 7.2.2)
- The file resides at the location specified by its URL in the Harvest List

7.2.4 Review Harvester Reports

Harvester sends an email report to the Site Contact after every scheduled site harvest. The report contains information about the performed operations, such as which EML documents were harvested and whether any errors were encountered. Errors are indicated by operations that display a status value of 1; a status value of 0 indicates that the operation completed successfully.

When errors are reported, the Site Contact should try to determine whether the source of the error is something that can be corrected at the site. Common causes of errors include:

- a document URL specified in the Harvest List does not match the location of the actual EML file on the disk
- the Harvest List does not contain valid XML as specified in the [harvestList.xsd](#) schema
- the URL to the Harvest List (specified during registration) does not match the actual location of the Harvest List on the disk
- an EML document that Harvester attempted to upload to Metacat does not contain valid EML

If the Site Contact is unable to determine the cause of the error and its resolution, he or she should contact the Harvester Administrator for assistance.

7.2.5 Unregister with Harvester

To discontinue harvests, the Site Contact must unregister with Harvester. To unregister:

- 1) Using a Web browser, log in to Metacat's Harvester Registration page. The Harvester Registration page is inside the skins directory. For example, if the Metacat server that you wish to register with resides at the following URL:

```
http://somehost.somelocation.edu:8080/knb/index.jsp
```

then the Harvester Registration page would be accessed at:

```
http://somehost.somelocation.edu:8080/knb/style/skins/knb/harvesterRegistrationLogin.html
```

- 2) Enter and submit your Metacat account information. On the subsequent screen, click Unregister to remove your site and discontinue harvests.

7.3 Running Harvester

The Harvester can be run as a [servlet](#) or in a [command window](#). Under most circumstances, Harvester is best run continuously as a background servlet process. However, if you expect to use Harvester infrequently, or if wish only to test that Harvester is functioning, it may desirable to run it from a command window.

7.3.1 Running Harvester as a Servlet

To run Harvester as a servlet (from a source code installation):

- 1) Remove the comment symbols around the `HarvesterServlet` entry (Figure 7.8) in the source code. The `HarvesterServlet` entry is located in the `lib/web.xml.tomcatN` file, where `tomcatN` corresponds to the version of Tomcat

you are running. For example, if you are running Tomcat 5, edit file `lib/web.xml.tomcat5`.

```

<!--
<servlet>
  <servlet-name>HarvesterServlet</servlet-name>
  <servlet-
class>edu.ucsb.nceas.metacat.harvesterClient.HarvesterServlet</s
ervlet-class>
  <init-param>
    <param-name>debug</param-name>
    <param-value>1</param-value>
  </init-param>
  <init-param>
    <param-name>listings</param-name>
    <param-value>>true</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>
-->

```

Figure 7.7: Remove the comment symbols (designated in red) from around the **HarvesterServlet** entry

- 2) Save the edited file.
- 3) Shut down Tomcat.
- 4) Redeploy Metacat by running the following two Ant commands from the top-level directory of your Metacat installation:

```

ant cleanweb
ant install

```

- 5) Restart Tomcat. Note that you will have to edit the `metacat.properties` file to specify harvester settings.

About thirty seconds after you restart Tomcat, the Harvester servlet will start executing. The first harvest will occur after the number of hours specified in the `metacat.properties` file (See Section 7.1 for information). The servlet will continue running new harvests until the maximum number of harvests have been completed, or until Tomcat shuts down (harvest frequency and maximum number of harvests are also set in the Harvester properties).

7.3.2 Running Harvester in a Command Window

To run Harvester in a Command Window:

- 1) Open a system command window or terminal window.
- 2) Set the `METACAT_HOME` environment variable to the value of the Metacat installation directory.

On Windows:

```
set METACAT_HOME=C:\somePath\metacat
```

On Linux/Unix (bash shell):

```
export METACAT_HOME=/home/somePath/metacat
```

5) cd to the following directory:

On Windows:

```
cd %METACAT_HOME%\lib\harvester
```

On Linux/Unix:

```
cd $METACAT_HOME/lib/harvester
```

6) Run the appropriate Harvester shell script, as determined by the operating system:

On Windows:

```
runHarvester.bat
```

On Linux/Unix:

```
sh runHarvester.sh
```

The Harvester application will start executing. The first harvest will occur after the number of hours specified in the [metacat.properties](#) file (See Section 7.1 for information). The servlet will continue running new harvests until the maximum number of harvests have been completed, or until you interrupt the process by hitting CTRL/C in the command window (harvest frequency and maximum number of harvests are also set in the Harvester properties).

7.4 Reviewing Harvest Reports

Harvester sends an email report to the Harvester Administrator after every harvest. The report contains information about the performed operations, such as which sites were harvested as well as which EML documents were harvested and whether any errors were

encountered. Errors are indicated by operations that display a status value of 1; a status value of 0 indicates that the operation completed successfully.

The Harvester Administrator should review the report, paying particularly close attention to any reported errors and accompanying error messages. When errors are reported at a particular site, the Harvester Administrator should contact the Site Contact to determine the source of the error and its resolution. Common causes of errors include:

- a document URL specified in the Harvest List does not match the location of the actual EML file on the disk
- the Harvest List does not contain valid XML as specified in the [harvestList.xsd](#) schema
- the URL to the Harvest List (specified during registration) does not match the actual location of the Harvest List on the disk
- an EML document that Harvester attempted to upload to Metacat does not contain valid EML

Errors that are independent of a particular site may indicate a problem with Harvester itself, Metacat, or the database connection. Refer to the error message to determine the source of the error and its resolution.

8 Event Logging

Metacat keeps an internal log of events (such as insertions, updates, deletes, and reads) that can be accessed with the `getlog` action. Using the `getlog` action, event reports can be output from Metacat in XML format, and/or customized to include only certain events: events from a particular IP address, user, event type, or that occurred after a specified start date or before an end date.

The following URL is used to return the basic log—an XML-formatted log of all events since the log was initiated (Figure 8.1):

<http://some.metacat.host/context/metacat?action=getlog>

Note that you must be logged in to Metacat using the HTTP interface or you will get an error message. For more information about logging in, please see [Logging In with the HTTP Interface](#).

Example of XML Log

```
<?xml version="1.0"?>
<log>
<logEntry><entryid>44</entryid><ipAddress>34.237.20.142</ipAddress><principal>uid=jones,
o=NCEAS,dc=ecoinformatics,dc=org</principal><docid>esa.2.1</docid><event>insert</event>
<dateLogged>2004-09-08 19:08:18.16</dateLogged></logEntry>
<logEntry><entryid>47</entryid><ipAddress>34.237.20.142</ipAddress><principal>uid=jones,o
=NCEAS,
dc=ecoinformatics,dc=org</principal><docid>esa.3.1</docid><event>insert</event><dateLogge
d>2004-
09-14 19:50:40.61</dateLogged></logEntry>
</log>
```

Figure 8.1: An example of a Metacat log in XML format.

The basic log can be quite extensive. To subset the report, restrict the matching events using parameters (Table 8.1). Query parameters can be combined to further restrict the report.

Parameter	Description and Values
ipAddress	Restrict the report to this IP Address (repeatable)
principal	Restrict the report to this user (repeatable)
docid	Restrict the report to this docid (repeatable)
event	Restrict the report to this event type (repeatable) Values: insert, update, delete, read
start	Restrict the report to events after this date Value: YYYY-MM-DD+hh:mm:ss
end	Restrict the report to events before this date. Value: YYYY-MM-DD+hh:mm:ss

Table 8.1: Parameters for the `getlog` action.

To view only the 'read' events, use a URL like:

```
http://some.metacat.host/context/metacat?action=getlog&event=read
```

To view only the events for a particular IP address, use a URL like:

```
http://some.metacat.host/context/metacat?action=getlog&ipaddress=107.9.1.31
```

To view only the events for a given user, use a URL like:

```
http://some.metacat.host/context/metacat?action=getlog&principal=uid=john  
ndoe,o=NCEAS,dc=ecoinformatics,dc=org
```

To view only the events for a particular document, use a URL like:

```
http://some.metacat.host/context/metacat?action=getlog&docid=knb.5.1
```

To view only the events after a given date, use a URL like:

```
http://some.metacat.host/context/metacat?action=getlog&start=2004-09-15+12:00:00
```

To view only the events before a given date, use a URL like:

```
http://some.metacat.host/context/metacat?action=getlog&end=2004-09-15+12:00:00
```

To view the 'insert' events for September 2004 (i.e., to combine parameters) use a URL like:

```
http://some.metacat.host/context/metacat?action=getlog&event=insert&start=2004-09-01+12:00:00&end=2004-09-30+23:59:59
```

9 Enabling Web Searches: Sitemaps

Sitemaps are XML files that tell search engines—such as Google, which is discussed in this section--which URLs on your websites are available for crawling. Currently, the only way for a search engine to crawl and index Metacat so that individual metadata entries are available via Web searches is with a sitemap. Metacat automatically [creates sitemaps](#) for all public documents in the repository. However, you must [register the sitemaps](#) with the search engine before it will take effect.

9.1 Creating a Sitemap

Metacat automatically generates a sitemap file for all public documents in the repository on a daily basis. The sitemap file(s) must be available via the Web on your server, and must be registered with Google before they take effect. For information on the sitemap protocol, please refer to [the Google page on using the sitemap protocol](#).

You can view Metacat's sitemap files at:

```
<webapps_dir>/sitemaps
```

The directory contains one or more XML files named

```
metacat<X>.xml
```

where <X> is a number (e.g., 1 or 2) used to increment each sitemap file. Because Metacat limits the number of sitemap entries in each sitemap file to 25,000, the servlet creates an additional sitemap file for each group of 25,000 entries.

Verify that your sitemap files are available to the Web by browsing to

```
<your_web_context>/sitemaps/metacat<X>.xml
```

(e.g., `your.server.org/knb/sitemaps/metacat1.xml`)

9.2 Registering a Sitemap

Before Google will begin indexing the public files in your Metacat, you must register the sitemaps. To register your sitemaps and ensure that they are up to date:

- 1) Register for a Google Webmaster Tools account, and add your Metacat site to the Dashboard.
- 2) From your Google Webmaster Tools site account, register your sitemaps. See [the Google help site](#) for more information about how to register sitemaps. Note:

Register the full URL path to your sitemap files, including the http:// (or https://) headers.

Once the sitemaps are registered, Google will begin to index the public documents in your Metacat repository.

NOTE: As you add more publicly accessible data to Metacat, you will need to periodically revisit the Google Webmaster Tools utility to refresh your sitemap registration.

10 Creating a Java Class that Implements AuthInterface

TO COME

11 Appendix: Metacat Properties

The most dynamic Metacat Properties are managed using [Metacat's Configuration Interface](#). These properties, as well as other, rarely modified ones can be found in the `metacat.properties` file. For more information about the properties, click one of the following:

- [Server Properties](#)
- [Application Properties](#)
- [Database Properties](#)
- [Authorization and Authentication Properties](#)
- [XML/EML Properties](#)

Server Properties

All of Metacat's server properties are managed with the form-based configuration utility, though they can also be accessed. More information on each is included in Table 10.1.

Metacat Server Properties

Property	Description	Example
<code>server.name</code>	The network host name used to access Metacat. Note that this is not necessarily the physical name of the server running Metacat. The host name should not include the protocol prefix (<code>http://</code>). Default Value: localhost	knb.ecoinformatics.org
<code>server.httpPort</code>	The network port used to access Metacat for non-secure (standard) connections. This is usually 80 if Apache Web server is running, and 8080 if Tomcat is running alone. Default Value: 80	80
<code>server.httpSSLPort</code>	The network port used to access Metacat for secure connections. This is usually 443 if Apache Web server is running, and 8443 if Tomcat is running alone. Default Value: 443	443

Table 11.1: Metacat Server Properties

Application Properties

Metacat's application properties are described in Table 10.2. Properties that can only be edited manually in the `metacat.properties` file are highlighted. All others are managed with the properties configuration utility.

Metacat Application Properties

Property	Description	Example
<code>application.metacatVersion</code>	The Metacat version number. It is set by the build engineer at build time. Usually, the value should never be changed. Default Value: X.X.X (where X.X.X is the current version of Metacat)	1.9.0
<code>application.metacatReleaseInfo</code>	Release information for display purposes. Typically the property is set during the release candidate cycle to let users know which candidate they are downloading.	Release Candidate 1
<code>application.deployDir</code>	The directory where Web applications are deployed. Usually, the value is a directory named "webapps" in the Tomcat installation directory.	/usr/local/tomcat/webapps

application.context	The name of the Metacat application directory in the deployment directory. This corresponds to the first part of the WAR file name (the part before .war). Most commonly, this is "knb", but it can be changed to other things.	knb
application.default-style	A custom Metacat Web skin usually associated with an organizational theme. If your organization has no custom skin, leave the value as "default".	default
application.knbSiteURL	The main KNB website. Default Value: http://knb.ecoinformatics.org	http://knb.ecoinformatics.org
application.datafilepath	The directory in which to store data files. The directory should be outside the Metacat installation directories so data files will not be lost when Metacat is upgraded. The data file directory must be writable by the user that starts Tomcat (and thus Metacat). Default Value: /var/metacat/data	/var/metacat/data
application.inlinedatafilepath	The directory where inline data files will be stored. Inline data files are created from data that is embedded in EML metadata. The directory should be outside the Metacat installation directories so data files will not be lost when Metacat is upgraded. For clarity of data, this should probably not be the same as application.datafilepath. The data file directory must be writable by the user that starts Tomcat (and thus Metacat). Default Value: /var/metacat/inline-data	/var/metacat/inline-data
application.documentfilepath	The directory where metadata files will be stored. The directory should be outside the Metacat installation directories so document files will not be lost when Metacat is upgraded. For clarity of organization, this should probably not be the same as application.datafilepath or application.inlinedatafilepath. The data file directory must be writable by the user that starts Tomcat (and thus Metacat). Default Value: /var/metacat/documents	/var/metacat/documents

<code>application.tempDir</code>	<p>The directory where the Metacat data registry stores temporary files. The directory should not be the same as <code>application.datafilepath</code> or <code>application.inlinedatafilepath</code> (or any other persistent file path) because all files in this may be purged programmatically. The temporary file directory must be writable by the user that starts Apache.</p> <p>Default Value: <code>/var/metacat/temporary</code></p>	<code>/var/metacat/temporary</code>
----------------------------------	---	-------------------------------------

Table 11.2: Metacat Application properties. Highlighted properties can only be set manually in the `metacat.properties` file.

Database Properties

Metacat's database properties are described in Table 10.3. Properties that can only be edited manually in the `metacat.properties` file are highlighted. All others are managed with the properties configuration utility.

Metacat Database Properties

Property	Description	Example
<code>database.connectionURI</code>	<p>The JDBC connection URI for the main database instance of Metacat. The URI is formatted like:</p> <p><code>jdbc:<database_type>:thin@<your_server_name>:1521:<metacat_database_name></code></p> <p>NOTE: You must create an empty database prior to initial Metacat configuration.</p> <p>Default Value: <code>jdbc:postgresql://localhost/metacat</code></p>	<code>jdbc:postgresql://yourserver.yourdomain.edu/metacat</code>
<code>database.user</code>	The user for the main database instance of Metacat. The user must have already been created on the database.	<code>metacat-user</code>
<code>database.password</code>	The password of the user for the main database instance of Metacat. The password must have already been created for the user.	<code>securepassword4843</code>
<code>database.type</code>	The type of database you are running. Currently, there are two supported types, Oracle and Postgres.	<code>postgres</code>
<code>database.driver</code>	The JDBC driver to be used to access the main database instance of Metacat. There is one driver associated with each type of database.	<code>org.postgresql.Driver</code>

<code>database.adapter</code>	<p>The adapter class that allows Metacat to access your database type. There is one adapter associated with each type of database.</p>	<code>edu.ucsb.nceas.dbadapter.PostgresqlAdapter</code>
<code>database.scriptsuffix.<database_type></code>	<p>The script suffix tells the system which database scripts to run (postgres or oracle) when installing or updating database schema.</p> <p>Default Values: <code>database.scriptsuffix.postgres=postgres.sql</code> <code>database.scriptsuffix.oracle=oracle.sql</code></p>	<code>postgres.sql</code>
<code>database.upgradeVersion.<database_version></code>	<p>Which database scripts to run when updating database schema. There is a <code>database.upgradeVersion</code> entry for every Metacat database schema version. Each schema version corresponds to an application version.</p> <p>Default Values:</p> <p><code>database.upgradeVersion.0.0.0=xmltables,loadtdschema</code></p> <p><code>database.upgradeVersion.1.2.0=upgrade-db-to-1.2</code></p> <p><code>database.upgradeVersion.1.3.0=upgrade-db-to-1.3</code></p> <p><code>database.upgradeVersion.1.4.0=upgrade-db-to-1.4</code></p> <p><code>database.upgradeVersion.1.5.0=upgrade-db-to-1.5</code></p> <p><code>database.upgradeVersion.1.6.0=upgrade-db-to-1.6</code></p> <p><code>database.upgradeVersion.1.7.0=upgrade-db-to-1.7</code></p> <p><code>database.upgradeVersion.1.8.0=upgrade-db-to-1.8</code></p> <p><code>database.upgradeVersion.1.9.0=upgrade-db-to-1.9</code></p>	<code>upgrade-db-to-1.2</code>
<code>database.initialConnections</code>	<p>The number of initial connection that Metacat creates to the database.</p> <p>Default Value: 5</p>	5

<code>database.incrementConnections</code>	The number of connections Metacat creates when it requires more connections. Default Value: 5	5
<code>database.maximumConnections</code>	The maximum number of database connections Metacat can make. Default Value: 200	25
<code>database.maximumConnectionAge</code>	The maximum time in milliseconds that a database connection can live. Default Value: 120000	120000
<code>database.maximumConnectionTime</code>	The maximum time in milliseconds that a database connection can accumulate in actual connection time. Default Value: 60000	60000
<code>database.maximumUsageNumber</code>	The maximum number of times a single connection can be used. Default Value: 100	100
<code>database.numberOfIndexingThreads</code>	The number of threads available for indexing. Default Value: 5	5
<code>database.indexingTimerTaskTime</code>	The time in milliseconds between indexing. Default Value: 604800000	604800000
<code>database.indexingInitialDelay</code>	The delay in milliseconds before first indexing is executed. Default Value: 3600000	3600000
<code>database.maximumIndexDelay</code>	The time in milliseconds that an indexing thread will wait when it can't get a doc id before retrying the indexing. Default Value: 5000	5000
<code>database.runDBConnectionRecycleThread</code>	Determines whether the database connection pool should run a thread to recycle connections. Possible values are "on" and "off" Default Value: off	off

<code>database.cycleTimeOfDBC onnection</code>	The time in milliseconds between connection recycling runs. Default Value: 30000	30000
<code>database.queryIgnoredpa rams</code>	Parameters to ignore in a structured XML query. Default Value: enableediting,foo	enableediting
<code>database.usexmlindex</code>	Determines whether to use XML indexes when finding documents. Possible values are true and false. Default Value: true	true
<code>database.appResultSetSi ze</code>	Determines the number of results that can be returned to an application from a query. Default Value: 7000	7000
<code>database.webResultSetSi ze</code>	Determines the number of results that can be returned to a Web browser from a query. Default Value: 7000	7000
<code>database.xmlReturnfield Count</code>	If the query results of a query are returned more times than this value, then those results will be inserted into the <code>xml_queryresult</code> table in the database. For example, if you want results for a query to be stored in <code>xml_queryresult</code> only when it has been requested 50 times, set this value to 50. Default Value: 0	0
<code>database.queryresultStr ingLength</code>	The max size of the query result string in the <code>queryresult</code> table. This should be set to some number less than 4000 if an Oracle database is being used. Default Value: 500000	500000
<code>database.queryresultCac heSize</code>	The number of query results that will be cached. Default Value: 500	500
<code>database.queryCacheOn</code>	Determines whether query caching is turned on. Possible values are "on" and "off" Default Value: on	on

Table 11.3: Metacat Database properties. Highlighted properties can only be set manually in the `metacat.properties` file.

Authorization and Authentication Properties

Metacat's authorization and authentication properties are described in Table 10.4. Properties that can *only* be edited manually in the `metacat.properties` file are highlighted. All others are managed with the properties configuration utility.

Authorization and Authentication Properties

Property	Description	Example
<code>auth.class</code>	<p>The class used for user authentication. Currently, only the <code>AuthLdap</code> class is included in the Metacat distribution.</p> <p>Note: If you implement another authentication strategy by implementing a Java class that extends the <code>AuthInterface</code> interface and rebuilding Metacat, change this property to the fully qualified class name of your custom authentication mechanism.</p> <p>Default Value: <code>edu.ucsb.nceas.metacat.AuthLdap</code></p>	<code>edu.ucsb.nceas.metacat.AuthLdap</code>
<code>auth.timeoutMinutes</code>	<p>The number of minutes that a user will stay logged in to Metacat without any activity.</p> <p>Default Value: 180</p>	180
<code>auth.administrators</code>	<p>A colon separated list of LDAP users or groups that have administrative Metacat privileges. At least one user or group must be entered when Metacat is first installed and configured. All accounts must exist in LDAP in order to continue with the configuration.</p>	<p>Example: <code>uid=youruser,o=NCEAS,dc=ecoinformatics,dc=org</code> <code>cn=yourgroup,o=NCEAS,dc=ecoinformatics,dc=org</code></p>
<code>auth.url</code>	<p>The URL of the server that Metacat should use for authentication.</p> <p>Default Value: <code>ldap://ldap.ecoinformatics.org:389/</code></p>	<code>ldap://ldap.ecoinformatics.org:389/</code>
<code>auth.surl</code>	<p>The URL of the server that Metacat should use for secure authentication.</p> <p>Default Value: <code>ldap://ldap.ecoinformatics.org:389/</code></p>	<code>ldap://ldap.ecoinformatics.org:389/</code>

auth.base	The base part of the distinguished name that Metacat uses for authentication. Default Value: dc=ecoinformatics,dc=org	dc=ecoinformatics,dc=org
auth.allowedSubmitters	A colon delimited list of users who should be allowed to submit documents to Metacat. If no value is specified, all users will be allowed to submit documents. Default Value: (none)	uid=youruser,o=NCEAS,dc=ecoinformatics,dc=org
auth.deniedSubmitters	A colon delimited list of users who should not be allowed to submit documents. If no value is specified, all users will be allowed to submit documents. Default Value: (none)	uid=youruser,o=NCEAS,dc=ecoinformatics,dc=org
ldap.connectTimeLimit	The time in milliseconds allowed for LDAP server connections. Default Value: 5000	5000
ldap.searchTimeLimit	The time in milliseconds allowed for LDAP server searches. Default Value: 30000	3000
ldap.searchCountLimit	The number of return entries allowed for LDAP server searches. Default Value: 30000	30000
ldap.referral	The type of LDAP referrals to use. Possible values are "follow", "throw" or "none". Refer to LDAP documentation for further information. Default Value: follow	follow
ldap.onlySecureConnection	Determines whether to use only a secure LDAP server. Acceptable values are "true" and "false". Default Value: false	false
ldap.onlySecureReferralsConnection	Determines whether to only use a secure referral server. Acceptable values are "true" and "false". Default Value: false	false

Table 11.4: Authentication properties. Highlighted properties can only be set manually in the metacat.properties file.

XML/EML Properties

Metacat's XML/EML properties are described in Table 10.5. Properties that can *only* be edited manually in the `metacat.properties` file are highlighted.

XML/EML Properties

Property	Description	Example
<code>xml.saxparser</code>	The SAX parser used to parse XML documents. Metacat requires a SAX2-compatible XML parser. Default Value: <code>org.apache.xerces.parsers.SAXParser</code>	<code>org.apache.xerces.parsers.SAXParser</code>
<code>xml.eml2_0_0namespace</code>	The namespace of EML 2.0.0 documents. Default Value: <code>eml://ecoinformatics.org/eml-2.0.0</code>	<code>eml://ecoinformatics.org/eml-2.0.0</code>
<code>xml.eml2_0_1namespace</code>	The namespace of EML 2.0.1 documents. Default Value: <code>eml://ecoinformatics.org/eml-2.0.1</code>	<code>eml://ecoinformatics.org/eml-2.0.1</code>
<code>xml.eml2_1_0namespace</code>	The namespace of EML 2.1.0 documents. Default Value: <code>eml://ecoinformatics.org/eml-2.1.0</code>	<code>eml://ecoinformatics.org/eml-2.1.0</code>
<code>xml.packagedoctype</code>	The doctype of a package file. The system will only recognize documents of this type as package files. See: package documentation. Default Value: <code>://ecoinformatics.org//eml-dataset-2.0.0beta6//EN, -//ecoinformatics.org//eml-dataset-2.0.0beta4//EN</code>	<code>://ecoinformatics.org//eml-dataset-2.0.0beta6//EN, -//ecoinformatics.org//eml-dataset-2.0.0beta4//EN</code>
<code>xml.accessdoctype</code>	The doctype of an access control list (ACL) file. The system will only recognize documents of this type as access files. See: access control documentation. Default Value: <code>://ecoinformatics.org//eml-access-2.0.0beta6//EN, -//ecoinformatics.org//eml-access-2.0.0beta4//EN</code>	<code>://ecoinformatics.org//eml-access-2.0.0beta6//EN, -//ecoinformatics.org//eml-access-2.0.0beta4//EN</code>

Table 11.5: XML/EML Properties. Highlighted properties can only be set manually in the `metacat.properties` file.