

ANU Data Commons

System Administrator's Manual



This work is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Australia License](http://creativecommons.org/licenses/by-nc-sa/3.0/au/deed.en).

Contents

[Overview 4](#_Toc348002452)

[License 4](#_Toc348002453)

[Acknowledgement 4](#_Toc348002454)

[Components 4](#_Toc348002455)

[Architecture 5](#_Toc348002456)

[Prerequisites 5](#_Toc348002457)

[PostgreSQL 6](#_Toc348002458)

[Java Runtime Environment 6](#_Toc348002459)

[Tomcat 6](#_Toc348002460)

[ClamAV 6](#_Toc348002461)

[Python 2.7 6](#_Toc348002462)

[Fido 6](#_Toc348002463)

[Configuration 7](#_Toc348002464)

[Configuring Tomcat 7](#_Toc348002465)

[Set up access to the Manager web application 7](#_Toc348002466)

[Deploying WAR files greater than 50 MB 7](#_Toc348002467)

[Configuring Maven 7](#_Toc348002468)

[Configuring Apache Solr and Fedora GSearch 8](#_Toc348002469)

[Configuring Fedora Commons 9](#_Toc348002470)

[Configuring Data Commons 11](#_Toc348002471)

[Setup Database 13](#_Toc348002472)

[Setup String 13](#_Toc348002473)

[Configuring Fido 14](#_Toc348002474)

[Configuring ClamAV 14](#_Toc348002475)

[Building 14](#_Toc348002476)

[Dependencies 14](#_Toc348002477)

[Fits Library 14](#_Toc348002478)

[BagIt Library 15](#_Toc348002479)

[Build Process 15](#_Toc348002480)

[Clone the source repository 15](#_Toc348002481)

[Execute Maven Build 15](#_Toc348002482)

[Deployment 15](#_Toc348002483)

[Deployment using Tomcat Manager 15](#_Toc348002484)

[Deployment using Maven Tomcat Plugin 16](#_Toc348002485)

[Troubleshooting 16](#_Toc348002486)

[SSL Exceptions 16](#_Toc348002487)

# Overview

This document lists and explains the steps required to deploy and maintain an instance of ANU Data Commons software.

# License

Use of ANU Data Commons is governed by the GNU GPL3 license.

# Acknowledgement

This project is supported by the Australian National Data Service (ANDS). ANDS is supported by the Australian Government through the National Collaborative Research Infrastructure Strategy Program and the Education Investment Fund (EIF) Super Science Initiative.

# Components

The ANU Data Commons has three main components:

1. **The Data Commons Web Application**
This is a the primary component of the project that provides the means to store and preserve Research Data and Metadata and make it accessible by making it searchable and by publishing it to other repositories. The system also implements a security framework allowing access only to those who should have access to the records or have gone through the required workflow to request and subsequently gain access to the records.
2. **The Web Services Web Application**
This component enables Machine to Machine (M2M) communication between an external system with the Data Commons Web Application. This component translates requests sent by client machines into requests the Data Commons can understand and process. It then receives the responses from Data Commons, and processes and packages them into a format the client machine or service understands.
3. **The DcClient Desktop Application**
This component allows common Data Commons tasks to be performed without a web interface. This component resides and executes on a client machine and interacts directly with Data Commons to create/update collection records and add/update/delete files associated with them. The absence of a web interface enables automating tasks facilitating bulk ingests without requiring human supervision.

# Architecture

Tomcat

ANU Data Commons

Web Service

Fedora Commons

PostgreSQL

Area specific web service translator

Fedora GSearch

Solr

# Prerequisites

Components 1 and 2 are Java applications that require a server (or virtual machine) capable of running an operating system that supports Oracle Java. Refer to http://www.oracle.com/technetwork/java/javase/downloads/index.html for a list of operating systems capable of hosting a Java Runtime Environment.

We are running Red Hat Enterprise Linux Server 5.8 running on a virtual machine with 2 GB of RAM.

1. PostgreSQL. Available from http://www.postgresql.org/download/
2. Java Runtime Environment
	1. Tomcat
		1. Fedora Commons Repository. Available from http://fedora-commons.org/
		2. Apache Solr. Available from http://lucene.apache.org/solr/
3. ClamAV. Available from http://www.clamav.net/lang/en/
4. Python 2.7. Available from http://www.python.org/
	1. Fido. Available from https://github.com/openplanets/fido

If using Linux as the operating system some of the aforementioned programs may be available in your distribution’s repository. Check your package manager for more details.

## PostgreSQL

Fedora Commons uses an instance of PostgreSQL to store digital objects. ANU Data Commons interfaces with Fedora Commons to operate on collection data while storing operational, security and other application related data in another database in the PostgreSQL instance. Creation of application specific databases within this instance is discussed in subsequent sections.

## Java Runtime Environment

Installation of the Java Runtime Environment (JRE) is specific to the platform it is being installed on. Refer to your operating system’s user manual for details. The Java Development Kit (JDK) is required if you intend to perform remote debugging.

### Tomcat

Tomcat is an application server that can be used for hosting web applications. Tomcat has a built-in web server that can be used to serve HTTP requests without the need for a dedicated web server such as Apache HTTP Server.

#### Fedora Commons Repository

Fedora Commons Repository is an open source program for use in long-term preservation of digital collections. Follow the instructions provided at https://wiki.duraspace.org/display/FEDORA35/Installation+and+Configuration to install and configure the software. The Data Commons component interacts with the Fedora Repository through HTTP requests that conform to REST API specifications.

#### Apache Solr

Apache Solr is a search platform used to index and search content stored in the Fedora Commons Repository. The search functionality provided by ANUDC relies heavily on this component.

## ClamAV

ClamAV is an antivirus for Linux that runs a network service for scanning files. The network service accepts file streams through a TCP connection and returns the Scan Status of the file as a String describing if the file contains a virus.

## Python 2.7

Python allows execution of Python scripts.

### Fido

Format Identification for Digital Objects (FIDO) is a Python command-line tool to identify the file formats of digital objects. It is designed for simple integration into automated workflows. Data Commons uses this script to identify the format of a file. Recognised formats include those that be successfully identified by Droid, a program by The National Archives (http://www.nationalarchives.gov.uk/PRONOM/Default.aspx) .

The script can be installed in any location accessible by the Java Virtual Machine hosting the Tomcat instance. The location is used in the Data Commons configuration files.

It is recommended the signatures in Fido be updated on a regular basis by executing the script update\_signatures.py . This allows Fido to recognise new file formats.

Fido specifically requires version 2.7 of Python and is deemed incompatible with earlier and later versions.

# Configuration

## Configuring Tomcat

### Set up access to the Manager web application

Accessing the Manager application through the web interface requires a Tomcat user to be setup with the manager-gui role. Accessing the same application through a scripted interface, such as through the Maven Tomcat Plugin requires a Tomcat user to be setup with the manager-script role. Refer to <http://tomcat.apache.org/tomcat-7.0-doc/manager-howto.html> for details on how to configure the Manager application.

### Deploying WAR files greater than 50 MB

Tomcat’s default configuration doesn’t allow files greater than 50 MB to be deployed using the Tomcat Manager application. Attempting to do so will result in the following error message:

Exception java.lang.IllegalStateException:

org.apache.tomcat.util.http.fileupload.FileUploadBase$SizeLimitExceededException:

the request was rejected because its size (XXX) exceeds the configured maximum (52428800)

To enable large WAR files to be deployed through the Manager application, open the file webapps/manager/WEB-INF/web.xml and search the following text:

<multipart-config>

<!-- 50MB max -->

<max-file-size>52428800</max-file-size>

<max-request-size>52428800</max-request-size>

<file-size-threshold>0</file-size-threshold>

</multipart-config>

The value 52428800 represents the maximum size in bytes of a WAR file that the Manager application will accept. Change this value to a higher value to allow larger files to be uploaded. As the WAR files can be quite large a fast connection between the client and the server hosting the tomcat instance is highly recommended.

Save the file after making the changes and restart the tomcat instance for the changes to take effect.

## Configuring Maven

For Maven to deploy applications to Tomcat, you’ll need to create one or more profiles that include information such as the URL where the Tomcat instance is hosted along with the credentials to use to access its manager application. Refer to the section Set up access to the Manager web application to setup the users whose credentials you’d like to use to deploy applications through Maven.

To create a profile for a tomcat instance, open the settings.xml file. Refer to <http://maven.apache.org/settings.html> to find the location of settings.xml . In the file, add the following XML:

<servers>

 ...

 <server>

 <id>INSTANCE\_ID</id>

 <username>USERNAME</username>

 <password>PASSWORD</password>

 </server>

 ...

</servers>

<profiles>

 ...

 <profile>

 <id>PROFILE\_ID</id>

 <properties>

 <maven.tomcat.url>http://HOSTNAME:PORT/manager/text</maven.tomcat.url>

 <maven.tomcat.server>INSTANCE\_ID</maven.tomcat.server>

 </properties>

 ...

</profile>

|  |  |
| --- | --- |
| Parameter | Value |
| INSTANCE\_ID | An arbitrary ID assigned to the username and password to be used for sending requests to the Manager application. |
| USERNAME | The username to which manager-script role is assigned. |
| PASSWORD | Password for the username above. |
| PROFILE\_ID | An arbitrary ID assigned to the tomcat instance to which a Web application will deploy. Generally, you’d have one profile for the development tomcat instance, one for testing and one for production. |
| HOSTNAME | The fully qualified hostname where the tomcat instance is located. For example, datacommons.com . |
| PORT | The port on which the tomcat instance is listening on. For example, 8080 . |
| INSTANCE\_ID | The ID assigned to the combination of username and password to be used for deploying applications through the Manager application. |

## Configuring Apache Solr and Fedora GSearch

Replace the file:

$CATALINA\_BASE/webapps/fedoragsearch/WEB-INF/classes/fgsconfigFinal/index/FgsIndex/foxmlToSolrCustom.xslt

with the one provided with the project in DataCommons/extras/solr/foxmlToSolrCustom.xslt .

Replace the file schema.xml in $SOLR\_HOME/conf with the one provided with the project in DataCommons/extras/solr/solr/schema.xml .

## Configuring Fedora Commons

The following objects will need to be loaded into the repository:

These are located in DataCommons/extras/foxml/ directory and can be ingested into Fedora Commons by executing for the following in $FEDORA\_HOME/client directory:

fedora-ingest d DIRECTORY info:fedora/fedora-system:FOXML-1.1 REPOSITORY\_HOST:REPOSITORY\_PORT REPOSITORY\_USERNAME REPOSITORY\_PASSWORD PROTOCOL

|  |  |
| --- | --- |
| Parameter | Value |
| DIRECTORY | is the path to the directory where the FOXML files are located. |
| REPOSITORY\_HOST and REPOSITORY\_PORT | are the hostname and port of the Fedora Commons web application. |
| REPOSITORY\_USERNAME and REPOSITORY\_PASSWORD | are the credentials to use to access the repository. |
| PROTOCOL | is the protocol to be used to connect to the repository. This will either be 'http' or 'https' depending on the setup of the Fedora Commons instance. |

Once the objects have been ingested, log into the administrative interface at /fedora/admin URL of the tomcat instance hosting the Fedora Repository. Click Search.



Then enter "tmplt:\*" in the search text box and click search.



That will give you a list of template objects in the repository. Click on each one of them, then click on XML\_TEMPLATE datastream.



That will bring up the XML\_TEMPLATE datastream dialog box. Click on the Edit Content button copy the text from the corresponding XML file in DataCommons/extras/xml and paste it in the text area. Then click Save Changes followed by Close.



Repeat these steps for each XML template.

## Configuring Data Commons

The Data Commons application accesses its configuration information from a number of properties files.

If instance of Tomcat hosting Data Commons is running on a Windows machine (or VM) the root directory of properties files will be C:\AnuDc . For Linux, it will be /etc/anudc .

The following tree structure along with the properties files must be created in the aforementioned properties root directory.

C:\AnuDc or /etc/anudc

| log4j.properties

|

+---datacommons

| datacommons.properties

| doi.properties

| tokens.properties

|

+---logs

|

+---ws-digitalhumanities

| constants.properties

| genericws.properties

| wslookup.properties

|

+---ws-gateway

| redir.properties

|

+---ws-geoscience

| constants.properties

| genericws.properties

| wslookup.properties

|

\---ws-phenomics

 constants.properties

 genericws.properties

 wslookup.properties

Each of these properties files contain comments describing the keys and values that should be used. Here is an overview of the roles these files play in configuring ANUDC.

|  |  |
| --- | --- |
| Properties File | Description |
| log4j.properties | This file specified logging configuration for the Log4j logging framework used by all modules in the anudc project. The ANU Data Commons logs vital information that can be very useful when investigating issues in the system. The location of these log messages is determined by the configuration specified in log4j.properties. Note that this configuration file only directs the logging framework used by the web applications that Tomcat hosts, not the logging performed by Tomcat itself, which is performed by the JULI framework. Refer to http://tomcat.apache.org/tomcat-7.0-doc/logging.html for more information about the way Tomcat performs logging. Detailed information about logging configuration can be found at http://logging.apache.org/log4j/1.2/manual.html . |
| datacommons/datacommons.properties | Refer to extras/properties/datacommons/doi.properties file that provides a template for this file. |
| datacommons/doi.properties | This file contains configuration information for the Digital Object Identifier (DOI) module. Refer to extras/properties/datacommons/doi.properties file that provides a template for this file. |
| datacommons/tokens.properties | This file contains configuration information related to the token-based authentication mechanism in Data Commons to enable client machines and services to communicate with the Data Commons Web Service without the need for user credentials in requests. |
| ws-gateway/redir.properties | This file contains redirection configuration used by the Data Commons Web Service Gateway layer to forward requests onto individual area-specific web services that in turn interact with the Data Commons Web Service. |
| ws-[area]/constants.properties | This file contains constants that get added to the XML request sent to the web service. Each area has a unique set of constants. This does away with the need to include area specific fields to be included in every XML request submitted to the Data Commons. |
| ws-[area]/genericws.properties | This file contains URL information about the generic web service to which requests get forwarded to. |
| ws-[area]/wslookup.properties | This file contains versioning information related to XML requests that come through. |

### Setup Database

In addition to creating a database for the Fedora Commons Repository instance as explained in the Fedora Commons Installation and Configuration document at [https://wiki.duraspace.org/display/FEDORA36/Installation+and+Configuration](https://wiki.duraspace.org/display/FEDORA36/Installation%2Band%2BConfiguration) , ANU Data Commons uses a relational database store application data such as permissions, collection requests, dropboxes, user information etc. Perform the following steps to create a database for use by the application:

Create a database in a PostgreSQL instance by executing the following commands:

psql -U postgres -f 1\_create\_database.sql

psql -U dcuser -f 2\_create\_tables.sql -d datacommonsdb

psql -U dcuser -f 3\_add\_data.sql -d datacommonsdb

Then execute each of the SQL files in the format YYYYMMDD\_NAME.sql in order:

psql –U dcuser –f YYYYMMDD\_NAME.sql –d datacommonsdb

To establish a connection between ANUDC and the database created, create a copy of the following file DataCommons/src/main/resources/META-INF/persistence-template.xml and save it in the same directory as persistence.xml .

Modify the following properties:

|  |  |
| --- | --- |
| Property | Value |
| hibernate.connection.driver\_class | Change to an appropriate value if using a database other than PostgreSQL |
| hibernate.connection.url | The URL of the database. For example, jdbc:postgresql://hostname:1234/dbname |
| hibernate.connection.user |  Username to use to connect to the database. For example, dcuser |
| hibernate.connection.password | Password to use to connect to the database. |
| dialect | Change to an appropriate value if using a database server other than PostgreSQL |

### Setup String

ANUDC uses a number of Spring Framework components that require configuration. These files are located in DataCommons/src/main/webapp/WEB-INF/ . Refer to Spring Framework documentation at <http://www.springsource.org/documentation> .

## Configuring Fido

For ANUDC to execute Fido on uploaded files the Fido scripts should be saved on a directory on the same server running the Tomcat instance. A file fido.properties should be created in the user's home directory with the following contents:

# Fido Properties

python.exe=LOCATION OF PYTHON EXECUTABLE

fido.py=LOCATION OF FIDO SCRIPT

|  |  |
| --- | --- |
| Property | Description |
| python.exe | Location of the python executable including the executable file. For example, C:\\Program Files\\Python\\Python 2.7\\python2.7.exe in Microsoft Windows or /usr/bin/python2.7 in Linux |
| fido.py | Location of the fido.py script including the filename. For example, C:\\Scripts\\Fido\\fido.py in Windows or ~/fido/fido.py in Linux |

Note the double backslashes in the examples above. Being a properties file, special characters like \, = and : must be escaped using the backslash character. Forward slashes do not need to be escaped.

## Configuring ClamAV

On Linux, edit the file /var/log/clamav/clamd.log and edit the following properties:

|  |  |  |
| --- | --- | --- |
| Property | New Value | Description |
| TCPAddr | 127.0.0.1 | This will make the ClamAV server listen to requests coming in only from the same server. |
| StreamMaxLength | 4000M | Allows ClamAV to scan files upto 4GB in size, which is the maximum file size of a file sent to the ClamAV server for scanning. |

Refer to section "Setting up auto-updating" in <http://www.clamav.net/doc/latest/clamdoc.pdf> to enable automatic background virus definitions updates.

# Building

## Dependencies

As ANU DataCommons is a Maven project, most of its dependencies will automatically be pulled from a Maven repository. Some dependencies, however, are not hosted in the Maven central repository must be manually installed in a local repository for the project to build. Following are a list of such dependencies that require manual installation:

### Fits Library

To install the Fits Library, download the fits\_src.jar file from <http://heasarc.gsfc.nasa.gov/docs/heasarc/fits/java/v1.0/v1.10.0/> and run the following command:

mvn install:install-file -DgroupId=nom.tam.fits -DartifactId=fits -Dversion=1.10 -Dfile="fits.jar" -Dpackaging=jar -DgeneratePOM=true -Dsources="fits\_src.jar"

### BagIt Library

A modified version of the BagIt Library has been provided along with the anudc project. Modifications include bug fixes and performance enhancements. To install this library in your local repository run the following command in the root of the BagIt Project directory.

mvn clean install -DskipTests

# Build Process

ANU Data Commons uses Maven as its build tool. To build the projects from source perform the following steps:

## Clone the source repository

Clone the GitHub repository where ANU Data Commons’s source code is hosted - <https://github.com/anu-doi/anudc> .

## Execute Maven Build

Execute the following command in the directory where the ANU Data Commons repository has been cloned to compile and build the project into JAR and WAR files:

mvn clean package

If any of the tests fail, run:

mvn clean package –DskipTests

# Deployment

Once the project has been built, the generated WAR files can be deployed to Tomcat by using the Tomcat manager application or using Maven itself.

## Deployment using Tomcat Manager

Assuming the Tomcat instance is hosted at <http://localhost:8080/>, Open Tomcat Manager application at <http://localhost:8080/manager> . Scroll down to the section titled ‘Deploy’. Click on the browse button, select the WAR file for the application you’d like to deploy in the file browser dialog box and select OK. Then Click on the Deploy button to deploy the application to Tomcat.



## Deployment using Maven Tomcat Plugin

The applications can be deployed to Tomcat by executing the following in the project directory:

mvn tomcat:deploy-only –p PROFILE\_ID

where PROFILE\_ID is the profile ID assigned to a tomcat instance. Refer to section Configuring Maven.

This command only deploys the web applications in the anudc project without executing the package phase. The modules must already be packaged for them to be deployed. If you’d like to package the application and deploy using a single command, execute the following:

mvn tomcat:deploy –p PROFILE\_ID

Refer to <http://mojo.codehaus.org/tomcat-maven-plugin/deployment.html> for more information.

If the application is already deployed to Tomcat and you need to redeploy the application after making code or resource file changes, the previously deployed WARs must be undeployed by executing the following command in the anudc root directory:

mvn tomcat:undeploy

# Troubleshooting

## SSL Exceptions

If the tomcat instance is configured for SSL connections from clients, it is vital that the private key and public certificate are correctly configured. Refer to the section SSL Configuration HOW-TO at <http://tomcat.apache.org/tomcat-7.0-doc/ssl-howto.html> .

If using a self-signed certificate on the server, all clients connecting to it must have the certificate in its trusted certs store. This includes applications within Tomcat that act as clients to other applications in the same Tomcat instance.