

NYU CAT (CT) Scan Web Application Deployment Guide

Version: 1.1

5-Jun-06

Changelog

<i>Version</i>	<i>Comments</i>	<i>Revision Date</i>	<i>Author</i>
1.0	Initial Revision	22-Apr-06	C. Mattmann
1.1	Fixed typos, added information about XDoclet deps, add revision date col. to change-log, update revision date.	5-Jun-06	C. Mattmann

1. Pre-requisites

The NYU CAT (CT) Scan Web Application is a Java-based web application. To install the application, first you'll need to install Java. The application is optimized for Java 1.4.2, and will not work with Java 1.5. If you are installing from source and want to build the application, you'll also need Apache's Maven software development toolkit, version 1.x (where *x* corresponds to minor version number from the 1.x series of the Maven toolkit), available from <http://maven.apache.org>. The application has been tested with Maven 1.0.2, but should also work with 1.0.1, and with 1.1. If you have a binary version of the application, then you can skip to Section 2. To run the application, you'll need a Servlet API compatible web container. It is recommended that Apache Tomcat 5.5.x (where *x* corresponds to one of the minor version numbers for the Tomcat 5.5 series) be downloaded and installed from <http://tomcat.apache.org>. The application has been tested with Tomcat 5.5.16, but should work with any Tomcat in the 5.5.x series. The application also needs to communicate with a MySQL database. The application has been tested with MySQL 4.x databases, but should also work with the 5.x series.

Here is a summary table of the pre-requisites:

	<i>Name</i>	<i>CT Scan component</i>	<i>URL</i>
1	Java Development Kit (1.4.2)	Runtime Environment	http://java.sun.com
2	Apache Maven (1.0.2)	Build-time Toolkit	http://maven.apache.org
3	Apache Tomcat (5.5.16)	Web Application Environment	http://tomcat.apache.org
4	MySQL Database (4.x)	CAT Scan Data	http://www.mysql.org

1.1 Building the CT Scan application from source

You'll need a source directory containing the code and maven build xml files from the project. For the 1.0 delivery, this file is a tar'ed up, g'zipped file called "nyu-ctscan-webapp-1.0.tar.gz". Any subsequent deliveries of software will follow this convention, nyu-ctscan-webapp-*x.x*, where *x.x* corresponds to the major and minor version number of the CTScan webapp software. First we'll need to unpack the application:

```
# tar -xvzf nyu-ctscan-webapp-1.0.tar.gz
# ls
nyu-ctscan-webapp-1.0      nyu-ctscan-webapp-1.0.tar.gz
```


EDRN INFORMATICS WORKING GROUP

5-Jun-06

```
[echo] Building WAR nyu-ctscan-webapp
[jar] Building jar: /Users/mattmann/src/nyu-ctscan-
webapp/trunk/target/nyu-ctscan-webapp.war
BUILD SUCCESSFUL
Total time: 23 seconds
Finished at: Sat Apr 22 16:12:05 PDT 2006
```

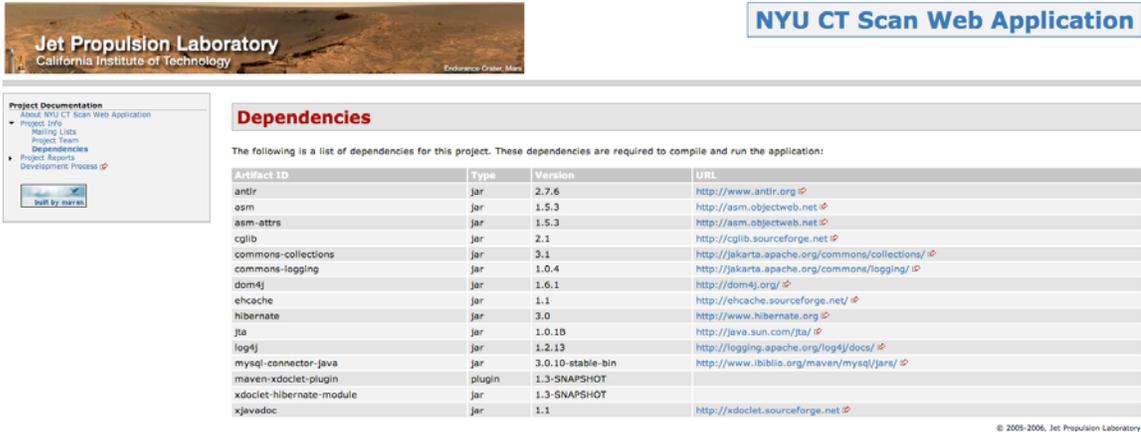
During the build process, maven will attempt to download needed jar file dependencies that the application relies on. If any of the jar files fail to download, you can also Google around on the internet to find the needed jar file. If maven is unable to find a jar file, it will print a message detailing the failed dependency, and suggest a website where you can go and find the jar file.

Maven creates a local jar file repository by default in `${HOME}/.maven/repository`. If Maven is unable to find a jar file while building the CTScan application, determine the jar file name from the Maven message, which will be of the form:

The build cannot continue because of the following unsatisfied dependency:

```
<artifactId>-<version>.jar
```

In order to satisfy this dependency, if you are able to find `<artifactId>-<version>.jar` (e.g., `commons-logging-1.0.0.jar`), then you need to place the jar file inside of `${HOME}/.maven/repository/<groupId>/jars/`. `groupId` corresponds to the group that publishes the jar file. You can find all the `groupIds` for the required jar files by examining the `siteDocs` which are included with both the binary and source distribution of the CTScan application. Navigate your browser to `file:/path/to/siteDocs/`, and then click on the *Project Info* link in the left hand navigation bar, and then click on *Dependencies*. You should see a page that looks like the following:



Jet Propulsion Laboratory
California Institute of Technology

NYU CT Scan Web Application

Dependencies

The following is a list of dependencies for this project. These dependencies are required to compile and run the application:

Artifact ID	Type	Version	URL
antlr	jar	2.7.6	http://www.antlr.org
asm	jar	1.5.3	http://asm.objectweb.net
asm-attrs	jar	1.5.3	http://asm.objectweb.net
cglib	jar	2.1	http://cglib.sourceforge.net
commons-collections	jar	3.1	http://jakarta.apache.org/commons/collections/
commons-logging	jar	1.0.4	http://jakarta.apache.org/commons/logging/
dom4j	jar	1.6.1	http://dom4j.org/
ehcache	jar	1.1	http://ehcache.sourceforge.net/
hibernate	jar	3.0	http://www.hibernate.org
java	jar	1.0.1B	http://java.sun.com/j2a/
log4j	jar	1.2.13	http://logging.apache.org/log4j/docs/
mysql-connector-java	jar	3.0.10-stable-bin	http://www.liblibia.org/maven/mysql/jars/
maven-xdoclet-plugin	plugin	1.3-SNAPSHOT	
xdoclet-hibernate-module	jar	1.3-SNAPSHOT	
xjavadoc	jar	1.1	http://xdoclet.sourceforge.net

© 2005-2006, Jet Propulsion Laboratory

One set of jar files that the CTScan web application relies on (build-time dependency) is the XDoclet (<http://xdoclet.sf.net>) library. Since CTScan relies on the development version (1.3), and not a released version (to take advantage of improvements in 1.3-dev that haven't been officially released, including an absolutely needed bug-fix in the Hibernate tag library), the Xdoclet dependencies are included in the delivery of the CTScan system. The dependency jar files are located in nyu-ctscan-xdoclet-deps-1.0.tar.gz. Untar the library into your \$HOME/.maven/repository folder and you're good to go.

Once the application successfully builds, you will have a target directory that contains the binary web application resource (WAR) file, along with the context.xml file (called "nyu-ctscan-webapp-tomcat-5.5.9.xml") needed to deploy the application. Now we can move onto the next section.

2. Configuring the CTScan Web Application

The two important files for deploying the CTScan application are:

1. The nyu-ctscan-webapp-tomcat-5.5.9.xml (also referred to as the *context.xml* file) file, where you can set dynamic properties for the application, and where you set the path on disk to the location of the WAR file
2. The web application resource (WAR) file for the CT Scan application, nyu-ctscan-webapp.war. This contains all the JSP pages and jar files containing the application logic for the system.

To deploy the CT Scan webapp, the first thing we need to do is make sure we have both the WAR file and the context.xml file:

EDRN INFORMATICS WORKING GROUP

5-Jun-06

```
# ls
nyu-ctscan-webapp-tomcat-5.5.9.xml  nyu-ctscan-
webapp.war
```

Okay, now we need to install the WAR file. It is recommended that the war file be installed in /usr/local/ctscan/webapp, however, it can be installed anywhere on disk.

```
# mkdir /usr/local/ctscan
# mkdir /usr/local/ctscan/webapp
# cp -R nyu-ctscan-webapp.war
/usr/local/ctscan/webapp
# ls /usr/local/ctscan/webapp
nyu-ctscan-webapp.war
```

After copying the war file, also copy the context.xml file to /usr/local/ctscan/webapp as well. Now, we have to edit the context.xml file, and make it point to the WAR file. We need to modify the Context tag inside the xml file:

```
...
<Context path="/ctscan" docBase="/path/to/nyu-ctscan-
webapp.war"
        debug="5" reloadable="true" crossContext="true">
....
```

Modify the docBase attribute to point to /usr/local/ctscan/webapp/nyu-ctscan-webapp.war. After you do that, you can also modify the dynamic properties of the CT Scan application, including the authentication mechanism to use (set to LDAP by default, but can be changed), you can see the searchDns for LDAP, along with the different LDAP provider URLs for authentication, and also for looking up information.

1. The property **gov.nasa.jpl.edrn.nyu.ctscan.ldap.searchdn** is a “|” delimited list of searchDns for the LDAP information servers that the application will query when obtaining user information. The first entry is entry [0] in the list, and the second [1], and so on (this information will become more important below)
2. The property **gov.nasa.jpl.edrn.nyu.ctscan.ldap.providerUrl** is a “|” delimited list of LDAP provider URLs that the CTScan system will use to obtain user information (such as email address and phone numbers) from. The first entry is entry [0] in the list, and the second [1], and so on. The providerUrl list, and the searchDn list specified in property 1 above must be of the same length, and must have entries that are corresponding to each other, for example, entry [0] in the searchDn list is the searchDn for provider URL entry [0] in the provider URL list.
3. The property **gov.nasa.jpl.edrn.nyu.ctscan.ldap.secureUrl** is a “|” delimited set of Boolean values specifying whether the providerUrl list entries are secure (SSL-

- based) LDAP URLs or not. Entry [0] in this list corresponds to entry [0] in the providerUrl list (property 2 above) and so on.
4. The property **gov.nasa.jpl.edrn.nyu.ctscan.ldap.auth.providerUrl** is a “[” delimited set of LDAP urls to authenticate against. The list is ordered by preference, and the first LDAP system that the user authenticates successfully against will be used.
 5. The property **gov.nasa.jpl.edrn.nyu.ctscan.lap.auth.secureUrl** is a “[” delimited set of Boolean values specifying whether or not the auth.providerUrl list entries (in property 4) are secure (SSL-based) LDAP URLs or not. Entry [0] in this list corresponds to entry [0] in the auth.providerUrl list (property 4).
 6. The property **ctscan.authenticator** controls the authentication mechanism used by the CT Scan application. The default value is set to *gov.nasa.jpl.edrn.nyu.ctscan.authentication.LDAPAuthenticator*, which is a java class that implements the *gov.nasa.jpl.edrn.nyu.ctscan.authentication.Authenticator* interface. This property can be set to any java class (that is on the classpath of the Servlet container that you deploy the webapp within) that implements the Authenticator interface. Specify the fully qualified classname in this property.

Once you're happy with the properties that you have set for the application, we can deploy the application to a Servlet container, Apache Tomcat.

3. Installing the Database

To install the CT Scan database, two SQL files are provided. The first file is called *CT Scan ER Model-Final.DDL*, and is the schema file for the database. The second file, *load_scan_data.sql* loads the default instance data into the database schema, once it has been loaded. Both files are located in `$CTSCAN_SRC_HOME/src/main/resources`.

First, execute the .DDL file inside of your SQL environment (if you are using MySQL, MySQL control center, or php_mySQLAdmin are recommended) to install the CTScan schema to the database. Then, once the schema is installed, execute the .sql file to load the default CT Scan follow up reasons, types, and so forth, to the database.

Before installing the instance data, edit the *load_scan_data.sql* file on the last line to add your administrator users to the database. By default, there are no administrators installed.

4. Deploying the CTScan Application

EDRN INFORMATICS WORKING GROUP

5-Jun-06

To deploy the application, we simply need to make the context.xml file available to Tomcat. We can do this in one of two ways:

1. We can copy the context.xml file to
\$TOMCAT_HOME/conf/Catalina/localhost/ctscan.xml
2. We can soft link the context.xml file wherever it's located on your system (this guide recommends /usr/local/ctscan/webapp, as stated in Section 2) to
\$TOMCAT_HOME/conf/Catalina/localhost/ctscan.xml

In this guide, we'll go with option 2. Issue the following command to soft link the context.xml file. Before soft-linking make sure that Tomcat is not running:

```
# echo $TOMCAT_HOME
/usr/local/tomcat
# $TOMCAT_HOME/bin/shutdown.sh
Using CATALINA_BASE:   /usr/local/tomcat
Using CATALINA_HOME:   /usr/local/tomcat
Using CATALINA_TMPDIR: /usr/local/tomcat/temp
Using JRE_HOME:        /usr/local/jdk
Created MBeanServer with ID: 2462b3:10ac2d47174:-
8000:cloud.local:1
# ln -fs /usr/local/ctscan/webapp/nyu-ctscan-webapp-
tomcat-5.5.9.xml
$TOMCAT_HOME/conf/Catalina/localhost/ctscan.xml
# ls -al $TOMCAT_HOME/conf/Catalina/localhost
/usr/local/tomcat/conf/Catalina/localhost/
total 24
drwxr-xr-x  5 mattmann  mattmann  170 Apr 22 11:24 ./
drwxr-xr-x  3 mattmann  mattmann  102 Mar  4 17:23 ../
lrwxr-xr-x  1 root      mattmann   85 Apr 22 11:24
ctscan.xml@ -> /usr/local/ctscan/webapp/nyu-ctscan-
webapp-tomcat-5.5.9.xml
-rw-----  1 mattmann  mattmann  297 Mar  4 17:25 host-
manager.xml
-rw-----  1 mattmann  mattmann  452 Mar  4 17:25
manager.xml
```



Now that the context.xml file has been linked correctly we can restart tomcat (by issuing a `$TOMCAT_HOME/bin/startup.sh` command). Once tomcat is restarted, the application should be running at:

http://<your_tomcat_server>:<your_tomcat_port>/ctscan/

(so if you installed Tomcat to run on gcr, on port 8443, you would visit:

<http://gcr.med.nyu.edu:8443/ctscan/>)

The application is now successfully deployed!

5. Uninstalling the CTScan Application

To uninstall the CT Scan application, we need to make sure that Tomcat isn't running. Once we make sure that it isn't running, issue the following commands to remove the application:

```
# rm -rf $TOMCAT_HOME/work/Catalina/localhost/ctscan/  
# rm -rf $TOMCAT_HOME/webapps/ctscan/  
# rm $TOMCAT_HOME/conf/Catalina/localhost/ctscan.xml  
# rm -rf /usr/local/ctscan/
```

This removes the unpacked war file from Tomcat's working directory, and webapps directory, and also removes the soft link to the context.xml file, and then removes the entire deployment directory.

To remove the instance data from the database, a sql file called *wipe_scan_data.sql* is provided. Run that SQL script in your database environment to remove the CTScan instance data from the database.