WEB CURATOR TOOL

System Administrator Guide

31 July 2009

Contents

Introduction 3
Contents of this document 3
Getting Started 4
Prerequisites 4
Supported platforms 4
Other platforms 4
Optional prerequisites 4
Setting up the WCT database 5
Setup using Oracle 10g 5
Setup using PostgresQL 8.1 6
Setup using MySQL 5.0 7
Setting up the WCT Application Servers 8
Deploying WCT to Tomcat 8
Configure the Database Connection 9
Configure LDAP Authentication (Unencrypted) 10
Configure LDAP Authentication (Encrypted using TLS or SSL) 13
Configure the Digital Asset Store 14
Configure a Harvest Agent 14
Set the Attachments Directories 16
Logon to WCT 17
Troubleshooting setup 18
Configuration options 19
Web Curator Core – context.xml 19
Web Curator Core – wct-core.xml 19
Web Curator Core – wct-core-security.xml 22
Web Curator Digital Asset Store – wct-das.xml 26
Web Curator Harvest Agent – wct- agent.xml 28
Web Curator Tool – SOAP Service Configuration 31
Graceful shutdown and restart 33
Appendix A: Creating a truststore and importing a certificate 34
Appendix B: The OMS archive adapter 35

Introduction

This guide, designed for a System Administrator, covers installation and setup of the Web Curator Tool. An electronic copy can be downloaded from the WCT Sourceforge website: http://webcurator.sourceforge.net/

For information on using the Web Curator Tool, see the Web Curator Tool Quick Start Guide and the Web Curator Tool online help.

Contents of this document

Following this introduction, the Web Curator Tool System Adminsitrator Guide includes the following sections:

- **Getting Started** (page 4) — covers prerequisites, supported platforms, other platforms, and optional prerequisites for using the Web Curator Tool
- **Setting up the WCT database** (page 5)— procedures for setup using Oracle 10g and PostgresWL 8.1
- **Setting up the WCT Application Servers** (page 8) — procedures for deploying WCT to Tomcat, includes configuration options and troubleshooting
- **Appendix A: Creating a truststore and importing a certificate** (page 34).

Getting Started

The following section explains how to get the Web Curator Tool up and running.

Prerequisites

The following are required to successfully install and run the Web Curator Tool:

- Java 1.5 JDK or above
- Apache Tomcat 5.5.X or above (the application has been tested on Tomcat 5.5.15)
- A database server (select one of the databases below)
  - Oracle 10g
  - Postgresql 8.1.
  - MySQL 5.0

Other versions of the required products may be compatible with the Web Curator Tool but they have not been tested. Due to the products use of Hibernate for database persistence other database platforms should work, if the product is rebuilt with the correct database dialect. However only Postgesql, Oracle 10g, and MySQL have been tested.

Supported platforms

The following platforms have been used during the development of the Web Curator Tool:

- Sun Solaris 9
- Red Hat Linux EL3.

Other platforms

The following platform was used during the Development of the Web Curator tool but is not explicitly supported:

- Windows 2000.

Optional prerequisites

The following prerequisites are optional:

- LDAP compliant directory (for external authentication)
- ANT 1.6.5 or above (required to build from source).

Setting up the WCT database

Currently the WCT has been tested with both Oracle 10g and PostgresQL 8.1.

Setup using Oracle 10g

This guide assumes you have installed and configured Oracle 10g prior to setting up the WCT database and schema.

1 Setup two schemas: one called DB_WCT that owns the tables and one called USR_WCT that the application uses to query the tables. The USR_WCT schema should have limited rights. You can use a script similar to the following:

```
create user db_wct identified by password default tablespace
wct_data quota unlimited on wct_data;

create user usr_wct identified by password default tablespace
wct_data quota unlimited on wct_data;

grant create session to usr_wct;
grant connect,resource to db_wct;
```

2 Run the following SQL scripts under the DB_WCT user or SYSTEM account:

```
sql/oracle/wct-schema-oracle.sql
sql/oracle/wct-schema-grants.sql
sql/oracle/wct-oracle-bootstrap.sql
sql/oracle/wct-indexes-oracle.sql
```

3 Locate the correct JDBC driver for Oracle, which should be distributed with the Oracle install media.
- The JDBC driver should be called ojdbc1410g.jar
- The driver will need to be placed into the $TOMCAT_HOME/common/lib/ directory.
- Also required in this directory is the jta.jar

Notes: A password strategy should be defined for the system, and the db_wct & usr_wct passwords should be changed in the scripts and application property files to conform to this strategy. To encourage this, the passwords in the supplied database creation script are not set to 'password'.

The bootstrap user script creates a User with a name of 'bootstrap' and a password of 'password'. Use this account to login to the application once it is up and running. You can use the bootstrap account to create other users and agencies. Once you have setup valid users, it is best to disable the bootstrap user for security reasons.

Setup using PostgresQL 8.1

This guide assumes you have installed and configured Postgresql 8.1 prior to setting up the WCT database and schema.

1 Setup two schema:

```
CREATE DATABASE "Dwct" WITH ENCODING='UTF8';
\c Dwct
CREATE SCHEMA db_wct;
CREATE ROLE usr_wct LOGIN PASSWORD 'password'
  NOINHERIT
   VALID UNTIL 'infinity';
grant usage on schema db_wct to usr_wct;
```

Then run the following SQL scripts under the DB_WCT user

```
sql/postgres/wct-schema-postgres.sql
sql/postgres/wct-schema-grants.sql
sql/postgres/wct-postgres-bootstrap.sql
sql/postgres/wct-indexes-postgresql.sql
```

2 The Postgres JDBC driver is included in the CVS repository under /etc/ directory.

- The Postgres driver is called postgresql-8.1-404.jdbc3.jar
- The driver will need to be placed into the $TOMCAT_HOME/common/lib/ directory.
- Also required in the $TOMCAT_HOME/common/lib/ directory is the jta.jar

Notes: A password strategy should be defined for the system, and the usr_wct password should be changed in the scripts and application property files to conform to this strategy. To encourage this, the password in the supplied database creation script is not set to 'password'.

The bootstrap user script creates a User with a name of 'bootstrap' and a password of 'password'. Use this account to login to the application once it is up and running. You can use the bootstrap account to create other users and agencies. Once you have setup valid users, it is best to disable the bootstrap user for security reasons.

Setup using MySQL 5.0

This guide assumes you have installed and configured MySQL 5.0 prior to setting up the WCT database and schema.

1 Create the database:

```
CREATE DATABASE DB_WCT;
\u DB_WCT
create user usr_wct@localhost identified by 'password';
grant all on DB_WCT.* to usr_wct@localhost;
```

Then run the following SQL scripts under the root user

```
sql/mysql/wct-schema-mysql.sql
sql/mysql/wct-schema-grants-mysql.sql
sql/mysql/wct-mysql-bootstrap.sql
sql/mysql/wct-indexes-mysql.sql
```

2 Download the MySQL JDBC driver from the MySQL website.

- The driver will need to be placed into the $TOMCAT_HOME/common/lib/ directory.
- Also required in the $TOMCAT_HOME/common/lib/ directory is the jta.jar

Notes: A password strategy should be defined for the system, and the usr_wct password should be changed in the scripts and application property files to conform to this strategy. To encourage this, the password in the supplied database creation script is not set to 'password'.

The bootstrap user script creates a User with a name of 'bootstrap' and a password of 'password'. Use this account to login to the application once it is up and running. You can use the bootstrap account to create other users and agencies. Once you have setup valid users, it is best to disable the bootstrap user for security reasons.

Deploying WCT to Tomcat

There are three major components to the deployment of the Web Curator Tool:

- the web curator core (wct.war)
- the web curator harvest agent (wct-harvest-agent.war)
- the web curator digital asset store (wct-store.war).

Each of these three components must be deployed for the Web Curator Tool to be fully functional and more than one harvest agent can be deployed if necessary. Each Harvest Agent is capable of carrying out harvest actions. The more harvest agents deployed the more harvesting that can be done at any one point in time. The harvest agents and digital asset store can reside on any machine within the network, as they use SOAP over HTTP to communicate with each other.

To deploy WCT to Tomcat:

1. Make sure you have installed and configured both Java 1.5 JDK and Apache-Tomcat 5.5.X successfully.

2. Set up the JMX Remote control and access files for the WCT core and every Harvest Agent.
   - Create a jmxremote.password file by copying the file jmxremote.password.template this file will be in your JDK's jre\lib\management directory.

     You can use the property -Dcom.sun.management.jmxremote.password.file to point to a different location.

   - It is important that this file is protected. If using Windows, refer to the following link to protect the file using the O/S: http://java.sun.com/j2se/1.5.0/docs/guide/management/security-windows.html

   - If using *nix platform, protect the file using

     ```
     chmod 600 jmxremote.password.
     ```

– Also enable the JMX Remote port (any high port can be used) by adding the following to your $TOMCAT_HOME/bin/catalina.sh script:

```
JAVA_OPTS=-Dcom.sun.management.jmxremote.port=9004
```

IMPORTANT: *Make sure this change is applied to the Core and any Harvest Agent deployed onto a different machine.*

3 Deploy the WAR files into Tomcat. The simplest deployment is to deploy all three WAR files into the same Tomcat container.

– You can copy the WAR files into the $TOMCAT_HOME/webapps/ directory.
– Provided Tomcat is configured correctly, when you start Tomcat the WAR files will be exploded and the application will start.

4 Shut down Tomcat once the WAR files have been extracted. This will allow you to modify the configuration files in the following steps.

Configure the Database Connection

The open source version of the Web Curator Tool is configured to use a local PostgreSQL database. If you are using any other database, or are using a database server, you will need to change the database configuration.

5 Set the correct database dialect in TOMCAT/webapps/wct/WEB-INF/classes/**wct-core.xml**:

```
<property name="hibernateProperties">
  <props>
    <prop key="hibernate.dialect">
      org.hibernate.dialect.PostgreSQLDialect
    </prop>
    <prop key="hibernate.show_sql">false</prop>
```

The appropriate dialects are shown in the table below.

Database	**Dialect**
Oracle	org.hibernate.dialect.OracleDialect
PostgreSQL	org.hibernate.dialect.PostgreSQLDialect
MySQL	org.hibernate.dialect.MySQLDialect

6 Edit the context.xml file in TOMCAT/webapps/wct/META-INF.

```
<?xml version="1.0" encoding="UTF-8"?>
<Context>
  <Resource
    name="jdbc/wctDatasource"
    type="javax.sql.DataSource"
```

```
    password="PASSWORD"
    driverClassName="DRIVER"
    maxIdle="2"
    maxWait="5000"
    validationQuery="VALIDATION_QUERY"
    username="USERNAME"
    url="JDBC_URL"
    maxActive="10 "/>
</Context>
```

Set the username and password properties as appropriate for your database. If you have followed the defaults, then these should remain as USR_WCT/USR_WCT.

The remaining properties should be set as follows:

Oracle

Attribute	Value
DRIVER	oracle.jdbc.driver.OracleDriver
VALIDATION_QUERY	select count(1) from DUAL
JDBC_URL	jdbc:oracle:thin:@servername:port/SID

PostgreSQL

Attribute	Value
DRIVER	org.postgresql.Driver
VALIDATION_QUERY	select 1+1
JDBC_URL	jdbc:postgresql://servername:port/database

MySQL

Attribute	Value
DRIVER	com.mysql.jdbc.Driver
VALIDATION_QUERY	select 1+1
JDBC_URL	jdbc:mysql://servername:port/database

7 Copy the context.xml file to the TOMCAT/conf/Catalina/localhost directory. Delete the existing wct.xml file if it exists. Now rename the context.xml file to wct.xml.

Configure LDAP Authentication (Unencrypted)

8 If you wish to use an external Directory for Authentication, then WCT should be configured to allow this. Unencrypted authentication can be done very simply with your directory by modifying the wct-core-security.xml file.

On the following page are the parameters that need changing in order to communicate with your Directory.

The Directory must support LDAP.

```
<bean id="authenticationManager"
class="org.acegisecurity.providers.ProviderManager" abstract="false"
singleton="true" lazy-init="default" autowire="default" dependency-
check="default">
      <property name="providers">
      <list>
            <ref bean="ldapAuthenticator"/>
            <ref bean="daoAuthenticationProvider"/>
      </list>
      </property>
</bean>

<bean id="initialDirContextFactory"
class="org.acegisecurity.providers.ldap.DefaultInitialDirContextFacto
ry">
      <constructor-arg value="ldap://hostname_of_directory/"/>
</bean>

<bean id="ldapAuthenticator"
class="org.acegisecurity.providers.ldap.LdapAuthenticationProvider"
abstract="false" singleton="true" lazy-init="default"
autowire="default" dependency-check="default">
      <constructor-arg>
      <bean
class="org.acegisecurity.providers.ldap.authenticator.BindAuthenticat
or">
            <constructor-arg>
            <ref local="initialDirContextFactory"/>
            </constructor-arg>
            <property name="userDnPatterns">
            <list>
            <value>cn={0},ou=wct,o=global</value>
            </list>
            </property>
      </bean>
      </constructor-arg>
      <constructor-arg>
      <bean class="org.webcurator.auth.ldap.WCTAuthoritiesPopulator">
            <property name="authDAO">
            <ref bean="userRoleDAO"/>
            </property>
      </bean>
      </constructor-arg>
</bean>
```

The authenticationManager Bean defines an LDAP provider called ldapAuthenticator. It is this bean that provides the connectivity to your Directory server via LDAP.

The two parameters of interest are:

- The constructor argument for the *InitialDirContextFactory*, which defines the URL for the directory. This is normally something like ldap://mydirectory.natlib.co.nz/
- The *userDNPatterns* property of the *ldapAuthenticator* bean. This allows the Directory DN to be defined. For example, if a user logs in with the username “gordonp” the Directory will be

queried using the distinguished name of “cn=gordonp, ou=wct, o=global”. So the user must exist within the global organisation and the wct organisation unit.

Configure LDAP Authentication (Encrypted using TLS or SSL)

9 If you want all credentials passed to the Directory server to be protected then the ldap traffic should be encrypted using TLS or SSL.

- The only difference to the wct-core-security.xml file from step 4 is the following change:

```
<bean id="initialDirContextFactory"
class="org.acegisecurity.providers.ldap.DefaultInitialDirCon
textFactory">
      <constructor-arg
value="ldaps://hostname_of_directory/"/>
</bean>
```

- If using TLS or SSL then you must configure Tomcat to allow secure communication with your Directory by adding the following to your $TOMCAT_HOME/bin/catalina.sh script:

```
JAVA_OPTS= -
Djavax.net.ssl.trustStore=/var/wctcore/ssl/wct.ts     -
Djavax.net.ssl.trustStorePassword=password
```

This points tomcat to a Truststore that contains the public key for you directory. If your directory utilises a correctly signed certificate, you may not need this, as the default truststore provided by Java contains all the major root certificates. However if you directory uses a self-signed certificate then you will need to export the public key of that certificate and import it into your truststore (i.e. /var/wctcore/ssl/wct.ts). Alternatively you can import the self-signed certificate into the default Java truststore.

For details on how to create a truststore and import a certificate, see *Appendix A: Creating a truststore and importing a certificate*.

Configure the Digital Asset Store

10 Set the Base Directory of the Digital Asset Store to a valid location on the server. Also make sure the directory or share has enough free disk space.

– The configuration for the DAS is found in the **wct-das.xml** file:

```
<bean name="arcDigitalAssetStoreService"
class="org.webcurator.core.store.arc.ArcDigitalAssetStoreService"
abstract="false" singleton="true" lazy-init="default"
autowire="default" dependency-check="default">
      <property name="baseDir">
            <value>/tmp/arcstore</value>
      </property>
</bean>
```

Configure a Harvest Agent

11 Make sure the following parameters are correct for your environment in the **wct-agent.xml** file:

```
<bean id="harvestAgent"
class="org.webcurator.core.harvester.agent.HarvestAgentHeritrix"
abstract="false" singleton="true" lazy-init="default"
autowire="default" dependency-check="default">
      <property name="baseHarvestDirectory">
            <value type="java.lang.String">/tmp/harvest-agent</value>
      </property>
      <property name="host">
            <value type="java.lang.String">localhost</value>
      </property>
      <property name="maxHarvests">
            <value>2</value>
      </property>
      <property name="port">
            <value>8080</value>
      </property>
      <property name="name">
            <value type="java.lang.String">My Harvest Agent</value>
      </property>
      <property name="provenanceNote">
            <value type="java.lang.String">Original Harvest</value>
      </property>
      <property name="allowedAgencies">
            <list>
            </list>
      </property>
      <property name="digitalAssetStore">
            <ref bean="digitalAssetStore"></ref>
      </property>
      <property name="harvestCoordinatorNotifier">
            <ref bean="harvestCoordinatorNotifier"></ref>
      </property>
</bean>
<bean id="harvestCoordinatorNotifier"
class="org.webcurator.core.harvester.coordinator.HarvestCoordinatorNo
tifier" abstract="false" singleton="true" lazy-init="default"
autowire="default" dependency-check="default">
```

```
      <property name="service">
            <value
type="java.lang.String">/wct/services/urn:WebCuratorTool</value>
      </property>
      <property name="host">
            <value type="java.lang.String">localhost</value>
      </property>
      <property name="port">
            <value>8080</value>
      </property>
</bean>
<bean id="digitalAssetStore"
class="org.webcurator.core.store.DigitalAssetStoreSOAPClient"
abstract="false" singleton="true" lazy-init="default"
autowire="default" dependency-check="default">
      <property name="service">
            <value type="java.lang.String">/wct-
store/services/urn:DigitalAssetStore</value>
      </property>
      <property name="host">
            <value type="java.lang.String">localhost</value>
      </property>
      <property name="port">
            <value>8080</value>
      </property>
</bean>

…

<bean id="heartbeatTrigger"
class="org.springframework.scheduling.quartz.SimpleTriggerBean">
      <property name="group" value="HeartBeatTriggerGroup"/>
      <property name="name" value="HeartBeatTrigger"/>
      <property name="jobDetail" ref="heartbeatJob"/>
      <!-- 10 seconds -->
      <property name="startDelay" value="10000"/>
      <!-- repeat every 30 seconds -->
      <property name="repeatInterval" value="30000"/>
</bean>
```

– In addition to setting the Harvest Agent parameters, you may also want to change the default Heritrix profile that is shipped with the WCT. The most likely settings to change are what web proxy server to use when harvesting content. The setting can be found in the **WEB-INF/classes/default-profile.xml**:

```
<newObject name="HTTP" class="org.archive.crawler.fetcher.FetchHTTP">
        <boolean name="enabled">true</boolean>
        <map name="filters">
        </map>
        <map name="midfetch-filters">
        </map>
        <integer name="timeout-seconds">1200</integer>
        <integer name="sotimeout-ms">20000</integer>
        <long name="max-length-bytes">0</long>
        <boolean name="ignore-cookies">false</boolean>
        <boolean name="use-bdb-for-cookies">true</boolean>
        <string name="load-cookies-from-file"></string>
        <string name="save-cookies-to-file"></string>
        <string name="trust-level">open</string>
        <stringList name="accept-headers">
        </stringList>
        <string name="http-proxy-host"></string>
        <string name="http-proxy-port"></string>
        <string name="default-encoding">ISO-8859-1</string>
        <boolean name="sha1-content">true</boolean>
        <boolean name="send-connection-close">true</boolean>
        <boolean name="send-referer">true</boolean>
        <boolean name="send-range">false</boolean>
      </newObject>
```

– If you don't have a web proxy then just leave the values blank.

Heritrix does not currently support authenticated proxy access, so the proxy server must allow unauthenticated access.

Set the Attachments Directories

12 Set the attachments directories in the server-config.wsdd files for all three components. This file is found in the WEB-INF directory of each application. This directory must exist and be accessible by the Tomcat server.

```
<parameter name="attachments.Directory" value="/tmp/attach"/>
```

Logon to WCT

Once you have started up the Web Curator Tool logon to the application using the 'bootstrap' user with the default password of 'password'. This account has enough privilege to create other Agencies and Users within the system. Once you have configured valid WCT users and tested their login's work, you should disable the bootstrap user.

The URL to access WCT running on Apache/Tomcat will be similar to the one displayed below:
http://localhost/wct/ where 'localhost' can be replaced with your server name. Note, if using tomcat only, the default port for tomcat is 8080, changing the URL to http://localhost:8080/wct/ will allow you to connect directly to Tomcat.

The other common trap is not defining the default bandwidth for the system. On start-up of WCT the system bandwidth is set to 0 KB's for every day of the week. Before Harvests can be initiated you must specify a base bandwidth for each of the days you plan to harvest on. In order to setup the bandwidth you must logon as a user that has the 'Manage Web Harvester System' privilege set (usually an WCT Administrator). The Bandwidth screen can be found under the 'Management -> Harvester Configuration -> Bandwidth' section of the site.

For more information on using the Web Curator Tool, refer to the Quick Start Guide.

Troubleshooting setup

See the following table to troubleshoot Web Curator Tool setup.

Problem	Possible solution
Database connection failure	Check that the WCT core data source is defined correctly in the wct/META-INF/context.xml and that the server can communicate with this host on the specified port.
LDAP configuration failure	If problems occur with getting TLS working with ldap, then switch on the SSL debug mode within Tomcat by adding the following to the JAVA_OPTS environment variable. The debug will display on the console. `-Djavax.net.debug=ssl`
JMX remote register failure	Tomcat will not start if the permissions are incorrect on the jmxremote.password file. Check that the jmxremote.password file exists and has the correct ownership.
Communication failure on Heartbeat	Validate that the distributed agents have the correctly defined central host and can communicate with this host over HTTP.
Failure on storing the harvest to the store	Validate that the Digital Asset Store has been configured with the correct directory settings and has write access to the specified directory.
Failure on Harvest attempt (or Harvest action appears to hang)	`2006-07-04 07:51:31,640 ERROR [http-8080-Processor24] agent.HarvestAgentHeritrix (HarvestAgentHeritrix.java:88) - Failed to initiate harvest for 262147 : Failed to create the job profile C:\tmp\harvest-agent\262147\order.xml. org.webcurator.core.harvester.agent.exception.HarvestAgentException: Failed to create the job profile **C:\tmp\harvest-agent\262147\order.xml**.` `at org.webcurator.core.harvester.agent.HarvestAgentHeritrix.createProfile(HarvestAgentHeritrix.java:542)` `at org.webcurator.core.harvester.agent.HarvestAgentHeritrix.initiateHarvest(HarvestAgentHeritrix.java:79)` `at org.webcurator.core.harvester.agent.HarvestAgentSOAPService.initiateHarvest(HarvestAgentSOAPService.java:37)` If any error similar to the one above occurs, it is usually related to an incomplete harvest taking place. If this occurs you will need to remove the Target Instance sub-directory from the deployed baseHarvestDirectory as specified in the wct-agent.xml. In the example above you would delete the directory called c:\tmp\harvest-agent\262147

Configuration options

This section describes options for configuring the Web Curator Tool.

Web Curator Core – context.xml

The /META-INF/context.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<Context>
  <Resource
    name="jdbc/wctDatasource"
    type="javax.sql.DataSource"
    password="@@SCHEMA_PASSWORD@@"
    driverClassName="@@SCHEMA_DRIVER@@"
    maxIdle="@@SCHEMA_MAXIDLE@@"
    maxWait="5000"
    validationQuery="@@SCHEMA_QUERY@@"
    username="@@SCHEMA_USER@@"
    url="@@SCHEMA_URL@@"
    maxActive="@@SCHEMA_MAXACTIVE@@"/>
</Context>
```

This file defines the data source to use for the WCT and specifies the JDBC driver class, database URL, username, password, max and min connections and the keep alive query. The parameters surrounded by @@ characters are replaced when this file is built using ANT, with the appropriate values from the build.properties file.

Web Curator Core – wct-core.xml

The /WEB-INF/classes/wct-core.xml

```
<bean id="harvestCoordinator"
class="org.webcurator.core.harvester.coordinator.HarvestCoordinatorIm
pl" abstract="false" singleton="true" lazy-init="default"
autowire="default" dependency-check="default">
      <property name="harvestCoordinatorDao">
            <ref bean="harvestCoordinatorDao"/>
      </property>
      <property name="targetInstanceDao">
            <ref bean="targetInstanceDao"/>
      </property>
      <property name="harvestAgentFactory">
            <ref bean="harvestAgentFactory"/>
      </property>
      <property name="digitalAssetStoreConfig">
            <ref bean="digitalAssetStore"/>
      </property>
      <property name="minimumBandwidth" value="10"/>
      <property name="maxBandwidthPercent" value="80"/>
      <property name="daysBeforeDASPurge" value="60" />
      <property name="daysBeforeAbortedTargetInstancePurge"
value="7"/>
      </bean>
```

The **harvestCoordinator** is responsible for the coordination of harvest activity across all of the Harvest Agents. This is where the minimum bandwidth (in KB/s) and maximum bandwidth percentages are defined for all agents. Also defined in the Co-ordinator is the number of days before the Digital Asset Store is purged as well as the number of days before data remaining after aborted harvests is purged.

```
<bean id="schedulePatternFactory"
class="org.webcurator.domain.SpringSchedulePatternFactory">
      <property name="patterns">
      <list>
      <bean class="org.webcurator.domain.model.core.SchedulePattern">
        <property name="scheduleType" value="1"/>
        <property name="description" value="Every Monday at 9:00pm"/>
        <property name="cronPattern" value="00 00 21 ? * MON *"/>
      </bean>
      </list>
      </property>
</bean>
```

The **schedulePatternFactory** defines all the default CRON patterns used by the WCT to schedule Targets for harvest. For each additional SchedulePattern required an additional SchedulePattern bean should be added to the list.

```
<bean id="processScheduleTrigger"
class="org.springframework.scheduling.quartz.SimpleTriggerBean">
      <property name="group" value="ProcessScheduleTriggerGroup"/>
      <property name="name" value="ProcessScheduleTrigger"/>
      <property name="jobDetail" ref="processScheduleJob"/>
      <!-- 10 seconds -->
      <property name="startDelay" value="10000"/>
      <!-- repeat every 30 seconds -->
      <property name="repeatInterval" value="30000"/>
</bean>
```

The **processScheduleTrigger** defines when the heartbeat activity is checked on the registered Agents. The time is measured in milliseconds.

```
<bean id="mailServer"
class="org.webcurator.core.notification.MailServerImpl"
abstract="false" singleton="true" lazy-init="default"
autowire="default" dependency-check="default">
        <constructor-arg>
            <props>
              <prop key="mail.smtp.host">mailhost</prop>
              <prop key="mail.transport.protocol">SMTP</prop>
              <prop key="mail.smtp.port">25</prop>
            </props>
      </constructor-arg>
</bean>
```

The **mailServer** bean is responsible for communicating with an SMTP mail server for sending email notifications.

```
<bean id="inTrayManager"
class="org.webcurator.core.notification.InTrayManagerImpl"
abstract="false" singleton="true" lazy-init="default"
autowire="default" dependency-check="default">
        <property name="inTrayDAO"><ref bean="inTrayDao"/></property>
        <property name="userRoleDAO"><ref
bean="userRoleDAO"/></property>
        <property name="agencyUserManager"><ref
bean="agencyUserManager"/></property>
        <property name="mailServer"><ref
bean="mailServer"/></property>
        <property name="sender"
value="wct@webcurator.sourceforge.net"/>
        <property name="messageSource"><ref
bean="messageSource"/></property>
        <property name="wctBaseUrl" value="http://wcthost/wct/"/>
</bean>
```

The **inTrayManager** is responsible for informing users of Tasks or Notification messages. This uses the mailServer bean to send email. Also defined here is the sender of the automated system Tasks and Notifications.

```
<bean id="groupSearchController"
class="org.webcurator.ui.groups.controller.GroupSearchController">
      <property name="targetManager" ref="targetManager"/>
      <property name="agencyUserManager" ref="agencyUserManager"/>
      <property name="messageSource" ref="messageSource"/>
      <property name="defaultSearchOnAgencyOnly" value="true"/>
</bean>
```

The **groupSearchController** defines how the default search is handled on the Groups tab. When **defaultSearchOnAgencyOnly** is set to *true*, the user name is omitted from the default Group search filter allowing the display of all groups for the current user's agency. When **defaultSearchOnAgencyOnly** is set to *false*, the user name is included in the filter and only those Groups owned by the current user are displayed.

```
<bean id="archiveAdapter"
class="org.webcurator.core.archive.ArchiveAdapterImpl"
abstract="false" singleton="true" lazy-init="default"
autowire="default" dependency-check="default">
            <property name="digitalAssetStore">
                  <ref bean="digitalAssetStore"></ref>
            </property>
            <property name="targetInstanceManager">
                  <ref bean="targetInstanceManager"></ref>
            </property>
            <property name="targetManager">
                  <ref bean="targetManager"></ref>
            </property>
            <property name="targetReferenceMandatory" value="false"/>
      </bean>
```

The **archiveAdapter** The archive adapter provides the mechanism for archiving a harvested target instance into an archive repository. When **targetReferenceMandatory** is set to *true (or is omitted)*, the owning Target for a Target Instance being archived must have a Target Reference defined in order for archiving to be attempted. When **targetReferenceMandatory** is set to *false*, there is no need for the owning Target to have a Target Reference defined.

The **WCT hyperlink** is also defined here (the hyperlink is used in Tasks to link the user back into the appropriate WCT object the task relates too).

Web Curator Core – wct-core-security.xml

The **wct-core-security.xml** contains all of the security, Authentication and Authorisation settings to be used by the Web Curator Tool.

```
<bean id="authenticationManager"
class="org.acegisecurity.providers.ProviderManager" abstract="false"
singleton="true" lazy-init="default" autowire="default" dependency-
check="default">
      <property name="providers">
      <list>
      <ref bean="ldapAuthenticator" />
      <ref bean="daoAuthenticationProvider" />
      </list>
      </property>
</bean>

<bean id="initialDirContextFactory"
class="org.acegisecurity.providers.ldap.DefaultInitialDirContextFacto
ry">
      <constructor-arg value="ldap://www.myldapserver.com/"/>
</bean>

<bean id="ldapAuthenticator"
class="org.acegisecurity.providers.ldap.LdapAuthenticationProvider"
```

```
abstract="false" singleton="true" lazy-init="default"
autowire="default" dependency-check="default">
      <constructor-arg>
      <bean
class="org.acegisecurity.providers.ldap.authenticator.BindAuthenticat
or">
      <constructor-arg><ref local="initialDirContextFactory"/>
            </constructor-arg>
            <property name="userDnPatterns">
                  <list>
                  <value>cn={0},ou=wct,o=global</value>
                  </list>
            </property>
      </bean>
      </constructor-arg>
      <constructor-arg>
      <bean class="org.webcurator.auth.ldap.WCTAuthoritiesPopulator">
      <property name="authDAO">
            <ref bean="userRoleDAO"/>
      </property>
      </bean>
      </constructor-arg>
</bean>
```

This is where the **LDAPAuthenticator** can be plugged in if the Tool is to use an external Directory service for Authentication. The example above identifies the parameters that are required to plug in the external directory service.

The **harvestResourceUrlMapper** bean is responsible for overwriting the standard WCT browse ULSs in with the review tool with a custom url and replacing elements of that url with the correct items in the harvest resource to configure it the following bean must be included (or un-commented)

```
<bean id="harvestResourceUrlMapper"
class="org.webcurator.ui.tools.controller.HarvestResourceUrlMapper">
            <property name="urlMap" value =
"http://localhost.archive.org:8080/wayback/wayback/*/{$HarvestResource.Name}"/>
      <property name="enabled" value="true"/>

</bean>
```

This bean must then be associated with the **treeToolController** , **treeToolControllerAJAX** and the **qualityReviewToolController** in their bean configuration (the appropriate properties in bold):

```
<bean id="treeToolController"
class="org.webcurator.ui.tools.controller.TreeToolController"
abstract="false" singleton="true" lazy-init="default"
autowire="default" dependency-check="default">
            <property name="supportedMethods">
                  <value type="java.lang.String">GET,POST</value>
            </property>
            <property name="qualityReviewFacade">
```

```
                  <ref bean="qualityReviewFacade"/>
            </property>
            <property name="validator">
                  <bean
class="org.webcurator.ui.tools.validator.TreeToolValidator"/>
            </property>
            <!-- HRM1: uncomment to enable redirect to the custom url-
->
            <property name="harvestResourceUrlMapper"
ref="harvestResourceUrlMapper"/>

      </bean>

      <bean id="treeToolControllerAJAX"
class="org.webcurator.ui.tools.controller.TreeToolControllerAJAX"
abstract="false" singleton="true" lazy-init="default"
autowire="default" dependency-check="default">
            <property name="supportedMethods">
                  <value type="java.lang.String">GET,POST</value>
            </property>
            <property name="qualityReviewFacade">
                  <ref bean="qualityReviewFacade"/>
            </property>
            <property name="validator">
                  <bean
class="org.webcurator.ui.tools.validator.TreeToolValidator"/>
            </property>
            <!-- HRM2: uncomment to enable redirect to the custom url-
->
            <property name="harvestResourceUrlMapper"
ref="harvestResourceUrlMapper"/>

      </bean>

            <bean id="qualityReviewToolController"
class="org.webcurator.ui.tools.controller.QualityReviewToolController"
abstract="false" singleton="true" lazy-init="default"
autowire="default" dependency-check="default">
            <property name="supportedMethods">
                  <value type="java.lang.String">GET</value>
            </property>
            <property name="targetInstanceManager"><ref
bean="targetInstanceManager"/></property>
            <property name="targetManager" ref="targetManager"/>
            <property name="archiveUrl"
value="http://web.archive.org/web/*/"/>
            <!-- HRM3: uncomment to enable redirect to the custom url-
->
            <property name="harvestResourceUrlMapper"
ref="harvestResourceUrlMapper"/>
            <property name="targetInstanceDao"
ref="targetInstanceDao"/>

      </bean>
```

The urlMap property of the **harvestResourceUrlMapper** can have any of the following substituted value from the harvest resource:

- {$HarvestResource.Name}
- {$HarvestResource.Length}

- {$HarvestResource.Oid}
- {$HarvestResource.StatusCode}
- {$HarvestResult.CreationDate[,DateFormat]}
- {$HarvestResult.DerivedFrom}
- {$HarvestResult.HarvestNumber}
- {$HarvestResult.Oid}
- {$HarvestResult.ProvenanceNote}
- {$HarvestResult.State}

The HarvestResult CreationDate substitution's format can be controlled by supply a valid simple date format after a comma within the curly brackets e.g. {$HarvestResult.CreationDate,ddMMyy } for 1 Nov 2008 will show "011108"

Web Curator Digital Asset Store – wct-das.xml

```
<bean name="arcDigitalAssetStoreService"

      class="org.webcurator.core.store.arc.ArcDigitalAssetStoreServic
e"
            abstract="false" singleton="true" lazy-init="default"
            autowire="default" dependency-check="default">
            <!-- the base directory for the arc store -->
            <property name="baseDir">
                  <value>/opt/digital-asset-store/</value>
            </property>
            <property name="archive">
                  <ref bean="archive" />
            </property>
</bean>
```

This section of the file specifies the location where Archives are stored on the file system. The Digital Asset store holds these files for a period of time before they are purged. See the wct-core.xml file for the purge parameters.

Using the File System Adapter (Default option)

```
<bean id="archive"
            class="org.webcurator.core.archive.file.FileArchive"
abstract="false"
            singleton="true" lazy-init="default" autowire="default"
            dependency-check="default">
            <property name="archiveRepository">
                  <value>/opt/library-archive/</value>
            </property>
            <property name="archiveLogDirectory">
                  <value>logs</value>
            </property>
            <property name="archiveReportDirectory">
                  <value>reports</value>
            </property>
            <property name="archiveArcDirectory">
                  <value>arcs</value>
            </property>
</bean>
```

The **FileArchive** bean writes files to a file system when they are archived. This directory should be permanent storage that is backed up, as these files are the definitive web archives that user wishes to store for prosperity.

Web Curator Harvest Agent – wct- agent.xml

The configuration for the harvest agent is stored in the within the /WEB-INF/classes/wct-agent.xml file.

```
<bean id="harvestAgent"
class="org.webcurator.core.harvester.agent.HarvestAgentHeritrix"
abstract="false" singleton="true" lazy-init="default"
autowire="default" dependency-check="default">
            <!-- name of the directory where the temporary harvest
data is stored -->
            <property name="baseHarvestDirectory"
value="/tmp/agentstore/"/>
            <!-- agent host name or ip address that the core knows
about -->
            <property name="host" value="agent1host"/>
            <!-- the max number of harvest to be run concurrently on
this agent -->
            <property name="maxHarvests" value="2"/>
            <!-- the port the agent is listening on for http
connections -->
            <property name="port" value="8080"/>
            <!-- the name of the agent. must be unique -->
            <property name="name" value="My Agent 1"/>
            <!-- the note to send with the harvest result. -->
            <property name="provenanceNote" value="Original
Harvest"/>
            <!-- the number of alerts that occur before a
notification is sent  -->
            <property name="alertThreshold" value="200"/>
            <!-- the list of agency names that can use this agent -->
            <property name="allowedAgencies">
                  <list>
                  </list>
            </property>
            <property name="digitalAssetStore">
                  <ref bean="digitalAssetStore"></ref>
            </property>
            <property name="harvestCoordinatorNotifier">
                  <ref bean="harvestCoordinatorNotifier"></ref>
            </property>
</bean>
```

The ***HarvestAgent*** is responsible for specifying where the harvest agent is located and it name. This is also where the agent specifies the maximum number of concurrent harvests it can carry out.

If this harvest agent can only harvest material for a set number of agencies, then they can be listed in the *allowedAgencies* property. An empty list implies that any Agency can use the Harvest Agent. The configuration below shows two agencies defined

```
<property name="allowedAgencies">
      <list>
            <value>National Library of New Zealand</value>
            <value>British Library</value>
      </list>
</property>
```

```
<bean id="harvestCoordinatorNotifier"
class="org.webcurator.core.harvester.coordinator.HarvestCoordinatorNo
tifier" abstract="false" singleton="true" lazy-init="default"
autowire="default" dependency-check="default">
      <!-- the name of the core harvest agent listener web service --
>     <property name="service"
value="/wct/services/urn:WebCuratorTool"/>
      <!-- the host name or ip address of the core -->
      <property name="host" value="wct-core-host"/>
      <!-- the port that the core is listening on for http
connections -->
      <property name="port" value="8080"/>
      <property name="agent" ref="harvestAgent"/>
</bean>
```

The ***harvestCoordinatorNotifier*** bean is used to specify how the Harvest Agent should communicate back to the WCT Core.

```
<bean id="digitalAssetStore"
class="org.webcurator.core.store.DigitalAssetStoreSOAPClient"
abstract="false" singleton="true" lazy-init="default"
autowire="default" dependency-check="default">
      <!-- the name of the digital asset store web service -->
      <property name="service" value="/wct-
store/services/urn:DigitalAssetStore" />
      <!-- the host name or ip address of the digital asset store -->
      <property name="host" value="wct-das-host"/>
      <!-- the port that the digital asset store is listening on for
http connections -->
      <property name="port" value="8080"/>
</bean>
```

The ***digitalAssetStore*** bean is used to specify how the Harvest Agent communicates back to the Digital Asset Store.

```
<bean id="memoryChecker"
            class="org.webcurator.core.check.MemoryChecker"
abstract="false"
            singleton="true" lazy-init="default" autowire="default"
            dependency-check="default">
            <!-- The amount of memory in KB that can be used before a
warning notification is sent  -->
            <property name="warnThreshold" value="56320"/>
            <!-- The amount of memory in KB that can be used before
an error notification is sent  -->
            <property name="errorThreshold" value="61440"/>
            <property name="checkType" value="Memory"/>
            <property name="notificationSubject" value="Agent"/>
            <property name="notifier">
                  <ref bean="harvestCoordinatorNotifier" />
            </property>
      </bean>
      <bean id="processorCheck"
            class="org.webcurator.core.check.ProcessorCheck"
abstract="false"
            singleton="true" lazy-init="default" autowire="default"
            dependency-check="default">
            <!-- the pattern and command properties can be set the
default setup is for linux sar -u -->
            <!--  for a solaris box use this pattern to get the
processor data for linux remove this line -->
            <property name="pattern"
value="(?m)\w+:\w+:\w+\s+\S+\s+\S+\s+\S+\s+(\S+)$"/>
            <!-- The minimum percentage of processor available before
a warning notification is sent  -->
            <property name="warnThreshold" value="30"/>
            <!-- The minimum percentage of processor available before
an error notification is sent  -->
            <property name="errorThreshold" value="20"/>
            <property name="checkType" value="Processor"/>
            <property name="notificationSubject" value="Agent"/>
            <property name="notifier">
                  <ref bean="harvestCoordinatorNotifier" />
            </property>
      </bean>
      <bean id="diskSpaceChecker"
            class="org.webcurator.core.check.DiskSpaceChecker"
abstract="false"
            singleton="true" lazy-init="default" autowire="default"
            dependency-check="default">
            <!-- the percentage of disk used before a warning
notification is sent -->
            <constructor-arg index="0" value="80"/>
            <!-- the percentage of disk used before an error
notification is sent -->
            <constructor-arg index="1" value="90"/>
            <property name="checkType" value="Disk Space"/>
            <property name="notificationSubject" value="Agent"/>
            <property name="notifier">
                  <ref bean="harvestCoordinatorNotifier" />
            </property>
            <!-- List of disk mounts to check -->
```

```
            <constructor-arg index="2">
                  <list>
                        <value>/</value>
                  </list>
            </constructor-arg>
</bean>
```

The three checker beans allow the Harvest Agent to monitor Disk, Processor and Memory. Each of the checkers are configurable to allow different alert and error thresholds. A Notification event will be sent on either the alert or error threshold being exceeded.

Web Curator Tool – SOAP Service Configuration

The /WEB-INF/server-config.wsdd

All three components have a server-config.wsdd file. This file is used by Apache Axis to configure the SOAP services used within the Web Curator Tool.

The only attribute that should be modified in the Axis configuration is the location of the temporary directory that Axis should use for attachments. Make sure that this directory exists and is accessible to the Apache Tomcat server.

```
<parameter name="attachments.Directory" value="/tmp/attach"/>
```

Graceful shutdown and restart

The system can be taken down manually or automatically for maintenance.

To shut down and restart the Core and the DAS, but leave the harvesters running (so that they can continue harvesting when the Core and DAS are unavailable), follow these steps:

1. Admin or script shuts down Tomcat on the server that hosts Core and DAS.
2. Admin or script shuts down Oracle.
3. Admin or script does backup or whatever. WCT Agents continue harvesting.
4. Admin or script starts Oracle.
5. Admin or script starts Tomcat.
6. WCT Harvest Agents re-register themselves with WCT Core, and then copy any completed harvests to DAS and notify Core.

To shut down everything including the harvest agents, then the procedure is:

1. Wait until harvest agents have no crawl jobs running and shut them down (either directly or Tomcat container). This can be best achieved by halting all Scheduled and Queued target instances using the 'Calendar' icon on the Harvester Configuration screen, and then waiting until the currently running jobs finish.
2. Admin shuts down Tomcat on the server that hosts Core and DAS.
3. Admin shuts down database.

Restart the system again in the reverse order.

Note that when you shut down a harvest agent, running jobs are lost (when the agent restarts it does not know how to restart the harvest. If you pause a harvest (or all the harvests) then it stays in a paused state on the harvest agent, and is similarly lost when you shut down.

Appendix A: Creating a truststore and importing a certificate

To create a truststore and import a certificate:

3 First export your public key from your Directory server.
- Refer to the documentation from your Directory server, in order to complete this task.
- If possible export the certificate as a binary file. We will assume your exported certificate is called mydirectorycert.der

4 Create a truststore and dummy key. Using the keytool provided with the java SDK:

```
keytool -genkey -dname "cn=dummy, ou=dummy, o=dummy, c=US"
-alias dummy -keypass dummy -keystore
/var/wctcore/ssl/wct.ts -storepass password
```

5 You need to import the X509 certificate for your directory server:

```
keytool -import -file mydirectorycert.der -keystore
/var/wctcore/ssl/wct.ts
```

Appendix B: The OMS archive adapter

The OMSArchive bean is only used for the National Library of New Zealand to archive files into their Object Management System. For all other implementations the more generic FileSystemArchive Bean should be used.

To enable the OMS Archive, set the **archive** property on the **arcDigitalAssetStoreService** bean to **omsArchive**.

```
<bean id="archive"
            class="org.webcurator.core.archive.oms.OMSArchive"
abstract="false"
            singleton="true" lazy-init="default" autowire="default"
            dependency-check="default">
            <property name="url">
                  <value>http://omsserver/oms/upload</value>
            </property>
            <property name="partSize">
                  <value>1000000</value>
            </property>
            <property name="ilsTapuhiFlag">
                  <value>RT_ILS</value>
            </property>
            <property name="collectionType">
                  <value>CT_EPB</value>
            </property>
            <property name="objectType">
                  <value>OT_WWW</value>
            </property>
            <property name="agencyResponsible">
                  <value>AR_NLNZ</value>
            </property>
            <property name="instanceRole">
                  <value>IRC_PM</value>
            </property>
            <property name="instanceCaptureSystem">
                  <value>CS_HER</value>
            </property>
            <property name="instanceType">
                  <value>IT_COM</value>
            </property>
            <property name="user_group">
                  <value>4</value>
            </property>
</bean>
```