Additional Tools Manual

This is a manual for the commandline tools bundled with the NetarchiveSuite software

Contents

Most of the tools have some functionality, that the NetarchiveSuite applications already have inside them. The tools are there in case the applications fail, and manual intervention is called for. The tools are here grouped based on their module adherence, and not their functionality.

Audience

The intended audience of this manual is system administrators who will be responsible for the actual setup of NetarchiveSuite as well as technical personnel responsible for the proper operation of NetarchiveSuite. Some familiarity with XML and Java is an advantage in understanding this manual.

Search manual

Download as pdf

Tools in the Common Module

Contents

dk.netarkivet.common.tools.ArcMerge

This tool takes as input several arcfiles, and inserts all the records of the input arcfiles into a single arcfile. This arcfile is currently dumped to stdout, but can of course be redirected to a file.

Usage

```
export CLASSPATH=/path/to/installdir/lib/dk.netarkivet.common.jar
java dk.netarkivet.common.tools.ArcMerge arcfile1.arc arcfile2.arc > resulting.arc
```

dk.netarkivet.common.tools.ArcWrap

This tool takes an input file, an URI for this file, and a mime-type for this file, and creates an ARC-file with only one ARCRecord, which has the input file as contents, and with the given mimetype and URI. The ARCfile is written to stdout, but can of course be redirected to a file.

usage

```
export CLASSPATH=/path/to/installdir/lib/dk.netarkivet.common.jar
java dk.netarkivet.common.tools.ArcWrap input_file uri mime-type > myarchive.arc
```

dk.netarkivet.common.tools.ExtractCDX

This tool generates CDX-entries for all ARC-records in the given ARC-files, and prints the CDX'es to stdout, but can of course be redirected to a file like we show below.

usage

```
export CLASSPATH=/path/to/installdir/lib/dk.netarkivet.common.jar
java dk.netarkivet.common.tools.ExtractCDX [arcfile]+ > output.cdx
```

dk.netarkivet.common.tools.WriteBytesToFile

This tool makes large files primarily for testing purposes. The tool needs two arguments, the number of bytes that must be written (nbytes), and the name of the file to write to. The number of bytes must be > chunksize (currently 10000000), otherwise 0 bytes will be written to the file.

usage

```
export CLASSPATH=/path/to/installdir/lib/dk.netarkivet.common.jar
java dk.netarkivet.common.tools.WriteBytesToFile nbytes filename
```

Tools in the Monitor Module

Contents

dk.netarkivet.monitor.tools.JMXProxy

This tool enables you to browse the logs from all your applications using jconsole. Note that you must run this tool on the machine where you are running the GUIApplication.

Usage

```
export CLASSPATH=/path/to/installdir/lib/dk.netarkivet.monitor.jar
java -Dcom.sun.management.jmxremote dk.netarkivet.monitor.tools.JMXProxy \
"dk.netarkivet.common.logging:*" &
jconsole
```

In the JConsole dialogbox you can now select "Local". Then in the main window the tab MBeans is chosen, and you can unfold **dk.netarkivet.common.logging** where MBeans for all the logs are represented.

Tools in the Harvester Module

Contents

dk.netarkivet.tools.harvester.CreateCDXMetadataFile (deprecated)

Given a specific jobID (e.g. 42), this tool can be used to create a metadata-1.arc containing the CDX-entries for all arc-files belonging to that job.

prequisites and arguments

You need to specify the repositoryclient used for accessing your archived-data. If you use the default client JMSArcRepositoryClient you also need to specify the archive replica you will use (defined by setting "settings.common.useReplicaId"), the environmentname, the applicationName, the applicationInstanceId. These can all be defined on the commandline as overrides to the default values, or defined in a local settings.xml file.

Needed jarfiles in the classpath: dk.netarkivet.harvester.jar, dk.netarkivet.archive.jar (if using default repositoryclient)

The tool only has one argument, the jobID

Sample usage of this tool

```
export INSTALLDIR=/home/test/netarchive
CLASSPATH=$INSTALLDIR/lib/dk.netarkivet.harvester.jar:
export CLASSPATH=$CLASSPATH:$INSTALLDIR/lib/dk.netarkivet.archive.jar
export OPTS=-Ddk.netarkivet.settings.file=localsettings.xml

java $OPTS dk.netarkivet.harvester.tools.CreateCDXMetadataFile 42
```

dk.netarkivet.harvester.tools.CreateLogsMetadataFile (deprecated)

In the beginning, the metadata-1.arc files did not include the Heritrix logs. This tool was made to allow us to make a metadata-2.arc file that contains the heritrix logs associated with a given job.

Consider this tool deprecated. For further information see the javadoc for this method. Note that settings file mentioned below need to contain proper values for the harvesting metadata settings:

```
<metadata>

<heritrixFilePattern>.*(\.xml|\.txt|\.log|\.out)</heritrixFilePattern>
                <reportFilePattern>.*-report.txt</reportFilePattern>
                <logFilePattern>.*(\.log|\.out)</logFilePattern>
            </metadata>
```

Sample usage of this tool

```
export INSTALLDIR=/home/test/netarchive
export CLASSPATH=$INSTALLDIR/lib/dk.netarkivet.harvester.jar
export OPTS=-Ddk.netarkivet.settings.file=localsettings.xml
java $OPTS dk.netarkivet.harvester.tools.CreateLogsMetadataFile \
jobid-harvestid.txt jobsdir
```

dk.netarkivet.harvester.tools.HarvestTemplateApplication

This tools enables you to create (create command), download (download command), update (update command) and show (showall command) the existing templates.

prequisites and arguments

You need to point to a settings file with connection information for your harvest database. In a standard NAS deployment, use the INSTALLDIR/conf/settings_GUIApplication.xml

Sample usage of this tool

```
export INSTALLDIR=/home/test/netarchive
export CLASSPATH=$INSTALLDIR/lib/dk.netarkivet.harvester.jar
export
OPTS=-Ddk.netarkivet.settings.file=$INSTALLDIR/conf/settings_GUIApplication.xml

java $OPTS dk.netarkivet.harvester.tools.HarvestTemplateApplication <command> <args>
```

The different <command> <args> possibilities:

1. `create <template-name> <xml-file for this template>`
2. `download [<template-name>]*`
3. `update <template-name> <xml-file to replace this template>`
4. `showall`

Note that with the download command, you can either download all templates in one go (with no args), or select the names of the templates to download (separated by space)

dk.netarkivet.harvester.tools.HarvestdatabaseUpdateApplication

This tools enables you to update the tables in the harvestdatasbase to the versions required this release of NetarchiveSuite. It should be run after installing the software, but before starting the NetarchiveSuite applications.

prequisites and arguments

You need to point to a settings file with connection information for your harvest database. In a standard NAS deployment, use the INSTALLDIR/conf/settings_GUIApplication.xml

And the harvest database needs to be running as well.

Sample usage of this tool

First, the harvestdatabase is started, if it isn't up and running already.
Then the update tool is executed:

```
export INSTALLDIR=/home/test/netarchive
export CLASSPATH=$INSTALLDIR/lib/dk.netarkivet.harvester.jar
java -Ddk.netarkivet.settings.file=$INSTALLDIR/conf/settings_GUIApplication.xml \
dk.netarkivet.harvester.tools.HarvestdatabaseUpdateApplication
```

Finally, the harvestdatabase is shutdown, if you're using derby as database.

Tools in the Archive Module

Contents

dk.netarkivet.archive.tools.ReestablishAdminDatabase

This is a tool which is only run when converting from a file based administration of ArcRepository to the database based administration of ArcRepository. It takes the admin.data file and enters its data into the database.

Prerequisites

You need to have the external database running, the admin.data file must exist, and the tool must be run from the installation directory of the installation.

Usage

```
java -Ddk.netarkivet.settings.file=conf/settings_ArcRepositoryApplication.xml \
dk.netarkivet.archive.tools.ReestablishAdminDatabase [admin.data]
```

The optional argument admin.data is the path to the admin.data file. As default it is assumed that it is called 'admin.data' and it is located in the directory where the tool is run. It is therefore only necessary if the admin.data is

in another directory or called by another name (e.g. backups/admin.data or admin.data.backup).

dk.netarkivet.archive.tools.CreateIndex

This tool forces the IndexServer to create indices preemptively. This tool can be used for retrieving logs and cdx'es for previously completed harvestjobs before they are actual needed. This can be helpful if you want to improve the time it takes to generate Deduplication indices.

Prerequisites

You need to have a IndexServerApplication online. If you use HTTP as file transport method, you probably also need to override the settings.common.remoteFile.port in order to avoid conflicts (In the example below, we have set the port number to 5000).

Furthermore all harvestjobs referred to in the CreateIndex commands must have metadata-1.arc files stored in the archive.

Usage

```
export INSTALLDIR=/fullpath/to/installdir
export CLASSPATH=$INSTALLDIR/lib/dk.netarkivet.archive.jar
export OPTS=-Dsettings.common.cacheDir=/tmp/cache \
-Dsettings.common.environmentName=QUICKSTART -Dsettings.common.remoteFile.port=5000
java $OPTS dk.netarkivet.archive.tools.CreateIndex -t dedup -l 1,2
ctrl-c
```

This requests a deduplication index based on the harvestjobs with id 1 and 2, and stores this index in /tmp/cache/DEDUP_CRAWL_LOG/1-2-cache

dk.netarkivet.archive.tools.GetFile

With this tool you can retrieve a file from your archive.

Prerequisites

If you want to use another arcrepositoryclient than the default (dk.netarkivet.archive.arcrepository.distribute.JMSArcRepositoryClient), you need to override the setting

```
settings.common.arcrepositoryClient.class
```

If you use the default, you need to set the environmentName correctly, so your ArcrepositoryApplication receives your GetFile request, and define your replicas, and the replicaId of the replica where you want to get the data. All this is most easily put into a local settings.xml:

```
<settings>
 <common>
    <environmentName>QUICKSTART</environmentName>
    <replicas>
            <replica>
                <replicaId>SH</replicaId>
                <replicaType>bitarchive</replicaType>
                <replicaName>SHB</replicaName>
            </replica>
        </replicas>
        <useReplicaId>SH</useReplicaId>
</common></settings>
```

In the setting.xml above, the environment name have been set to QUICKSTART, you only have a single replica with replicaId=SH, and the Id of the replica where you want to get the data is "SH".

Usage

```
export INSTALLDIR=/fullpath/to/installdir
export CLASSPATH=$INSTALLDIR/lib/dk.netarkivet.archive.jar
export SETTINGSFILE=/home/user/conf/settings_ArcRepositoryApplication.xml
export OPTS=-Ddk.netarkivet.settings.file=$SETTINGSFILE \
-Dsettings.common.remoteFile.port=5000
java $OPTS dk.netarkivet.archive.tools.GetFile 3-metadata-1.arc
```

If the file 3-metadata-1.arc exists in your SH replica, the file is downloaded from the archive, and written to the current working directory. If not, you are going to wait for a long time, until the arcrepository client times out. The tool has an optional second argument, which is a destination file:

```
export INSTALLDIR=/fullpath/to/installdir
export CLASSPATH=$INSTALLDIR/lib/dk.netarkivet.archive.jar
export SETTINGSFILE=/home/user/conf/settings_ArcRepositoryApplication.xml
export OPTS=-Ddk.netarkivet.settings.file=$SETTINGSFILE \
-Dsettings.common.remoteFile.port=5000
java $OPTS dk.netarkivet.archive.tools.GetFile 3-metadata-1.arc destination-file.arc
```

dk.netarkivet.archive.tools.Upload

The tool "dk.netarkivet.archive.tools.Upload" allows one to upload ARC files to a repository of your choice.

The type of arcrepository you are uploading your files to are defined by the setting

```
settings.common.arcrepositoryClient.class
```

, where the default is dk.netarkivet.archive.arcrepository.distribute.JMSArcRepositoryClient. This client uses JMS

messages to communicate with a repository.

Prerequisites

If you use the client dk.netarkivet.archive.arcrepository.distribute.JMSArcRepositoryClient, you need to ensure, that you send upload requests to the correct JMS queue, and that you receive the responses from the client. This is ensured by setting the setting

```
settings.common.environmentName
```

to the proper value (e.g. PROD or DEV). The same holds for the setting

```
settings.common.applicationName
```

(e.g. Upload), and finally "settings.common.applicationInstanceId" (e.g. ONE or TWO) If you intend to override any of the settings mentioned above, you can either do the overrides on the commandline or writing the overrides to a settings file.

Using the tool

This tool will upload a number of local files to all replicas in the archive. An example of an execution command is:

```
export SETTINGSFILENAME=settings_ArcRepositoryApplication.xml
java -Ddk.netarkivet.settings.file=/home/user/conf/$SETTINGSFILENAME \
        -cp lib/dk.netarkivet.archive.jar \
        dk.netarkivet.archive.tools.Upload \
        file1.arc [file2.arc ...]
```

where file1.arc file2.arc ... is the files to be uploaded

This will cause the files to be uploaded. The behaviour of the default client (JMSArcRepositoryClient) is furthermore, that if a file is uploaded successfully, it is deleted locally. This means that if there are files left after Upload is finished, these files are probably not stored safely.

dk.netarkivet.archive.tools.GetRecord

This tool takes a CDX based lucene-index, and an URI, and retrieves the corresponding ARC-record from the archive, and dumps it to stdout.

Prerequisites

The same as for getFile.

Usage

```
export INSTALLDIR=/fullpath/to/installdir
export CLASSPATH=$INSTALLDIR/lib/dk.netarkivet.archive.jar
export SETTINGSFILE=/home/user/conf/settings_ArcRepositoryApplication.xml
export LUCENE_INDEX=/tmp/cache/DEDUP_CRAWL_LOG/1-cache
export URI=http://www.netarkivet.dk
export OPTS=-Ddk.netarkivet.settings.file=$SETTINGSFILE \
-Dsettings.common.remoteFile.port=5000
java $OPTS dk.netarkivet.archive.tools.GetRecord $LUCENE_INDEX $URI
```

If the URI is not in the given index, an exception is sent to stdout with the message: **Resource missing in index or repository for URI**

TODO: Mention how to make an luceneindex for your stored arcfiles.

dk.netarkivet.archive.tools.RunBatch

The bitarchives are designed to receive batch-programs to run on all the arc-files stored in the bitarchive. This is true no matter whether the bitarchive is installed as a local arc-repository or a distributed repository with several bitarchives. Batch programs are also used internally by the NetarchiveSuite software to do specific tasks like getting a CDX'es for a specific job, or checksums of arc-files stored in the bitarchive, or lists of arc-files from the bitarchive.

The RunBatch program is used to send your own batchjobs to the bitarchives.

Note that a batchjob will only be sent to one bitarchive replica!

It is not possible to send batchjobs to checksum replicas, as only bitarchive replicas can handle batchjobs.

Prerequisites for running a batch job

A number of prerequisites must be taken care of before a batch job can be executed. These are:

- A **Settings file** must be present and must include declarations of at least the following setttings:
- Replicas to identify the replica you want to communicate with:
- **settings.common.replicas** in order for the batch program to identify and messages to the bitarchive.
- **settings.common.useReplicaId** in order to determine default bitarchive replica to use.

Channel settings to be able to make channel names to communicate with running system:

- **settings.common.environmentName** (typically PROD)
- **settings.common.applicationName** (RunBatchApplication, but currently set automatically)

Other settings related to communication where the running systems settings differs from default.

- **Batch program:** The batch program must be designed as a Java class that extend ARCBatchJob or FileBatchJob depending on whether you want to make a batch program over arc records or a batch program over files.
- **Call location:** The RunBatch program can be started from any of the machines in the distributed system where the system runs.
- **Disk space requirement on bitarchive:** The disk space needed will depend on the batch program concerned. As an example the ChecksumJob produces about 100 bytes per arc-file, whereas a batch program writing the full contents of arc-files would require as much space as the archive it self.

- **Class Path:** Running RunBatch requires **lib/dk.netarkivet.archive.jar** in the class path
- **Memory space on bitarchive:** The memory space needed will depend on the written batch program. If the batch program is written using a lot of jar files, these files will be needed to be kept in memory while the batch program is running, and on top of that comes the memory requirenments for the batch job it self.
- **Timeout on bitachive monitor:** To set an specific timeout for a concrete BatchJob, you need to override the variable 'protected long batchJobTimeout = -1;' in FileBatchJob.java. Otherwise the default timeout is 14 days.

Execution and Arguments

The execution of a batch program is done by calling the **dk.netarkivet.archive.tools.RunBatch** program with the following arguments:

If the batch program is given in a single class file, this must be specified in the parameter:

- **-C<classfile>** is a file containing a FileBatchJob/ARCBatchJob implementation
  If the batch program is given in one or more jar files, this must be specified in the parameters:
- **-N<className>** is the name of the primary class to be loaded and executed as a FileBatchJob/ARCBatchJob implementation
- **-J<jarfile>** is on or more files containing all the classes needed by the primary class. The files must be separated by commas.
  To specify which files the batch program must be executed on, the following parameters may be set optionally:
- **-B<replica>** is the name of the bitarchive replica which the batchjob must be executed on. The default is the name of the bitarchive replica identified by the setting **settings.common.useReplicaId**. Note that it is the replica name and not replica id which are refered to here. Also it cannot be the name of a checksum replica, since batchjob can only be executed on bitarchive replicas.
- **-R<regexp>** is a regular expression that will be matched against file names in the archive. The default is ***.*** which means it will be executed on all files in the bitarchive replica.
  To specify output files from the batch program, the following parameters may be set optionally
- **-O<outputfile>** is a file where the output from the batch job will be written. By default, it goes to **stdout**, but it will be mixed with other output to **stdout**.
- **-E<errorFile>** is a file where the errors from the batch job will be written. By default, it goes to stderr.
  An example of an execution command is:

```
export SETTINGSFILENAME=settings_ArcRepositoryApplication.xml
java -Ddk.netarkivet.settings.file=/home/user/conf/$SETTINGSFILENAME \
  -cp lib/dk.netarkivet.archive.jar \
   dk.netarkivet.archive.tools.RunBatch \
   -CFindMime.class -R10-*.arc -BReplicaOne -Oresfile
```

which will take in **lib/dk.netarkivet.archive.jar** in the class path and execute the general NetarchiveSuite program **dk.netarkivet.archive.tools.RunBatch** based on settings from file **/home/user/conf/settings_ArcRepositoryApplication.xml**. This will result in running the batch program **FindMime.class** on the bitarchive replica named **ReplicaOne**, but only on files with names matching the pattern

```
10-*.arc
```

The results written by the batch program is concatenated and placed in the output file named **resfile**.

Example of packing and executing a batch job

To package the files do the following:

```
jar -cvf batchfile.jar PATH/batchProgram.class
```

where **PATH** is the path to the directory where the batch class files are placed. This is under the **bin/** directory in the eclipse project. The **batchProgram.class** is the compiled file for your batch program.

The call to run this batch job is then:

```
export SETTINGSFILENAME=settings_ArcRepositoryApplication.xml
  java -Ddk.netarkivet.settings.file=conf/$SETTINGSFILENAME \
        -cp lib/dk.netarkivet.archive.jar \
       dk.netarkivet.archive.tools.RunBatch \
       -Jbatch.jar -Npath.batchProgram
```

where **path** in the **-N** argument has all **'/'** changed to **'.'**.

E.g. to run the batch job from the file **myBatchJobs/arc/MyArcBatchJob.java**, which inherits the ARCBatchJob class (**dk/netarkivet/common/utils/arc/ARCBatchJob**), do the following.

1. Place yourself in the bin/ folder under your project:

```
cd bin/
```

2. Package the compiled Java binaries into an .jar file:

```
jar -cvf batch.jar myBatchJobs/arc/-
```

3. Move the packaged batch job to your NetarchiveSuite directory.

```
mv batch.jar ~/NetarchiveSuite/
```

4. Run the following command to execute the batch job:

```
export SETTINGSFILENAME=settings_ArcRepositoryApplication.xml
   java -Ddk.netarkivet.settings.file=conf/$SETTINGSFILENAME \
        -cp lib/dk.netarkivet.archive.jar:lib/dk.netarkivet.common.jar
        dk.netarkivet.archive.tools.RunBatch -Jbatch.jar
-NmyBatchJobs.arc.MyArcBatchJob
```

The **lib/dk.netarkivet.common.jar** library need to be included in the classpath since the batch job (**myBatchJobs/arc/MyArcBatchJob**) inherits from a class within this library (**dk/netarkivet/common/utils/arc/ARCBatchJob**).

Security

If the security properties for the bitarchive (independent of this execution) are set as described in the Configuration Manual the batch program will not be allowed to:

- to write files to the bitarchive
- to change files in the bitarchive
- to delete files in the bitarchive

Tools in the Wayback module

Contents

Tools in Wayback Module

In addition to the tools described here, the NetarchiveSuite Java applications for continuous indexing of an arcrepository are described in the Configuration Manual.

dk.netarkivet.wayback.NetarchiveResourceStore

Wayback is a tool for browsing in webarchives. It can be downloaded from http://archive-access.sourceforge.net/projects/wayback/. The NetarchiveSuite plugin for wayback is a class `NetarchiveResourceStore` which implements `org.archive.wayback.ResourceStore`. `NetarchiveResourceStore` instantiates a connection to a NetarchiveSuite ArcRepository and retrieves archive data from it via NetarchiveSuite.
In order to make use of the plugin, it is necessary to. Copy the required jar files into the lib-directory of your wayback installation. Ensure that wayback has access to a NetarchiveSuite settings file with the necessary connection information. Configure wayback to use NetarchiveResourceStore
The lib directory for wayback will be under

```
wayback/WEB-INF/lib
```

in your tomcat webapps directory. Copy into the lib directory all the jar files in `netarchivesuite/lib` **except**

```
dk.netarkivet.deploy.jar, dk.netarkivet.harvester.jar, dk.netarkivet.viewerproxy.jar
,
dk.netarkivet.monitor.jar
```

and the jar files for the packages `wayback, je, jericho, jetty, junit, poi` and `libidn`. These are either not required or are already included in the wayback distribution.
The NetarchiveSuite settings file location can be specified in the catalina.sh file of your tomcat with a line like `CATALINA_OPTS="-Ddk.netarkivet.settings.file=/home/user/settings_for_my_repository.xml"`.
NetarchiveResourceStore has been tested with a wayback localcdxcollection using settings like:

```
<bean id="localcdxcollection" class="org.archive.wayback.webapp.WaybackCollection">
     <property name="resourceStore">       <bean
class="dk.netarkivet.wayback.NetarchiveResourceStore">       </bean>
 </property>

    <property name="resourceIndex">             <bean
class="org.archive.wayback.resourceindex.LocalResourceIndex">          <property
name="source">          <bean
class="org.archive.wayback.resourceindex.CompositeSearchResultSource">
<property name="CDXSources">              <list>
 <value>index1.cdx</value>                 <value>index2.cdex</value>
</list>            </property>          </bean>              </property>
<property name="maxRecords" value="40000" />      </bean>
    </property>  </bean>
```

but should work with other types of wayback collection.
There is an ant build file which can be used to repack the wayback war-file with the addition of the netarchivesuite plugin.Ant tasks to unpack and repack the wayback war-file are in wayback.build.xml and there are samples settings in "examples/wayback/*".

dk.netarkivet.wayback.batch.ExtractWaybackCDXBatchJob

This batch job is a wrapper for the parts of the wayback API which generate CDX index files for use in wayback. The job can be called with a script like

```
java \   -Ddk.netarkivet.settings.file=../../settings_wayback.xml \
 -Dsettings.common.applicationInstanceId=CDX_BATCH \   -cp
../lib/dk.netarkivet.archive.jar \   dk.netarkivet.archive.tools.RunBatch \
 -Ndk.netarkivet.wayback.ExtractWaybackCDXBatchJob \
 -J../lib/dk.netarkivet.wayback.jar,../lib/wayback-core-1.4.2.jar \
 -R1042-.*(?<metadata-[0-9]).arc \   -BMYREPLICA \   -Oout.cdx
```

Note the syntax of the regular expression which selects all arcfiles generated by job 1042 "except" for metadata arcfiles. The cdx filesgenerated are unsorted. For use in wayback they must be sorted and merged e.g. using unix sort:

```
export LC_ALL=C; sort --temporary-directory=/tmp 1.cdx 2.cdx 3.cdx 4.cdx >
sorted.cdx
```

Our experience is that sorting and merging of files with total size up to 100GB can be accomplished in a few hours on a moderately powerful server machine.

dk.netarkivet.wayback.batch.ExtractDeduplicateCDXBatchJob

In netarchivesuite, duplicate objects which are not harvested are recorded as extra metadata in the heritrix crawl log. In order to be able to browse these items, these deduplication records need to be indexed. Each deduplication record will generate a cdx record showing the harvested time as the time when the duplicate record was discovered, but pointing to the archive location where the original record is stored. The batch job to execute this indexing is invoked in exactly the same way as that described above for indexing the archived data, except that in this case we would use a regular expression which matches only metadata files, rather than one which matches everything except metadata files; for example

```
-R1042-'.*'metadata'.*'arc
```

As in the above case, the returned cdx files are unsorted.

dk.netarkivet.wayback.DeduplicateToCDXApplication

This is a command line interface to the same code for generating CDX indexes from deduplication records in crawl log files (not metadata arcfiles). It can be invoked by

```
java -cp dk.netarkivet.wayback.jar dk.netarkivet.wayback.DeduplicateToCDXApplication
crawl1.log crawl2.log crawl3.log > out.cdx
```

dk.netarkivet.wayback.accesscontrol.RegExpExclusionFilterFactory

This is an alternative to the class StaticMapExclusionFilterFactory supplied with wayback. The class is a spring bean which can be added to any wayback access point in wayback.xml with a specification like

```
<property name="exclusionFactory">
       <bean
class="dk.netarkivet.wayback.accesscontrol.RegExpExclusionFilterFactory"
init-method="init">
               <property name="file" value="/home/test/wayback_regexps.txt" />
       </bean>
</property>
```

The file wayback_regexps.txt is a plain-text file containing a list of Java regular expressions for url's which are to be blocked in the access point. The regular expressions are applied directly to the original harvested urls. The file is only read when wayback is initialised. If the file is changed then wayback needs to be reloaded, for example by restarting tomcat.

Appendix A - How To Do Examples

Contents

Install the QuickStart according to the Quick Start Manual, e.g. in /home/test/netarchive.

- Add some domains to harvest using the the ADMGUI e.g. netarkivet.dk, kb.dk, statsbiblioteket.dk
- Create and run a snapshot with a byte limit of 100.000
- Wait until the job is done
- Setup your browser for browsing and index your harvest job

```
cd /home/test/netarchive/scripts/simple_harvest/bitarchive1/filedir
export CLASSPATH=/home/test/netarchive/lib/dk.netarkivet.common.jar
ls
```

e.g.

Arc Merge:

```
export FILEONE=1-1-20090519073602-00000-dia-test-int-01.kb.dk.arc
export FILETWO=1-1-20090519073602-00001-dia-test-int-01.kb.dk.arc
java dk.netarkivet.common.tools.ArcMerge $FILEONE $FILETWO  > resulting.arc
```

Extract CDX:

```
export FILEONE=1-1-20090519083732-00002-dia-test-int-01.kb.dk.arc
java dk.netarkivet.common.tools.ExtractCDX $FILEONE > output.cdx
```

Get Record using Lucene:

```
#e.g. an URI from the harvest found in your "viewerproxy"
export URI=http://netarkivet.dk/index-da.php
cd /home/test/netarchive/scripts/simple_harvest/cache/fullcrawllogindex
cp -r 1-cache 1-cache.unzip
cd 1-cache.unzip/
ls
gunzip *
export SETTINGSFILE=/home/test/netarchive/scripts/simple_harvest/settings.xml
export LUCENE_INDEX=/home/test/netarchive/scripts/simple_harvest/cache/\
  fullcrawllogindex/1-cache.unzip
export OPTS=-Ddk.netarkivet.settings.file=$SETTINGSFILE \
      -Dsettings.common.remoteFile.port=5000
java $OPTS dk.netarkivet.archive.tools.GetRecord $LUCENE_INDEX $URI
```

Upload:

```
cd /home/test/netarchive/scripts/simple_harvest/
cp /home/test/netarchive/scripts/simple_harvest/bitarchive1/filedir/resulting.arc
new_resulting.arc
export SETTINGSFILE=/home/test/netarchive/scripts/simple_harvest/settings.xml
export OPTS=-Ddk.netarkivet.settings.file=$SETTINGSFILE
-Dsettings.common.remoteFile.port=5000
java $OPTS -cp /home/test/netarchive/lib/dk.netarkivet.archive.jar
dk.netarkivet.archive.tools.Upload new_resulting.arc
#just press <CTRL-C> to stop the job
```

Batch e.g. with checksum:

```
cd /home/test/netarchive
mkdir batchprogs

#copy example batchprog ChecksumJob.java to  batchprogs/.
cd batchprogs
javac ChecksumJob.java
export SETTINGSFILE=/home/test/netarchive/scripts/simple_harvest/settings.xml
java -cp lib/dk.netarkivet.archive.jar -Dsettings.common.remoteFile.port=5000 \
-Ddk.netarkivet.settings.file=$SETTINGSFILE dk.netarkivet.archive.tools.RunBatch \
-Cbatchprogs/ChecksumJob.class -Ooutput.checksum
```

ChecksumJob.java