

NetarchiveSuite Additional Tools Manual

Printer friendly version

Contents

1. 1. Introduction
 1. Contents
 2. Audience
2. Tools in the Common Module
 1. dk.netarkivet.common.tools.ArcMerge
 1. usage
 2. dk.netarkivet.common.tools.ArcWrap
 1. usage
 3. dk.netarkivet.common.tools.ExtractCDX
 1. usage
 4. dk.netarkivet.common.tools.WriteBytesToFile
 1. usage
3. Tools in the Monitor Module
 1. dk.netarkivet.monitor.tools.JMXProxy
 1. Usage
4. Tools in the Harvester Module
 1. dk.netarkivet.tools.harvester.CreateCDXMetadataFile (deprecated)
 1. prerequisites and arguments
 2. Sample usage of this tool
 2. dk.netarkivet.harvester.tools.CreateLogsMetadataFile (deprecated)
 1. Sample usage of this tool
5. Tools in the Archive Module
 1. dk.netarkivet.archive.tools.ReestablishAdminDatabase
 1. Prerequisites
 2. Usage
 2. dk.netarkivet.archive.tools.CreateIndex
 1. Prerequisites
 2. Usage
 3. dk.netarkivet.archive.tools.GetFile
 1. Prerequisites
 2. Usage
 4. dk.netarkivet.archive.tools.Upload
 1. Prerequisites
 2. Using the tool
 5. dk.netarkivet.archive.tools.GetRecord
 6. Prerequisites
 1. Usage
 7. dk.netarkivet.archive.tools.RunBatch
 1. Prerequisites for running a batch job
 2. Execution and Arguments
 3. Example of packing and executing a batch job
 4. Security
 5. Outstanding Issues
2. Appendix A - How To Do Examples

0.1. Introduction

edit

This manual describes how to use the additional standalone tools bundled with the NetarchiveSuite web archive software package.

0.1.1. Contents

Most of the tools have 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.

0.1.2. 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.

0.2. Tools in the Common Module

edit

0.2.1. 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.

0.2.1.1. usage

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

0.2.2. 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.

0.2.2.1. usage

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

0.2.3. 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.

0.2.3.1. usage

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

0.2.4. 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.

0.2.4.1. usage

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

0.3. Tools in the Monitor Module

edit

0.3.1. 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.

0.3.1.1. 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.

0.4. Tools in the Harvester Module

edit

0.4.1. 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.

0.4.1.1. prerequisites 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

0.4.1.2. Sample usage of this tool

```
export INSTALLDIR=/home/test/netarchive
export CLASSPATH=$INSTALLDIR/lib
/dk.netarkivet.harvester.jar:$INSTALLDIR
/lib/dk.netarkivet.archive.jar
java -Ddk.netarkivet.settings.file=localsettings.xml
dk.netarkivet.harvester.tools.CreateCDXMetadataFile 42
```

0.4.2. 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>
```

0.4.2.1. Sample usage of this tool

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

0.5. Tools in the Archive Module

edit

0.5.1. 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 puts it into the database.

0.5.1.1. 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.

0.5.1.2. Usage

```
java
-Ddk.netarkivet.settings.file=conf/settings_ArcRepositoryApplicatio
n.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).

0.5.2. dk.netarkivet.archive.tools.CreateIndex

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

0.5.2.1. 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.

0.5.2.2. Usage

```
export INSTALLDIR=/fullpath/to/installdir
export CLASSPATH=$INSTALLDIR/lib/dk.netarkivet.archive.jar
cd $INSTALLDIR
java -cp lib/dk.netarkivet.archive.jar
-Ddk.netarkivet.settings.file=conf/settings_ArcRepositoryApplicatio
n.xml -Dsettings.common.applicationInstanceId=INDEX
dk.netarkivet.archive.tools.CreateIndex -t dedup -l 2
```

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

0.5.3. dk.netarkivet.archive.tools.GetFile

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

0.5.3.1. 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 do 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".

0.5.3.2. Usage

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

0.5.4. 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.

0.5.4.1. 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.

0.5.4.2. 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:

```
java -Ddk.netarkivet.settings.file=/home/user/conf
/settings_ArcRepositoryApplication.xml \
-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 execution, these files are not known to be stored safely.

0.5.5. 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.

0.5.6. Prerequisites

The same as for getFile.

0.5.6.1. Usage

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

0.5.7. 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. Only bitarchive replicas can handle batchjobs.

0.5.7.1. Prerequisites for running a batch job

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

These are:

- *Setting*  `file:"`
 - must be present and must include declarations of at least the following settings:*
 - *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 requirements for the batch job it self.

- **Timeout on bitarchive monitor:**

To set an specific timeout for a concrete BatchJob, its needed to override 'protected long batchJobTimeout = -1;' in FileBatchJob.java. Otherwise the default timeout is 14 days.

0.5.7.2. 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 comma separated.*

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:

```
java -Ddk.netarkivet.settings.file=/home/user/conf
/settings_ArcRepositoryApplication.xml \
-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**.

0.5.7.3. 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:

```
java
```

```
-Ddk.netarkivet.settings.file=conf/settings_ArcRepositoryApplicatio
n.xml \
    -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/utills/arc/ARCBatchJob**), do the following.

- **cd bin/** - Place yourself in the **bin/** folder under your project.
- **jar -cvf batch.jar myBatchJobs/arc/*** - Package the compiled Java binaries into an **.jar** file.
- **mv batch.jar ~/NetarchiveSuite/.** - Move the packaged batch job to your **NetarchiveSuite** directory.
- **cd ~/NetarchiveSuite/** - Go to your **NetarchiveSuite** directory.
- Run the following command to execute the batch job:

```
java
-Ddk.netarkivet.settings.file=conf/settings_ArcRepositoryApplicatio
n.xml \
    -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/utills/arc/ARCBatchJob**).

0.5.7.4. 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

0.5.7.5. Outstanding Issues

As it will be described in archive assignment B.2.4, there are plans to make logging of internal logs and exceptions from batch jobs better. The only way to code internal logging today is to write to the output of the batch program.

If the batch job today is running at a time when not all bitarchives (i.e. BitarchiveApplications) are online, this will not be discovered by the NetarchiveSuite software. As described in archive assignment B.2.2 there are plans to change this at a later stage.

1. Appendix A - How To Do Examples

edit

Install the QuickStart according to https://netarchive.dk/suite/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:

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

Extract CDX:

```
java dk.netarkivet.common.tools.ExtractCDX 1-1-20090519083732-00002-dia-test-int-01.kb.dk.arc > 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 LUCENE_INDEX=/home/test/netarchive/scripts/simple_harvest
/cache/fullcrawllogindex/1-cache.unzip
java -Ddk.netarkivet.settings.file=$SETTINGSFILE
-Dsettings.common.remoteFile.port=5000
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
java -Ddk.netarkivet.settings.file=/home/test/netarchive/scripts
/simple_harvest/settings.xml
-Dsettings.common.remoteFile.port=5000 -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 attached example batchprog ChecksumJob.java to batchprogs/.
cd batchprogs
javac ChecksumJob.java
java -cp lib/dk.netarkivet.archive.jar
-Dsettings.common.remoteFile.port=5000
-Ddk.netarkivet.settings.file=/home/test/netarchive/scripts
/simple_harvest/settings.xml dk.netarkivet.archive.tools.RunBatch
-Cbatchprogs/ChecksumJob.class -Output.checksum
```

 *ChecksumJob.java*