# Grouper Book - Running in production

## Running Grouper-loader daemon:

The Grouper-loader daemon is not absolutely required for running Grouper, but you will benefit greatly if you do run it all the time. The benefits include:

- Daily report
- Change log processing to process entries entered into temporary tables
- Flattened membership processing

Grouper-loader is also required if you will be loading groups from an external database (covered later), or using change log clients. It can be run in an interactive session using :

Linux: Note, on linux, you will run this in the background as a long running process:

```
nohup GROUPER_HOME/bin/gsh.sh -loader &
```

Windows:

```
GROUPER_HOME/bin/gsh.bat -loader (windows)
```

Its is not always convenient to keep the loader running in an interactive session, as the session may be ended on an event such as a logout. Using something like GNU screen on Linux makes it easier to maintain a persistent session, but it is better to run it as daemon started at boot (for linux), or as a windows service (for windows). Fortunately this is a straightforward task using JavaWrapper from http://wrapper.tanukisoftware.com

## Running grouper-loader as a daemon (Linux) or service (Windows)

See this wiki for linux

~~Download the correct version for your system http://wrapper.tanukisoftware.com/doc/english/download.jsp – I'll be using the community edition version 3.5.6 for the example here. The standard and professional editions have additional features, such as monitoring the running java process, but the community edition is enough for our purposes. This is available for numerous operating systems, but not for 64bith Windows, which is only available in the standard and professional editions.~~

~~Extract the contents of the archive you downloaded. I prefer to keep it separate from my grouper installation as far as possible, and I normally put it alongside grouper. On Linux if my GROUPER_HOME on linux is /usr/local/grouper/grouper.api-1.6.2, I would put the wrapper files in /usr/local/grouper.api-1.6.2/wrapper-linux-x86-64-3.5.6. On windows I would put it in c:\program files\grouper.api-1.6.2\wrapper-windows-x86-64-3.5.6. I will refer to this as WRAPPER_HOME.~~

~~Here is the user-editable section of a sample configuration file required for Linux which sits in the WRAPPER_HOME/bin directory, called grouper. It stops at the before the line "# Do not modify anything beyond this point ". The remainder of the file content is required, but not included here. This file is the init script for a linux system:~~

### ~~A Linux startup script~~

```
#! /bin/sh

#
# Copyright (c) 1999, 2010 Tanuki Software, Ltd.
# http://www.tanukisoftware.com
# All rights reserved.
#
# This software is the proprietary information of Tanuki Software.
# You shall use it only in accordance with the terms of the
# license agreement you entered into with Tanuki Software.
# http://wrapper.tanukisoftware.com/doc/english/licenseOverview.html
#
# Java Service Wrapper sh script.  Suitable for starting and stopping
#  wrapped Java applications on UNIX platforms.
#

#-----------------------------------------------------------------------------
# These settings can be modified to fit the needs of your application
# Optimized for use with version 3.5.6 of the Wrapper.
```

```
# Application
APP_NAME="grouper"
APP_LONG_NAME="Grouper"

# Wrapper
WRAPPER_CMD="/usr/local/grouper/wrapper-linux-x86-32-3.5.6/bin/wrapper"
WRAPPER_CONF="/usr/local/grouper/wrapper-linux-x86-32-3.5.6/conf/grouper.conf"

# Priority at which to run the wrapper.  See "man nice" for valid priorities.
#  nice is only used if a priority is specified.
PRIORITY=

# Location of the pid file.
PIDDIR="/usr/local/grouper/wrapper-linux-x86-32-3.5.6/bin/"

# FIXED_COMMAND tells the script to use a hard coded action rather than
# expecting the first parameter of the command line to be the command.
# By default the command will will be expected to be the first parameter.
#FIXED_COMMAND=console

# PASS_THROUGH tells the script to pass all arguments through to the JVM
#  as is.  If FIXED_COMMAND is specified then all arguments will be passed.
#  If not set then all arguments starting with the second will be passed.
#PASS_THROUGH=true

# If uncommented, causes the Wrapper to be shutdown using an anchor file.
#  When launched with the 'start' command, it will also ignore all INT and
#  TERM signals.
#IGNORE_SIGNALS=true

# Wrapper will start the JVM asynchronously. Your application may have some
#  initialization tasks and it may be desirable to wait a few seconds
#  before returning.  For example, to delay the invocation of following
#  startup scripts.  Setting WAIT_AFTER_STARTUP to a positive number will
#  cause the start command to delay for the indicated period of time
#  (in seconds).
#
WAIT_AFTER_STARTUP=0

# If set, wait for the wrapper to report that the daemon has started
WAIT_FOR_STARTED_STATUS=true
WAIT_FOR_STARTED_TIMEOUT=120

# If set, the status, start_msg and stop_msg commands will print out detailed
#   state information on the Wrapper and Java processes.
#DETAIL_STATUS=true

# If set, the 'pause' and 'resume' commands will be enabled.  These make it
#  possible to pause the JVM or Java application without completely stopping
#  the Wrapper.  See the wrapper.pausable and wrapper.pausable.stop_jvm
#  properties for more information.
#PAUSABLE=true

# If specified, the Wrapper will be run as the specified user.
# IMPORTANT - Make sure that the user has the required privileges to write
#  the PID file and wrapper.log files.  Failure to be able to write the log
#  file will cause the Wrapper to exit without any way to write out an error
#  message.
# NOTE - This will set the user which is used to run the Wrapper as well as
#  the JVM and is not useful in situations where a privileged resource or
#  port needs to be allocated prior to the user being changed.
RUN_AS_USER=grouper

# By default we show a detailed usage block.  Uncomment to show brief usage.
#BRIEF_USAGE=true

# When installing on On Mac OSX platforms, the following domain will be used to
#  prefix the plist file name.
PLIST_DOMAIN=org.tanukisoftware.wrapper
```

```
# The following two lines are used by the chkconfig command. Change as is
#  appropriate for your application.  They should remain commented.
# chkconfig: 2345 20 80
# description: Test Wrapper Sample Application

# Initialization block for the install_initd and remove_initd scripts used by
#  SUSE linux distributions.
### BEGIN INIT INFO
# Provides: testwrapper
# Required-Start: $local_fs $network $syslog
# Should-Start:
# Required-Stop:
# Default-Start: 2 3 4 5
# Default-Stop: 0 1 6
# Short-Description: Grouper
# Description: Grouper loader daemon
### END INIT INFO
```

~~The variables "WRAPPER_COMMAND" and "WRAPPER_CONF" contain the full paths to files in my installation. WRAPPER command points to an executable file included in the wrapper distribution. WRAPPER_CONF points to a config file for loading Grouper. I set Grouper to start as user grouper with the line "RUN_AS_USER=grouper". This user needs rights in the grouper directory, the wrapper directory and to run java. On a standard distribution you won't need to do anything for Java, but will need to change the owner of GROUPER_HOME, WRAPPER_HOME and all their sub-directories:~~

```
chown -r grouper $GROUPER_HOME
chown -r grouper $WRAPPER_HOME
```

~~For testing, you may prefer simply to run as root, which you can do by simply commenting out the "RUN_AS_USER=grouper" line.~~

## A grouper.conf file for Linux

Here is a sample configuration file for the grouper application, which sits in WRAPPER_HOME/conf and is called grouper.conf. It is referenced in the WRAPPER_HOME/bin/grouper file as GROUPER_CONF. Please note that GROUPER_HOME, WRAPPER_HOME and JAVA_HOME are defined as a system variables in both the Linux and Windows versions.

```
# Include file problems can be debugged by removing the first '#'
#  from the following line:
##include.debug


#*****************************************************************
# Wrapper Localization
#*****************************************************************
# Specify the locale which the Wrapper should use.  By default the system
#  locale is used.
#wrapper.lang=en_US # en_US or ja_JP

# Specify the location of the Wrapper's language resources.  If these are
#  missing, the Wrapper will default to the en_US locale.
wrapper.lang.folder=../lang

#*****************************************************************
# Wrapper Java Properties
#*****************************************************************
# Java Application
#  Locate the java binary on the system PATH:
wrapper.java.command=$JAVA_HOME/bin/java

# Tell the Wrapper to log the full generated Java command line.
wrapper.java.command.loglevel=INFO

# Java Main class.  This class must implement the WrapperListener interface
#  or guarantee that the WrapperManager class is initialized.  Helper
#  classes are provided to do this for you.  See the Integration section
#  of the documentation for details.
wrapper.java.mainclass=org.tanukisoftware.wrapper.WrapperSimpleApp
# Java Classpath (include wrapper.jar)  Add class path elements as
```

```
#  needed starting from 1
wrapper.java.classpath.1=$GROUPER_HOME/lib/wrapperdemo.jar
wrapper.java.classpath.2=$WRAPPER_HOME/lib/*
wrapper.java.classpath.3=$GROUPER_HOME/lib/grouper/*
wrapper.java.classpath.4=$GROUPER_HOME/lib/jdbcSamples/*
wrapper.java.classpath.5=$GROUPER_HOME1/lib/custom/*
wrapper.java.classpath.6=$GROUPER_HOME/lib/ant/*
wrapper.java.classpath.7=$GROUPER_HOME/dist/lib/*
# this is the path to the grouper config files
wrapper.java.classpath.8=$GROUPER_HOME/conf/
# Java Library Path (location of Wrapper.DLL or libwrapper.so)
wrapper.java.library.path.1=$WRAPPER_HOME/lib
# Java Bits.  On applicable platforms, tells the JVM to run in 32 or 64-bit mode.
wrapper.java.additional.auto_bits=TRUE

# Java Additional Parameters
wrapper.java.additional.1=

# Initial Java Heap Size (in MB)
#wrapper.java.initmemory=3

# Maximum Java Heap Size (in MB)
#wrapper.java.maxmemory=64

# Application parameters.  Add parameters as needed starting from 1
# This line tells java wrapper to start the GrouperLoader class using the main method
wrapper.app.parameter.1=edu.internet2.middleware.grouper.app.loader.GrouperLoader

#*********************************************************************
# Wrapper Logging Properties
#*********************************************************************
# Enables Debug output from the Wrapper.
# wrapper.debug=TRUE

# Format of output for the console.  (See docs for formats)
wrapper.console.format=PM

# Log Level for console output.  (See docs for log levels)
wrapper.console.loglevel=INFO

# Log file to use for wrapper output logging.
wrapper.logfile=$GROUPER_HOME/wrapper.log

# Format of output for the log file.  (See docs for formats)
wrapper.logfile.format=LPTM

# Log Level for log file output.  (See docs for log levels)
wrapper.logfile.loglevel=INFO

# Maximum size that the log file will be allowed to grow to before
#  the log is rolled. Size is specified in bytes.  The default value
#  of 0, disables log rolling.  May abbreviate with the 'k' (kb) or
#  'm' (mb) suffix.  For example: 10m = 10 megabytes.
wrapper.logfile.maxsize=10m

# Maximum number of rolled log files which will be allowed before old
#  files are deleted.  The default value of 0 implies no limit.
wrapper.logfile.maxfiles=10

# Log Level for sys/event log output.  (See docs for log levels)
wrapper.syslog.loglevel=NONE

#*********************************************************************
# Wrapper General Properties
#*********************************************************************
# Allow for the use of non-contiguous numbered properties
wrapper.ignore_sequence_gaps=TRUE

# Title to use when running as a console
wrapper.console.title=Grouper
```

```
#*********************************************************************
# Wrapper JVM Checks
#*********************************************************************
# Detect DeadLocked Threads in the JVM. (Requires Standard Edition)
wrapper.check.deadlock=TRUE
wrapper.check.deadlock.interval=10
wrapper.max_failed_invocations=99
wrapper.console.fatal_to_stderr=FALSE
wrapper.console.error_to_stderr=FALSE
wrapper.check.deadlock.action=RESTART
wrapper.check.deadlock.output=FULL

# Out Of Memory detection.
# (Simple match)
wrapper.filter.trigger.1000=java.lang.OutOfMemoryError
# (Only match text in stack traces if -XX:+PrintClassHistogram is being used.)
#wrapper.filter.trigger.1000=Exception in thread "*" java.lang.OutOfMemoryError
#wrapper.filter.allow_wildcards.1000=TRUE
wrapper.filter.action.1000=RESTART
wrapper.filter.message.1000=The JVM has run out of memory.
```

## Grouper.conf file - Windows version

On Windows no startup script is required, only a configuration file is needed. I call it grouper.conf and put it in WRAPPER_HOME/conf.

```
# Include file problems can be debugged by removing the first '#'
#  from the following line:
##include.debug


#*********************************************************************
# Wrapper Localization
#*********************************************************************
# Specify the locale which the Wrapper should use.  By default the system
#  locale is used.
#wrapper.lang=en_US # en_US or ja_JP

# Specify the location of the Wrapper's language resources.  If these are
#  missing, the Wrapper will default to the en_US locale.
wrapper.lang.folder=../lang

#*********************************************************************
# Wrapper Java Properties
#*********************************************************************
# Java Application
#  Locate the java binary on the system PATH:
wrapper.java.command=%JAVA_HOME%\bin\java

# Tell the Wrapper to log the full generated Java command line.
wrapper.java.command.loglevel=INFO

# Java Main class.  This class must implement the WrapperListener interface
#  or guarantee that the WrapperManager class is initialized.  Helper
#  classes are provided to do this for you.  See the Integration section
#  of the documentation for details.
wrapper.java.mainclass=org.tanukisoftware.wrapper.WrapperSimpleApp
# Java Classpath (include wrapper.jar)  Add class path elements as
#  needed starting from 1
wrapper.java.classpath.1=%GROUPER_HOME%\lib\wrapperdemo.jar
wrapper.java.classpath.2=%WRAPPER_HOME%\lib\*
wrapper.java.classpath.3=%GROUPER_HOME%\lib\grouper\*
wrapper.java.classpath.4=%GROUPER_HOME%\lib\jdbcSamples\*
wrapper.java.classpath.5=%GROUPER_HOME%\lib\custom\*
wrapper.java.classpath.6=%GROUPER_HOME%\lib\ant\*
wrapper.java.classpath.7=%GROUPER_HOME%\dist\lib\*
# this is the path to the grouper config files
wrapper.java.classpath.8=%GROUPER_HOME%\conf\
# Java Library Path (location of Wrapper.DLL or libwrapper.so)
```

```
wrapper.java.library.path.1=%WRAPPER_HOME%\lib
# Java Bits.  On applicable platforms, tells the JVM to run in 32 or 64-bit mode.
wrapper.java.additional.auto_bits=TRUE

# Java Additional Parameters
wrapper.java.additional.1=

# Initial Java Heap Size (in MB)
#wrapper.java.initmemory=3

# Maximum Java Heap Size (in MB)
#wrapper.java.maxmemory=64

# Application parameters.  Add parameters as needed starting from 1
# This line tells java wrapper to start the GrouperLoader class using the main method
wrapper.app.parameter.1=edu.internet2.middleware.grouper.app.loader.GrouperLoader

#*********************************************************************
# Wrapper Logging Properties
#*********************************************************************
# Enables Debug output from the Wrapper.
# wrapper.debug=TRUE

# Format of output for the console.  (See docs for formats)
wrapper.console.format=PM

# Log Level for console output.  (See docs for log levels)
wrapper.console.loglevel=INFO

# Log file to use for wrapper output logging.
wrapper.logfile=%GROUPER_HOME%\wrapper.log

# Format of output for the log file.  (See docs for formats)
wrapper.logfile.format=LPTM

# Log Level for log file output.  (See docs for log levels)
wrapper.logfile.loglevel=INFO

# Maximum size that the log file will be allowed to grow to before
#  the log is rolled. Size is specified in bytes.  The default value
#  of 0, disables log rolling.  May abbreviate with the 'k' (kb) or
#  'm' (mb) suffix.  For example: 10m = 10 megabytes.
wrapper.logfile.maxsize=10m

# Maximum number of rolled log files which will be allowed before old
#  files are deleted.  The default value of 0 implies no limit.
wrapper.logfile.maxfiles=10

# Log Level for sys/event log output.  (See docs for log levels)
wrapper.syslog.loglevel=NONE

#*********************************************************************
# Wrapper General Properties
#*********************************************************************
# Allow for the use of non-contiguous numbered properties
wrapper.ignore_sequence_gaps=TRUE

# Title to use when running as a console
wrapper.console.title=Grouper

#*********************************************************************
# Wrapper JVM Checks
#*********************************************************************
# Detect DeadLocked Threads in the JVM. (Requires Standard Edition)
wrapper.check.deadlock=TRUE
wrapper.check.deadlock.interval=10
wrapper.max_failed_invocations=99
wrapper.console.fatal_to_stderr=FALSE
wrapper.console.error_to_stderr=FALSE
wrapper.check.deadlock.action=RESTART
wrapper.check.deadlock.output=FULL
```

```
# Out Of Memory detection.
# (Simple match)
wrapper.filter.trigger.1000=java.lang.OutOfMemoryError
# (Only match text in stack traces if -XX:+PrintClassHistogram is being used.)
#wrapper.filter.trigger.1000=Exception in thread "*" java.lang.OutOfMemoryError
#wrapper.filter.allow_wildcards.1000=TRUE
wrapper.filter.action.1000=RESTART
wrapper.filter.message.1000=The JVM has run out of memory.
```

## Testing the wrapper config

It is a good idea to do an initial application start in an interactive command line session as then any errors will be printed out onto the screen:

Linux:

```
WRAPPER_HOME/bin/grouper console
```

Windows:

```
WRAPPER_HOME\bin\wrapper.exe -c ..\conf\grouper.conf
```

## Running as a daemon in Linux

Symlink the script so that the link exists in /etc/init.d/ with:

```
ln -s WRAPPER_HOME/bin/grouper /etc/init.d/grouper
```

Alternatively you can copy the file to the /etc/init.d directory.

Next install it into your system startup with:

```
chkconfig -add grouper
```

.. and start it with:

```
/etc/init.d/grouper start
```

## Install and run as a service in Windows

A single command installs the service:

```
WRAPPER_HOME\bin\wrapper.exe -i ..\conf\grouper.conf
```

.. which you can then control as a service as normal.