Job Scheduler Daemon Configuration Guide

A component of Mark Dickinsons Unix Job Scheduler

This manual covers the server daemon component of Mark Dickinsons unix Job Scheduler.
This manual is for version V1.13 of Marks Job Scheduler

Other reference material available

jobsched_cmd User guide

Job Scheduler Messages Manual

Job Scheduler Security Manual

Job Scheduler Utilities Manual (jobsched_take_snapshot/jobutil)

GPLV2 license is at http://www.gnu.org/licenses/gpl-2.0.html

Tested under Linux (Fedora, CentOS), and Solaris (2.8, 10 and OpenSolaris)

Table of Contents

1. Who is this manual for

This manual is for experienced system administrators who are comforable working on the command line and maintaing system startup and shutdown scripts.

It covers how to get the job scheduler daemon up and running. It does not cover how to then go through and define users and jobs; that is the responsibility of the security and batch management teams who will likely never read this manual so that information is not needed in this manual.

The manual also leaves how to actually start the job scheduler to well into the manual, you should read through the manual to understand what the directories are, and what customisations you can make prior to starting the job scheduler; rather than just starting it and getting into a mess.

2. The Job Scheduler Daemon, what is it

The Job Scheduler Daemon program jobsched_daemon is the actual workhorse of the Job Scheduler application. It is (or should be) a permanently running daemon task that should be started and shutdown by system startup and shutdown scripts.

All communication to the job scheduler is via a tcpip connection to the server. You should read the security considerations section on how to lock that down.

As expected one of it's functions is to manage the scheduling and running of scheduled batch jobs. It does an awfull lot more though, it is the central component of the application. Some of its functions are

- manage the scheduling of batch jobs, including job and file dependency
- checking, managing failed jobs and restarts, alert forwarding for failed jobs etc
- the central (and only) point to manage all databases; including the job, calendar and user databases
- responding to interactive user session requests
- and more

3. Installation

The installation of the application should be done using the supplied installation ***jobsched_install_binary.sh*** script provided in the application package file, that is the only supported method, and the easiest.

The installation script will

- install the application into the directory you select
- attempt to recompile the application from source (if you have the source installation package; if you have the binary only package the **Linux** binaries will have been installed as part of the application install)
- prompt for interface and port number information
- customise configuration and startup scripts based upon parameters you provided
- optionally (recommended) create the init.d script and symbolic links needed for automatic startup and shutdown.
- optionally start the job scheduler
- optionally create test jobs under a userid you select

4. Configuration Information

4.1 Minimum Directories Needed

If you installed the Job Scheduler using the supplied installation script you will find a lot more directories than these (doc, html, samples etc) but these are the minimum directories required for the application to run, <approot> is where you installed the app, ie: /opt/job_scheduler.

approot/
approot/bin - main application binaries
approot/etc - config info used by some scripts
approot/joblogs - all batch job output goes here
approot/logs - daily (datestamped) server activity logfiles

4.2 Server Activity Logs, where are they (not changeable)

All server activity logs are stored in the directory "logs" ***which is expected to be under the directory you were in which the jobsched_daemon task was started***.

This directory will contain server activity logs of the form server_YYYYMMDD.log

The logs will contain a variety of information depending upon what logging level you have configured the job scheduler to use, at a minimum it should show all job start/stop history. Any other information that may be displayed is dependent upon your job scheduler server settings (loglevel info|warn|error and debugging levels etc that may be set).

The supplied init.d script handles being in the correct directory at the correct time during startup, if you are writing your own startup script ensure you have cd'ed to the correct directory before starting the job scheduler or results may be unpredictable.

4.3 Job Logs, where are they (not changeable)

All job output logs are stored in the directory “joblogs” ***which is expected to be under the directory you were in which the jobsched_daemon task was started***.

This directory will contain server activity logs of the form JOBNAME.log, for example a jobname of TEST-JOB will **append** any output produces to STDERR or STDOUT to file joblogs/TEST-JOB.log

The joblogs are appended to, so you don't lose any information from prior job runs.

This solution was chosen in preference to the way cron emails output from cron jobs if any is produced, as
(a) my batch jobs can now be as verbose as I want
(b) I prefer fixing problems from being able to review logs than having to check emails
(c) logs provide a better acrhivable audit trail
Anyway, that is the solution I have chosen and implemented, you don't have a choice.

The supplied init.d script handles being in the correct directory at the correct time during startup, if you are writing your own startup script ensure you have cd'ed to the correct directory before starting the job scheduler or results may be unpredictable.

4.4 Customising Alert Message Text

When a job fails and goes into an alert state the result code (exit code) from the script is one of the alert description fields. As the job scheduler does know the value of these result codes a limited form of alert text customisation has been provided.

As most 'unplanned' scripts generally use a simple 'exit 1' when something goes wrong it's not much use to them, but if you are writing new batch scripts from scratch the job scheduler will allow customisation of alert test for exit codes in the range 150-199 which should be more than enough for most sites; simply because the alert text fiels is minimal and this facility is more designed to provide a brief instruction rather than a detailed problem description.

The alert customisation file is a simple text file named alerts_custom.txt which must be in the directory you started the jobsched_daemon task from.
This example should explain it's use better than a few more paragraphs here

```
[mark@xxxxx/]$ cat /opt/dic*/job*/alerts_custom.txt
# alerts_custom.txt, comments at the end
#
# callout list
150 3 CALLOUT SYSTEM ADMINISTATOR
151 3 CALLOUT SQL DBA GROUP
152 3 CALLOUT DEVELOPMENT SUPPORT
#
# operator actions
160 1 LEAVE FOR DAYSHIFT
161 1 SCHED DELETE JOB
162 2 OPERATOR TO CORRECT AND RESTART
163 2 IGNORE, AUTOMATION HANDLING THIS
#
# --- alerts_custom.txt ---
# This file contains a set of user customisable messages for any job
# alerts. The user range is limited to 150-199.
# If any user job exists a program or script with an exit code in the
# 150-199 range this file will be searched for the appropriate error
# text value to place in the alert.
# Syntax: nnn S message-text
#   nnn must be three disgits followed by a space, is the exit code from the script
#   S is severity, 1=manual action needed, 2=rerun needed, 3=page support
#   message-text limited to 40 bytes
# ----------------------- end of file --------------------------
```

4.5 Internal customisation options

These settings must be set within the running job scheduler, they are discussed here as you should review these prior to starting the job scheduler for the first time so that you know what settings are available.

4.5.1 Log Level

Controls the amount of information logged by the server. The default level is INFO. It may be altered to ERROR, WARN or INFO. These are changed dynamically using the jobsched_cmd interface.

4.5.2 Debug Levels

These control how much debugging information is logged. It can be set to levels from 0 to 9, either

globally for all server modules, or on specifically selected server modules for focussed debugging. The default is 0 (no debugging information). This should only be changed at the request of support staff. These are changed dynamically using the jobsched_cmd interface.

4.5.3 New Day time

The scheduler newday time is the time the scheduler performs its database maintenance. This should be set to a time when you expect the current days batch run to be totally completed.

The new day job will not run if there are still jobs for the current day waiting to be processed.

If there are still jobs waiting to be processed at the time the new day job is scheduled to run one of two things will happen, depnding on how you have the newday action flag configured.

generate an alert and go into a failed state. The job must be manually restarted when all jobs for the current day have completed.

add a dependency onto the newday job of one of the jobs still waiting to run. This allows the newday job to schedule back on automatically after the job its waiting on has completed.

The new day time and new day fail/requeue actions may be changed dynamically using the jobsched_cmd interface.

4.5.4 Catchup Processing

The scheduler allows a facility called 'CATCHUP' where if the server has been shutdown for a few days on restart it will run the scheduler newday to schedule and run jobs that were due to be run on all of the days missed while the server was down. This is turned on by default. It may be turned off by jobsched_cmd.

When defining individual jobs the job record itself can be set to ignore scheduler newday catchup processing (ie: you don't want a job to delete files over 3days old running N times, just once when the server restarts will be fine for jobs like that).

It is IMPORTANT to note that catchup processing is for the scheduler newday processing task. It IS NOT intended to repeatedly run jobs that are behind on their schedule date.

For example it operations have been deleting a job from the scheduler queue before it runs for three days, the jobs next rundate will be three days in the past. If the server was shutdown for two days (jobs date now 5 days in past) and catchup is on, when the server is restarted it will run the scheduler newday for the two missed days, so the job will still run only twice and remain 3 days in the past. THIS IS NOT A BUG. Jobs should only be manually deleted off the active scheduler if there is an intent to manually either re-submit them or to delete them from the job database; this is clearly covered in the documentation provided for the jobsched_cmd interface.

4.5.5 Forwarding alerts to external monitoring tools

It is possible to forward alerts to external applications. This is done using the external alert forwarding interface to run a script of your choice when an alert occurs, and a script of your choice when an alert ends.

If you configure this alerts can be raised and canceled automatically in any external alert manager you may already be using, should you wish to do so. I of course wished to do so which is why the facility exists :-).

This can only be enabled or disabled using the jobsched_cmd program supplied with this application. To enable this facility when using jobsched_cmd and logged into the job scheduler server as a user with admin authority...

SCHED ALERT FORWARDING EXECRAISECMD /full/program/path

SCHED ALERT FORWARDING EXECCANCELCMD /full/program/path

SCHED ALERT FORWARDING ACTION EXTERNALCMD

To disable it again...

SCHED ALERT FORWARDING ACTION OFF

Both the raise alert script and end alert script executed must handle two parameters. The jobname as parameter one and a descriptive text field as message text in parameter two.

5. How the job scheduler runs jobs

5.1 Execution shell

5.1.1 Linux

The execution shell for the this release is /bin/bash. This is not currently modifiable. If you do not have a /bin/bash create a symbolic link named /bin/bash to another shell.

5.1.2 Solaris

The execution shell for the this release is /bin/ksh. This is not currently modifiable. If you do not have a /bin/ksh create a symbolic link named /bin/ksh to another shell.

5.2 Job Output Management

5.2.1 Normal Jobnames

You must have a directory called "**joblogs**" under the scheduler directory. If this directory does not exist execution of all jobs will fail.

Each job run has its output piped with the standard unix >> pipe to a file called joblogs*jobname*.log under the startup directory.

For example a job called TEST-JOB will have the output piped to the file joblogs\TEST-JOB.log .

IMPORTANT NOTE: using the >> concatenates to the end of existing files which is great for viewing job histories, BUT YOU WILL NEED to consider a process of cleaning these files up periodically. See the NULL job type below.

5.2.2 The NULL reserved jobname type

Jobs beginning with the special NULL- job prefix will log job output to /dev/null. They also have another special attribute which is that thay can be guaranteed to be the only job running on the scheduler at the time they execute.

This job type was added specifically so that the scheduler could run a NULL-xx job knowing that there is absolutely no job scheduler triggered activity within the joblogs directory; and archive off the logs and do cleanups.

How you decide to achive off old logs and clean then up is up to you, but there is this method of using the job scheduler to do so. I personally run a weekly NULL job to do this.

5.3 Job Input Management

The job scheduler is for running batch jobs, your batch jobs should never prompt for interactive input if written correctly.

However one of my jobs on a restart from an alert state tried to overwrite a tar file and prompted for permission to continue and hung as the scheduler didn't respond. That was an error in my batch job, but the job hung. Fixed... batch jobs that prompt for input now get a response.

All the batch jobs that are run from the scheduler will have an STDIN of /dev/null, so if your batch jobs prompt they will not hang. It may not be the answer they want and they should go splat (and if your script is coded correctly go into failed state).

This redirection of input from /dev/null is intended to stop jobs hanging when a batch job prompts for input and screwing up the scheduler, not to make your script work. Batch jobs should never prompt for interactive input, it is an error if they do.

5.4. Job Failure Handling, what is a failed job

A program or shell script defined as the execprog in a job definition entry is expected to return 0 if it completed without error.

If a job terminates with a non-zero error or is terminated by an external signal (ie: kill -9) the job is considered to have failed and is placed into an alert state.

By default the alert message text contains the reason code the job ended with, for example

MSG TEXT: JOB TEST-RUN4-60MINS : EXIT CODE=1

MSG TEXT: JOB TEST-RUN4-60MINS : KILLED BY SIGNAL 9

A failed job can be restarted or forced-ok using the alert management functions; once the problem that caused it to fail has been fixed of course.

6. Starting and Stopping the Scheduler

6.1 Must run as root, why ?

The job scheduler is designed to be a central batch point, while it is not intended to replace cron obviously any batch job you want monitored really can't run under cron. So for important batch jobs rather than running them out of individual 'batch user' crontabs they should be run under control of the job scheduler.

Which simply means the job scheduler must be able to submit jobs under multiple unix userids, the root user can of course submit jobs under other userids, a non-privalidged user cannot. Thats it basically.

6.2 Starting and Stopping

6.2.1 Recommended, use installed scripts for automatic start

If you installed the Job Scheduler using the supplied ***jobsched_install_binary.sh*** installation scripts it will be automatically started at system boot time and automatically stopped at system shutdown time. The installation script also gives you the option to start it running immediately as part of the install.

Not only that the install script will have customised all configuration files

If you used the install scripts simply /etc/init.d/xxx start and /etc/init.d/xxx stop (or in Linux service xxx stop and service xxx stop) where xxx is the name you gave the init.d script whne you were replying to the prompts during the installation.

6.2.2 Manually

If you must write your own scripts. The rules to follow are
cd /installeddir
./jobsched_daemon [port-number [interface-ip-address]]
port number defaults to 9002
ip-interface-address defaults to listen on all available interfaces

To shutdown, review the supplied server.rc3_d script and merge the shutdown commands into your script as needed. The job scheduler itself is now robust enough to recover from a complete unscheduled halt but you should at least try to shut it down cleanly.

6.2.3 Solaris Notes

Under Linux the jobsched_daemon can switch to daemon mode automatically within the program code. That is not possible under Solaris.

For solaris update any script you use to use 'daemon startupscript &', remember the & at the end.

6.3 Startup Logic

It's always possible the job scheduler was not cleanly stopped. Database consistency checks are done at application initialisation time

When the scheduler is restarted is scans its active job queue to see if any jobs were last recorded as 'executing'. Any jobs that were executing are set to a 'failed' state with an alert raised of 'job executing when scheduler stopped, check joblog'

Jobs in this state need to have the job logs checked so it can be decided whether an 'alert restart', 'alert forceok' or 'sched delete' should be used to reset them on the active job queue.

6.4 Shutdown Logic

6.4.1 Controlled shutdown – sched shutdown

When the scheduler gets a shutdown request it disables the scheduling of jobs. Running jobs will be allowed to complete to ensure a complete audit trail, and all commands still work; but any jobs that become available to run (or new jobs submitted to run) will not be permitted to start. They will be available, and be able to run, when the scheduler is restarted.

6.4.2 Immediate shutdown – sched forcedown

If the scheduler is shutdown with a 'forcedown' . If the scheduler is shutdown with a forcedown option then all tasks the scheduler has started directly will be killed with a -9 signal and the jobs placed onto the alert queue. This option should only be used as provided in the .rc scripts during a system shutdown.

It will NOT stop any tasks that have been started on behalf of the child tasks we started. When the scheduler is restarted jobs on the alert queue will need to be manually checked to see if they completed. If as recomended the forcedown is only used during a system shutdown then the child tasks should have been killed by the system and not completed, requiring intervention.

The forcedown is intended for system shutdown, when we need the server to clean up and close files normally, but cannot wait for running jobs to complete normally. It is not intended for regular use.

7. Database Rules – Admin Faststart

You users will probably have silly questions if they have not bothered to read the user manuals. All key rules are in the jobsched_cmd user guide they probably didn't bother to read.

This section is not required for an administrator to start/stop the job scheduler.

But while all these rules are documented in the jobsched_cmd user manual, the chances that any user will actually use a manual is about zero percent.

When a user calls you up and says a command will not work, one of these is probably why.

7.1 Job database

The Job definition file contains the job definition entries. Used at schedule on time and at execution time.

A job cannot be deleted if it is on the active queue. It must be deleted from the scheduler queue first. Job update has not been implemented. To update a job delete then re-add.

Filename: job_details.dbs

7.2 Active job queue

The active job queue contains jobs scheduled to run 'today', 'today' being 24hrs from the newday jobs time. Jobs may be in time-wait, dependency-wait, failed or completed/deleted state.

If a job is deleted from the scheduler queue, all jobs waiting with a dependency on that job will have the dependencies cleared for the job being deleted. All alerts for the job will also be deleted.

If a job fails (exit code <> 0 or killed by a signal) you should fix the problem before restarting the job. The job can only be restarted from the ALERT functions (restart command).

Filename: scheduler_queue.dbs

7.3 Alert Queue

The alert queue contains entries for jobs that completed with a non-zero exit code or that were killed by a signal. This queue must be cleared before a newday job can run.

Failed jobs can only be restarted from the alert queue. You cannot delete a job from the alert queue. You must delete it from the scheduler queue which will also delete associated alerts.

Filename: alerts.dbs

7.4 Dependency Queue

Contains entries for events a job is waiting on; other job names or disk filenames.

You may manually delete a dependency item if you decide a failed job is not going to be run that day. You may manually clear all dependencies against a specific job if you decide it can run immediately (although in that case you would be better off not defining dependencies in the first place). You can list entries on the dependency queue by what dependencies is a job waiting on or by what jobs are waiting on this dependency

Filename: dependency_queue.dbs

7.5 Calendar File

Contains job and holiday calendars. You cannot delete a holiday calendar if a job or another calendar refers to it. You cannot delete any calendar if a job refers to it. You may at any time adjust the calendar values with date merge and unmerge commands.

Filename: calendar.dbs

7.6 User file

The job scheduler server maintains its own list of authorised users. This is required as the final GUI solution will allow PC users who may not have unix logons to administer the server. Each user record defines the server authority level the user is entitled to. Users can currently be broken down into Admin, Operator, Security, Jobauth or browse-only access groups.

User records may have the optional autologin-allowed flag set to allow them (**only**) using the jobsched_cmd program to logon without a password if the unix userid running the jobsched_cmd program matches a scheduler user record with the autologin flag enabled. This is required by root for system shutdown to avoid codeing an admin password in the init.d script. I also use it for the user apache to give cgi scripts run by my web server autologin to operator access to manage the active job schedule.

Filename: user_records.dbs

7.7 Alert Customisation file

This is a normal text file that allows user customisation of alert messages.

How to use this file is covered in section 4.3.

Filename: alerts_custom.txt.

8. Reserved job names

*8.1 SCHEDULER**

Jobs beginning with SCHEDULER are jobs used internally by the application. They are coded internally and do not appear in the job definition file as they cannot be customised by users.

If a user does manage to (by external editing I assume) add a job beginning SCHEDULER then the job scheduler will just delete it on its next database consistency check.

The only current internal job is SCHEDULER-NEWDAY.

8.1.1 SCHEDULER-NEWDAY

The time this job runs is controlled by the scheduler configuration file 'new day' time, and whether the catchup function is on or not. If the catchup function is on, when the scheduler is shutdown for a few days the newday will run for each day missed.

The SCHEDULER-NEWDAY job will not run if any jobs are remaining to be run. It will either go into an alert state requiring a manual restart or add a dependency to itself on one of the jobs waiting to run. The action taken depends on how the newdayfail action has been set, the newdayfailaction is set using the job_cmd utility. The default is to go into alert state.

*8.2 NULL**

Any jobs submitted with a jobname beginning NULL will have their output written to /dev/null instead of to a log file. This class was created to allow scheduled jobs to perform maintenance on the logs directory, which could not be done if they were writing to it. Because of the requirement for an inactive logs directory these jobs run exclusive of any other jobs, they will not run while other jobs are running, and when they do start no other jobs will be permitted to run while they are executing.

9. Backing up, and Recovering your scheduler databases, jobsched_take_snapshot

At some point you will break your job scheduler environment, whether due to user error or a hardware failure. You should be taking regular database backups in case you need to restore from them.

The jobsched_take_snapshot utility will dump the contents of all your database files into a text file that can be piped into the jobsched_cmd program to rebuild your databases from scratch.

9.1 Backing up your databases, jobsched_take_snapshot

The jobsched_take_snapshot utility is provided to take a snapshot of the environment and jobs of the running job scheduler system.

This snapshot can be used to rebuild the job scheduler environment as at the time the snapshot was taken.

It will create a file with all the job definitions and rundates as at the time of the snapshot. It will also generate as commented lines the submit commands required for the jobs to be immediately rescheduled on.

If using the snapshot file for recovery it will have to be manually checked/adjusted for your environment, as the databases are unlikely to become corrupted at the second the snapshot was taken. You will have to manually adjust for jobs that may have run between the failure time and the time the snapshot was taken.

It has been updated to dump out the user database information also, although as it doesn't know the passwords the user passwords are all set to 'default' so if you recover users from this file you will need to update all these also.

The output produced is a standard text file containing commands that can be fed into the jobsched_cmd program to redefine all the database entries that existed at the time the snapshot was run.

```
The syntax of the jobsched_take_snapshot program is...
     jobsched_take_snapshot path/to/install/directory > config file


Example if you installed the scheduler to /opt/scheduler
     cd /opt/scheduler
     ./jobsched_take_snapshot ./ > backup_file_yymmdd

     Or
       /opt/scheduler/jobsched_take_snapshot  &
              /opt/scheduler > /opt/scheduler/backup_file_yymmdd
```

9.2 Restoring from a snapshot backup

9.2.1 Customise the snapshot file you wil restore from

FIRST, customise the snapshot file. The snapshot file as taken will have the jobs set to run as of the time of the snapshot, but job submit commands will have been commented out.

You must review the file to adjust dates where appropriate for those jobs that will have already run at the time your system became corrupt and uncomment the submit lines for the jobs you want to schedule back on immediately.

You will also need to change the user initial passords if you are recovering the user file also.

After you have corrected the file for differences that have occurred between the time the snapshot was taken, and the actual time your system failed...

9.2.2 Shutdown scheduler and delete database files

Shutdown the scheduler !!!, it must be down. Either logon as an administrator and shutdown the scheduler or run the /etc/init.d script you installed to stop the scheduler.

Delete the database files you have determined are corrupt. To delete everything and start from scratch use...

cd to the directory you installed the application to

rm *.dbs

rm config.dat

9.2.3 Restart scheduler

Restart the scheduler using your sites standard scheduler startup method, whether from a custom script or the /etc/init.d script you installed. When the scheduler is restarted it will create the database files again, and initialise them with default values.

9.2.4 Rebuild scheduler databases from snapshot

To rebuild the scheduler as it was at the time the snapshot was taken you need to pipe your edited snapshot file into the jobsched_cmd program. ***The scheduler must be running***.

Example (reloading from the snapshot file created in the backup example):

```
cd /opt/scheduler
cat  backup_file_yymmdd | ./jobsched_cmd
```

Done, jobs, calendars and users have been re-defined to the job scheduler, and global configuration values returned to what they were at that time also.