jobsched_cmd user guide

A component of Mark Dickinsons Unix Job Scheduler

This manual covers the use of the command line interface to Mark Dickinsons unix Job Scheduler.

This manual is for version V1.13 of Marks Job Scheduler

Other reference material available

Job Scheduler Daemon Configuration Manual

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. What this jobsched_cmd program is

The jobsched_cmd program is a command line interface to the job scheduler server task. It is totally full-function as far as supported server operations are concerned, as this program was used to test the server functions and api calls so encompassed all server functions that have been coded so far.

The program design is based toward its use in script files, ***as such it works on a fast fail philosophy*** (it dies/exits onthe first error)as I don't want piped input commands to keep streaming into this program if one has failed as part of a batch job I intend to keep the fast fail policy for this program, although it does get a bit annoying during interactive use.

As the job scheduler server task accepts its commands via a tcpip socket, this jobsched_cmd program does not have to reside on the same system as the server as the ip-address and port number can be provided when the program is run. This could be used as a security option if needed (see security issues below).

This program is currently THE ONLY WAY to communicate with the job scheduler server task. GUI development has not yet been started (apart from some simple auto-refreshing web pages I used to save typing during testing).

2. Security issues through use of this program

2.1 Issues that exist currently

The job scheduler must run as the root user, as it needs to be able to log down to other userids when it submits jobs.

NEVER setuid the jobsched_cmd program to root ANYWHERE as the root use FROM ANY SYSTEM is always granted full admin access rights to any job scheduler server, even if the server is running remotely from where the root user was logged on.

As such there is a very high risk that if the jobsched_cmd interface program it not tightly secured unauthorised user can run commands as root. Do DO NOT setuid the jobsched_cmd program to root.

Other users require individual authorisation records to be setup within the job scheduler on each system they are to be granted access to.

2.2 Security that has been implemented IN THE SERVER so far

The default access level for any new connection is 'guest'. This allows browse only access to the job scheduler server.

From this level connected sessions can logup to a higher authority levels if their individual jobscheduler userid entry allows it. Note that the root user is a special case that can always logup to administrator level.

Each individual user logon record specifies the access level they are to be granted after a successfull logon. The access levels are

guest/browse	- read only
security	- can only add/delete users, cannot see job activity
jobauth	- can only define or delete job entries in job file. No access to running jobs
operator	- can manage jobs on the active scheduler, and submit jobs from the job database, cannot define new jobs
admin	- can do anything

An individual user can also be granted user management rights even if not a user with an access level of security.

The root user running the jobsched_cmd program can always logon to the administrator level without a password. THIS WILL ALWAYS BE ALLOWED as the root user needs to be able to issue commands from the init.d scripts at shutdown and startup time so applying a password to root root would be detrimental in that the root password would have to be hard coded in the scripts.

2.3 Recommended security settings

NEVER setuid the jobsched_cmd program, not even on a development system or this would give the development system full admin rights to all your production systems also.

Only add user records within the job scheduler environment on those systems the user is expected to perform scheduler tasks on. DO NOT set up adduser scripts that add users enterprise-wide.

If you use the supplied html pages to monitor the scheduler a userid must be created allowing autologin to the operator level for the unix userid running the web server (ie: apache).

ALWAYS ENSURE THE SHELL SCRIPT FILES TO BE EXECUTED ARE TIGHTLY SECURED. It serves no purpose to lock down who can add jobs to the server, if anybody can edit the script file that is to be executed with their own desired commands.

It is strongly recomended that your system administrators be assigned individual user records with administrator access. If this is not done then all their work will have to be done under the root userid making the audit trail useless.

2.4 Default users

There are always two default users in the database, those are the 'guest' and 'auto-login' users. You may never delete these, if these users are not present the server will attempt to recreate them, and if not possible will refuse to run.

Guest provides a documented profile for guest users, while not essential I have decided it must remain.

auto-login is used by the root user, and must always be present. The root user needs this for such things as the init.d shutdown scripts.

YOU MAY NOT REMOVE the default users. The server will refuse to delete the auto-login id. While the guest may be deleted there is a sanity check done each newday processing that will add it back again if it is missing. If both newday users do not exist, or cannot be recreated at the newday processing time the job scheduler server will become inoperable.

As part of the install you should have also added an entry for the unix userid that runs your web server. If this user does not exist then the browser pages will not function.

3. Syntax for running the jobsched_cmd program

```
jobsched_cmd [ portnumber  [ ipaddress ] ]
```

note that if you wish to override the default ip-address you must also provide a port number to the program.

The default port number is 9002. If you have changed the default port number used by the server you will need to provide the new port number here.

The default ip-address is 127.0.0.1 (localhost). You can overwrite this when you need to talk to a job scheduler server on a remote system.

Example: ./jobsched_cmd 9002 127.0.0.1

NOTE: Use ip address for this release, not a hostname.

4. User definition and management

4.4 User Management

4.4.1 Adding new users

```
Some users may not have unix passwords (if using the PC GUI), in
that case you must pick a meaningfull name. If they do have a
native unix userid then you should use that as their username
as that can be used for 'aoutologin' if desired.

If the user has a unix password and the 'aotoauth' is set to yes
then that user may use the 'autologin' command to logon without a
password. NOTE that autoauth is done on the server the user is
running the jobsched_cmd program on, NEVER give a development
user autoauth if their userids are also setup on production
servers also, as the autoauth function is global (ie: a dev user
with autoauth can use the dev machine to alter production
schedules on production machines).
The AUTOAUTH function should be set to yes only for the
systems administrators, never for development OR SECURITY
users.

USER ADD username, PASSWORD somepassword,
     AUTH ADMIN|OPERATOR|JOB|SECURITY,
     AUTOAUTH YES|NO, SUBSETADDAUTH YES|NO|ADMIN,
     USERLIST "<1>,<2>...<10>",DESC "user description"

 username can be any arbitrary name you like. HOWEVER if the user
 is to be permitted 'autoauth' access then it MUST match their
 unix userid (as in the /etc/passwd file) exactly, as for the
 autoauth function the contents of the passwd file are used by
 the jobsched_cmd program to fill in the username FROM THE
 LOCAL SERVER the jobsched_cmd is run on.

 password must be at least six characters

AUTH levels are defined as...
    ADMIN     - all access
    SECURITY  - user management only
    JOB       - job definition management only
    OPERATOR  - scheduler queue management only
    BROWSE    - display only access (default)

AUTOAUTH is
  YES=no passord required (user must be logged on at
     the unix level as this id)
  NO=must provide a username password.

SUBSETADAUTH is
  NO, can't add any users
  YES, can add users with the same access level this user has
  ADMIN, adds full security access to the user (user has root auth)
```

```
USERLIST “xxx,xxx,xxx,xxx”, up to 10 users
  a comma seperated list of unix userids that this user is
  permitted To create jobs for. NOTE: only used if a jobauth
  level. Ignored at at admin level as admin can do whatever they
  wish anyway, and other levels don't have access to define jobs.

DESC “user description”
  a freeform 50 byte description of who the user is or what the
  userid is being added for if a generic user.
```

4.4.2 Deleting a user

Deleting a use from the job scheduler server databases will remove the entry from the job scheduler on the system you are currently connected to. IT WILL NOT AFFECT the status of any user that is currently logged on with this userid to that system until they logout.

```
USER DELETE username
```

username is the name of the user known by the selected job scheduler server that is to be deleted from that job schedulers databases.

4.4.3 Changing a user password

Sometimes users forget their passwords. While I'm personally in favour of deleting such users, you may change the password field of a user as below.

```
USER PASSWORD username,newpassword
```

usernameis the name of the user known by the selected job scheduler, newpassword is the nw password for the user.

4.4.4 Displaying a user record

This command will display the settings associated with a job scheduler user record.

```
USER INFO username
```

username is the name of the user known by the selected job scheduler that is being queried.

5. Job definition and management

5.0 Who can define and delete jobs

Only users with JOB authority set on their scheduler user record are able to manage jobs and calendars.

The unix root root may also run jobsched_cmd and use the AUTOLOGIN command to switch to administration access level to gain access to manage jobs and calendars.

5.1 Defining a job, syntax and considerations

5.1.1 Job definition syntax

```
Syntax...
JOB ADD jobname,OWNER unixuser,TIME YYYYMMDD HH:MM,
       CMD unixcommand[,PARM "parameters to the unix command"]
       [,DEP "JOB jobn|FILE filen" total of five dependencies]
       [,REPEATEVERY nn]
       [,DAYS "MON,TUE...,SUN"][,DESC description with no commas]
       [,CATCHUP YES|NO]
       [,CALENDAR calname]
```

```
The command parameters
   OWNER      ** Required **
              Identifies the unix userid under which the job will run
   TIME       ** Required **
              The date and time of the first execution of the job, this
              may be before the current date and time as I have found that
              usefull for stress testing.
   CMD        ** Required **
              The program or shell script to be run. This must the full
              path name of the file to be executed. THIS SHOULD NOT
              INCLUDE ANY PARAMETERS. May only be up to 80 characters.
   PARM       Optional..... use only if the program or script needs
                            parameters provided to it.
              Any parameters that need to be passed to the program or
              shell script. These need to be defined here. They must be
              provided within a " quote pair. There are 'special'
              internal parameters that may be passed, see the section
              on special parameters (5.1.2).
   DEP        Optional..... use only if the job being added is to be
                            dependant upon other jobs completing or a file
                            ariiving on the system.
              A list of dependencies that must be preovided within a "
              quote block. Each dependency must identify whether it is a
              physical disk file or scheduler scheduler jobname prior to
              the dependency value.
              The dependency may be another job scheduler job name (job
              will wait until that prior job completed) or a unix filename
              (job will wait until file exists and hasn't been modified
              for 5mins, to stop job trigerring while file being waited
```

```
                on is still being written to by whatever created it).
                A jobname may be 30 characters, a filename 80 characters.
   REPEATEVERY  Optional..... use only if job is to repeat every nn minutes
                If this parameter is define the job will be created as a
                automatically requeueing job that will run every nn minutes.
                (*See the Considerations below, it may not work as you
                expect)
      DAYS      Optional..... use only if a job is to run on specific
                weekdays. A list of daynames that this job is permitted
                to run on, if provided the dayname list must be within a
                " quote block, and be three character names seperated by
                a ,. The datnames are MON TUE WED THU FRI SAT and SUN.
      DESC      Optional..... but should be used
                A description for the job WITH NO COMMAS (,). If a
                description is not provided for a job the entry
                'No decsription entered' will be used.
      CATCHUP   The default is yes, which is that when the scheduler is
                configured with catchup on then the job will be executed
                for each scheduler newday catchup done as long as it's
                next scheduled rundate remains within the execution date
                range. Using CATCHUP NO for jobs such as disk cleanups
                that delete files over n days old prevents those
                jobs from repeatedly running during the catchup
                cycles, as obviously they only need to run the once.
      CALENDAR  The name of a calendar to be used to control the job
                scheduling times. This calendar must exist.
```

```
   Considerations
   * If the DAYS option is used and the time provided does not fall on one
     of those days the server will adjust the date to the next correct day
   * If REPEATEVERY and DAYS are both provided the job will only repeat on
     the days specified.
   * A maximum of five (5) dependencies are permitted. This may be job names
     as known to the scheduler, or files you expect to appear on the system.
     Five in total, not five of each.
   * REPEATEVERY considerations
     - jobs requeue to the next slot. If a job is set to run every 15
       minutes but takes 20 minutes to run the effect is it runs every 30
       minutes (a execution slot is missed as it ran past it).
     - repeat jobs will not be requeued if the next scheduled runtime for
       the job is after the scheduled SCHEDULER-NEWDAY runtime, it will
       start running again after the newday job has completed.
     - So don't use for jobs that MUST run every nn minutes.
   * File dependencies are checked every 5 minutes. A dependency will not be
     released just because a file exists, but will wait until the file
     being checked has not been modified in over five minutes. This ensures
     that things like 60 minute ftp's actually complete before the job runs
     rather than triggering the second the file is created. The effect is
     that a file can be on a system for up to 10 minutes before the trigger
     is released. This is normal.
```

5.1.2 Special parameters a job may use

The following special parameters may be passed as keywords in the job being defined. They will be substituted with values at the time the job is executed.

~~DATE~~

```
If this is provided it will be replaced by the date (YYYYMMDD)
that the job was supposed to run on, not the current date.
This can be used by any date sensitive jobs when the scheduler is
catching up on any missed days processing, or for jobs supposed to run
after 23:00 that may actually be running on a new date after
midnight if the scheduler is busy.
Any shell script using this will need to check for it as per
    any normal parameter, ie: parm="~~DATE~~", .sh script uses ${1}
    to test against and will receive a YYYYMMDD numeric value.
```

5.1.3 Examples

Examples
(note: commands need to be entered on one line, just broken for formatting here)

```
1) A job to run the 'df -k' command every monday and friday at 17:00 as
   userid fred.
   JOB ADD TEST-DF,OWNER fred,TIME 20020408,CMD /bin/df,PARM "-k",
           DAYS "MON,FRI",DESC test day selection

2) A job to run a shell script /opt/fred with no parameters every 15
   minutes
   JOB ADD TEST-REPEAT,OWNER fred,CMD /opt/fred,
          TIME 20020408 09:00,DESC test repeating job

3) A job to run after TEST-JOB1 completes and file /opt/testfile has been
   received onto the system. Will run command /opt/fredagain
   JOB ADD TEST-WAIT,OWNER fred,CMD /opt/fredagain,TIME 20020408 09:00,
       DESC test waits,DEP "JOB TEST-JOB1,FILE /opt/testfile"
   (note: command needs to be entered on one line)
```

5.1.4 Reserved and Special Job Names

SCHEDULER-NEWDAY

```
    This is a reserved jobname. You may not use it. It is created and
    managed by th scheduler itself and any commands issued against it
    will be rejected.
```

NULL*

```
    Jobs beginning with NULL are intended for scheduler maintenance jobs,
    I created this special prefix so I could use the scheduler itself to
    run a job to archive scheduler job output logs.
    Any job beginning with NULL runs with the following special
    characteristics
      - is the only job running (it will not run if any other job is
        currently running, once it starts running no other job is
        permitted to start until it completes running)
      - it does not write any output to a job log file
    This ensures the joblogs can be safely archived as it guarantees
    no joblog activity while it is running. There may be other uses
    you can find for jobs like this.
```

5.2 Deleting a job

Deleting a job removes it from the job definition file. Once this is done the job is gone forever, never to be scheduled again.

THIS SHOULD NOT BE CONFUSED wil the command SCHED DELETE which is used to delete a job off the active job queue. The JOB DELETE command is forever.

Syntax...
```
JOB DELETE jobname
```

Considerations
* You cannot delete a job from the database if it is on the scheduler active queue. You need to delete it from the scheduler active task queues before deleting it from the job database. This means it must be deleted from the alert queues and the scheduler queues (if it is on the scheduler queue) with the SCHED DELETE jobname command first.

5.3 Displaying a list of defined jobs

To display a list of all jobs in the job defininition database use the jobsched_cmd command

```
JOB LISTALL [jobmask]
```

The optional jobmask allows you to display only a subset of jobs, for example to display only jobs of the form TEST-JOB-xxx
you can enter the jobsched_cmd...
```
JOB LISTALL TEST-JOB-*
```

5.4 Displaying a defined jobs details

The full definition details of any job in the job details database can be displayed with the JOB INFO command.

```
JOB INFO jobname
```

Where jobname is the jobname you wish to display the details for.

5.5 Submitting a job, manually (and why it may be bad)

It is possible to manually schedule a job onto the active queue. This is inadvisable unless it is a new job you have just added that you wish to run under the current days schedule.

If you manually schedule a job on it will be moved to the active queue NOT FOR THE CURRENT DAY but scheduled to run at the next time it would have normally been automatically scheduled to run for. THIS IS BAD.

WHY IS THIS BAD ?
The scheduler newday task will not run until all jobs scheduled for the 'current business day' have completed running. It on monday you manually schedule on a task that is due to run on a sunday that job will go into time-wait until sunday, preventing the newday task from running. The effect of this is that jobs scheduled for TUE-SUN will not be scheduled on each day. SO DON'T DO IT. Scheduling is done automatically, see below.

If you have just added a new job, that has a date/timestamp for the current days run, you can add it to the current days active queue with the command...

```
JOB SUBMIT jobname
```

where jobname is the name of the job you have just added.

5.6 Automatic scheduling on of jobs, how it works

An internal job SCHEDULER-NEWDAY runs each day at or after the time configured to the server as the newday time. It may run late if there are still jobs on the active queue waiting to run; it will wait until all jobs have completed other that itself before running.

This job scheduler system is based around production days, where an entire days work must be completed before the next days work is scheduled on.
If at the time the newday job is scheduled to run the newday job will either go into alert state, or just add to itself a dependency of one of the jobs still on the queue. Which action is taken depends upon how you have the server configured; the newday failaction is user configurable.

When all previous days jobs have completed the newday task will scan the jobdatabase file and automatically schedule on all jobs that were due to run within the next 24hour period of its scheduled (as opposed to actual) start time.
If the newday job was configured to go into an alert state then operator intervention would be required to restart it after the last queued job had completed.

This method ensures that each days jobs are discretely seperated so operators can easily track them.
The newday job time is customisable, so can acurately relfect when you expect your sites daily batch to complete.

6. Calendar Functionality

6.1 Calendar functionality overview

```
Calenders may be used in the following ways...
* A holiday calendar is a calendar that contains dates that
  any calendar that references it is not to run. A holiday
  calendar may only be attached to a job calendar to override
  dates in that pre-existing calenday, not to a job directly.
* A job calendar can be defined with a list of dates that
  a job is to run. It can be created to also refer to a
  holiday calendar that will override some of those dates. A
  job calendar can only be attached to a job.
* Any Job be defined to use a job calendar will inherit
  the holiday calendar associated with the job calendar
  also.

PERFORMANCE HIT WARNING :::
When a calendar is updated every job record must be checked
to see if it needs it's next runtime updated as a result of
the calendar change. For a job calendar change only the
jobs that use the calendar must be updated; for a holiday
calendar change any job that uses ANY calendar will be
updated just in case the jobs calendar uses the holiday
calendar further down the chain. If you have a lot of jobs
defined expect a big performance hit with calendar
commands.
```

6.2 The FORMAT function used to define dates

```
On the calendar commands used to change data in the calendar
database a FORMAT parameter must be provided. This parameter
is used to define the way you wish to define dates to the
calendar, and the actual dates themselves in the format you
have selected.

The possible format parameters are...
     FORMAT MONTHSDAYS "MM/DD,MM/DD,MM/DD,MM/DD,MM/DD..."
     FORMAT DAYS ALL|JAN|FEB...DEC "DD,DD,DD,DD..."
     FORMAT DAYNAMES "SUN,MON,TUE,WED,THU,FRI,SAT"

These are pretty self explainatory, the first format
allows you to enter dates to the calendar using a
month day pair, the second allows you to specify
individual dates in a month (or all months), and the
third will identify dates for the entire year by
day name.
```

6.3 Adding Calendar Entries

The syntax for adding a new calendar entry is as follows (to be enetered on one line)...

```
   CALENDAR ADD calname,DESC "XXX",TYPE JOB|HOLIDAY,YEAR YYYY[,HOLCAL
holcalname],FORMAT format-string
```

The values of format-string were covered in section 6.2.

calname is the name of the calendar. You may use the same name for a job calendar and holiday calendar if desired for clarity as they will still be unique by calendar type.

The description "XXX" may be any description up to 40 bytes describing the calendar, it must be between " characters.

The TYPE may be JOB or HOLIDAY. A JOB calendar contains dates a job will execute on, a HOLIDAY calendar is used to define dates a job will NOT execute on.

The year is also part of the calendar key. It MUST be provided. It ensures that calendars do not accidentally wrap across years from the job execution perspective. (ie: if a job uses calendar JOB1-CAL in 2002 when the calendar 2002 expires it will look for a JOB-CAL1 for 2003, if found good, it has a new set of dates; if not found the job will be placed into a 'calendar error' state). This is also used to maintain the database, as in 2003 all calendars for 2002 will be purged.

IMPORTANT: A job will use a calendar for it's current year, the calendar will use the holiday calendar for the current year etc. Ensure you have your next years calendars in place before the end of the year if you are using calendars.

6.4 Changing Calendar Values

6.4.1 Adding extra days into a calendar

Once a calendar has been created you may add additional execution dates to it with the CALENDAR MERGE command. This command adds extra dates to an existing calendar.

The command needs to be entered on one line.

```
CALENDAR MERGE calname,TYPE JOB|HOLIDAY,YEAR YYYY,FORMAT format-string
```

The format-string parameter values were covered in section 6.2

6.4.2 Removing days from a calendar

Once a calendar has been created you may selectively remove execution dates from it with the CALENDAR UMMERGE command. This command subtracts dates from an existing calendar.

The command needs to be entered on one line.

```
   CALENDAR UNMERGE|SUBDELETE calname,TYPE JOB|HOLIDAY,YEAR YYYY,FORMAT
format-string
```

The format-string parameter values were covered in section 6.2

6.5 Deleting a calendar entry

Deleting a calendar is not as simple as it seems. The command syntax is below, and the parameteres have been explained earlier; the restrictions on the command are...
* You may not delete a holiday calendar if any job calendar references it.
* You may not delete a job calendar if any job references it.

```
CALENDAR DELETE calname,TYPE JOB|HOLIDAY,YEAR YYYY
```

6.6 Listing all calendar entries

To list all active calendar entries from the calendar database use the command below. It will list the key fields of year, type, name and description for each calendar in the server database.

```
CALENDAR LISTALL [calmask]
```

The optional calmask can be used to select entries by wildcard to limit the amount of responses returned (ie *IRST* would find FIRST-OF-MONTH)

6.7 Displaying details of a calendar

To display detailed on a specific calendar entry use the command below. It will display a the yearly table of execution dates for the calendar.

```
CALENDAR INFO calname,TYPE JOB|HOLIDAY,YEAR YYYY
```

7. Managing jobs in alert status

7.1 Displaying the alert queue

7.1.1 Display all jobs on the alert queue

A summary of all jobs currently on the alert queue is generated with the follwoing command.

```
ALERT LISTALL
```

If no alerts are on the queue an empty display will be returned.
If there are alerts they are returned in the following format...

7.1.2 Display details for a specific alert

To find out more details on an alert the command

```
ALERT INFO jobname
```

Where is the name of the failed job will display the detailed information for the alert.
Common errors would be exit code 127 (program to be run was not found) or a signal error indicating the program was killed with the signal number in the detailed display.

As a general rule any other exit code will be the exit code returned by the program or script being executed so you will need to refer to the documentation for the program or script to determine why the exit code was generated.

Your job scheduler adminsitrator may have used the alert customisation table when installing the scheduler, in which case you may actually see meaningfull text for the error description; if you get in the alert description field no text for error then they have not yet done that. But setting that up is in the job scheduler daemon manual and only for the administrator to play with.

7.2 Acknowledging an alert

When an alert is written to the queue it is in an unacknowledged state. Acknowledging an alert really has no effect on the scheduler but is usefull for monitoring interfaces (ie: the supplied sample monitoring interfaces can display the alert in yellow instead of red once it has been acknowledged).

To acknowledge an alert use the command

```
ALERT ACK jobname
```

where jobname is the name of the job on the alert queue you intend to acknowledge.

7.3 Restarting a failed job

Restarting a job assumes that the job can be rerun from the start. All jobs run by the scheduler should be able to be restarted from the beginning without causing problems. See the job scheduler job writing guide manual for that.

also obvously assumes you have fixed the problem that caused the job to fail in the first place.

Restarting a job will delete the alert, and reschedule the job to be run as soon as the scheduler is able to do so.

The command to do this is

```
ALERT RESTART jobname
```

where is the job you wish to requeue back onto the execution queue to run again.

7.4 Forcing a job from alert status to OK status

There will be times where a job is unable to be fixed but you do not wish the batch stream to be held up; The forceOK may not be appropriate for this (see considerations), one of the dependency delete functions may be better in that instance.

If however you have examined the job output of a failed job you may decide that the job did complete OK. In this case the forceOK should be used to mark the job completed as it ensures ALL the database flags are updated correctly to indicate completion of the job.

The forceOK will
* delete the alert
* mark the job completed in the scheduler
* update the job database last runtime to the forceok time
* mark the jobname dependencies on all jobs waiting for this job as satisfied.
* If it is a repeating job, will requeue it to the next time

The syntax for this command is

```
ALERT FORCEOK jobname
```

where is the name of the job on the alert queue.

Considerations

As this command marks the job OK AND UPDATES THE LAST RUN AND NEXT RUN

TIMESTAMPS it is not possible to resubmit the job for the current day.

If the actual intent is to allow other jobs to begin running while this one is fixed then you should use the dependency queue commands to release dependencies from waiting jobs, allowing them to run; rather than forceOKing the alert state job off the system. Thie dependency releases would allow the other jobs to run and the alert job to still be restarted when the problem is fixed.

8. Dependency queue management

8.1 Listing all dependencies on the dependency queue

To list all dependencies still outstanding for the current days processing use the command

```
DEP LISTALL
```

which will list the jobnames in the dependency queue, and what dependencies each job is waiting for.

Example:

```
command:DEP LISTALL

Job TEST-FILE-DEP                    is waiting on...
    FILE /home/mark/scheduler/testing/trigger
Job TEST-JOB-DEP                     is waiting on...
    JOB  TEST-FILE-DEP
    JOB  TEST-FAIL-AND-RESTART
Job SCHEDULER-NEWDAY                 is waiting on...
    JOB  TEST-FILE-DEP
```

8.2 List what jobs are waiting on a dependency name

To see what impact globally clearing a dependency will have on the current days scheduling, the command

```
DEP LISTWAIT depname
```

where is a full dependencyname value, either a jobname or flename. This command will show all jobs that are waiting for the dependency to be satisfied.

Example:

```
command:DEP LISTWAIT /home/mark/scheduler/testing/trigger

TEST-FILE-DEP
```

8.3 Clearing all dependencies for a specific job

Occasionally, for whatever reason, you may not wish a job to wait until all its dependencies have been satisfied before starting running.

This may be a DANGEROUS decision, as I'm sure the dependencies were setup for a reason. If however you are absolutely sure this will have no nasty impacts on your days batch processing...

The command

```
DEP CLEARALL JOB jobname
```

where jobname is the jobname of a job waiting on dependencies, will mark as satisfied FOR THAT JOB ONLY all the dependencies it is waiting on.
The job will then be requeued on time-wait rather than dependency-wait waiting for its scheduled time to run.

8.4 Forcing a dependency name to be globally satisfied

There will always, however infrequently, be the occasion where a job fails that just cannot be fixed in time, or pherhaps a file transmission that is just not going to arrive that day. In the later instance of course you would delete jobs relying on the file rather than just removing the dependecy I hope.

A command to delete a dependency across all jobs waiting for it has been provided for this eventuality. This is the command

```
DEP CLEARALL DEP depname
```

where is the full dependencyname (whether jobname or filename) that you have deterimined is just not going to be satisfied in the current days batch run.

This command will, for each job waiting on the specific dependency named, have that specific dependency marked as satisfied. No other dependencies jobs may be waiting on will be affected for jobs being modified.

9. Running job management commands

9.1 Listing all jobs on the active queue

```
To list all jobs on the current scheduler active queue use the command

SCHED LISTALL [jobmask]

Using a wildcard mask can be used to limit the response returned, for
example to display test jobs all following the naming standard TEST-JOB-xxx
you could enter...
SCHED LISTALL TEST-JOB-*

This command only displays jobs still on the active scheduler queue;
those running, waiting to run or in failed state. It does not provide
information on jobs that have completed as they are of no interest to
the active schedule queue anymore.
Note: your administrator can turn on the display of completed/deleted
jobs, but that is primarily a debugging tool and not enabled by default.
If you do see completed jobs when you issue this display command, your
administrator has enabled debugging options. That is mentioned in 14.6.

The displayed status for each job will be one of
    SCHEDULED FOR        = waiting for its time to run
    DEPENDENCY WAIT      = time to run has passed but waiting
                           on dependencies
    FAILED:SEE ALERTS    = job has failed
    PENDING (REQUE WAIT) = job or scheduler conditions prevented execution
                           execution at the ideal time, requed to 5 minutes
                           later
    HOLD IS ON           = job has been held and will not be processed
                           until released from hold
    EXECUTING (PID=nnnn) = job is running, the pid of the job is shown.

If the full details display option is on then any dependencies a job is
currently waiting on will also be shown in this display.
```

9.2 Forcing a job to RUNNOW

```
This is for impatient operators and system managers. If a job is in
time-wait for 23:00, and its currently 19:00, and all dependency waits
have been satisfied; they may not want to hang around until 23:00 for
it to run.

SCHED RUNNOW jobname

where jobname is the name of the job on the active queue will alter the
queued jobs next runtime to the current time. It will then be scheduled
on as soon as possible.

Considerations
* A job waiting on dependencies cannot be RUNNOW. See the dependency
```

  section on how to delete dependencies if this is required.
* A job on the alert queue cannot be RUNNOW. If you wish to restart a
  failed job use the ALERT RESTART command.
* A held job cannot be RUNNOW, turn hold off for that to run.
* An executing job cannot be RUNNOW, as its already running
* A completed/deleted job cannot be RUNNOW. It has done its dash for the
  day.
* A REPEATING JOB, if issued a runnow will run immediately BUT WILL ALSO
  RESCHEDULE itself back onto the time it was next due to run also. It will
  not skip a cycle as repeating jobs schedule back onto the next nn interval
  expected in the future reguardless of how many times they have been
  manually forced to run within that interval.

Note: If EXECJOBS is set to off then the job will be rescheduled to the current time assuming it's not prevented from doing so by any of the considerations above; however no jobs will run while EXECJOBS is off so it will remain on the ready queue. See section 10.3 and 10.4 on the EXECJOBS flag useage.

9.3 Placing a job into a hold state

There may be a reason you do not want a job to run at the scheduled time. One of the developers may have found a bug and want you to hold off for a few hours while they move in a new program for example.

The command

```
SCHED HOLD-ON jobname
```

where jobname is the name of a job on the active queue will set the job hold flag on. This will prevent the job from running until the hold flag is turned off again.

9.4 Releasing a job from a hold state

If a job has been placed in the hold queue as described in section 8.3 you can use the command

```
SCHED HOLD-OFF jobname
```

where jobname is the name of the job on the active queue currently in hold state that you wish to make available for scheduling on again.

9.5 Deleting a job from the active queue

This command should be used with extreme caution.
Deleting a job from the active queue is just that, it is deleted and may be resubmitted. IT IS STILL SCHEDULED FOR THE CURRENT DAY as far as the jobs database is concerned.
This command should only be used if you also intend to permanently delete the currently existing job record from the job definition database once you have deleted it from the scheduler queue.

```
If a job is deleted it is important to note...
* any alerts for the job will also be deleted
* any jobs that are dependant upon the job being deleted will immediately
  have their depencencies satisfied so they can run

What will NOT happen, is that the jobs last-run-time and next-run-time
timestamps will not be updated. This is the correct handling, as the job
has not run.

This means that unless the job is manually submitted back on at some time
during the current days processing it will
1) be automatically scheduled on again in the next newdays processing as
   its next runtime indicates it needs to be run; this may fail as
   calendar checking may block it (when implemented).
2) If it is a daily job, until it is at some point manually scheduled on
   to run twice in a day its next-run-time will always be one day behind.
```

To summarise the above two points, unless you are deleting the job from the scheduler active queue with the intent to also immediately delete it from the jobs file, don't do it !.

```
The command syntax is

SCHED DELETE jobname

where jobname is the name of the job on the scheduler active job queue.
```

10. Scheduler management commands

10.1 Displaying the server status

The server internal settings and configuration flags can be displayed using the command

```
SCHED STATUS
```

This returns information such as server version, hostname it is licensed for, the license key and company information; and more importantly... whether the server is enabled to run jobs, what the log level is set to, what debugging levels are in effect, is the server is pending a shutdown and so forth.

10.2 Displaying connected sessions with the server

This command is more a debugging tool than a provider of usefull information. It lists the current connections to the server. I suppose suppose it could be usefull if you were trying to track which machines were connected to the scheduler without enabling debugging.
It also displays the access level of the connected session. This will be one of "A" (Admin), "S" (Security), "O" (Operator), "J" (JobAuth) or "0" (Guest/browse-only).

The command is

```
SCHED SHOWSESSIONS
```

Example:

```
command:SCHED SHOWSESSIONS

127.0.0.1:32778 default (access level A)
```

10.3 Preventing further jobs from executing

Only a user logged on with Administrator privilage can use this command.

There may be periods where for whatever reason you wish to suspend job scheduler activity.

The scheduler command

```
SCHED EXECJOBS OFF
```

will quiesce the job scheduler. All currently executing jobs will be allowed to complete, but no new jobs will be permitted to start running.

Note: This setting since V1.10 is now remembered through server restarts.

10.4 Allowing jobs to start executing (reverse of 10.3)

Only a user logged on with Administrator privilage can use this command.

If you have quiesced the server, jobs may be pernitted to start executing again with the command

```
SCHED EXECJOBS ON
```

which will allow the server to resume scheduling on jobs.

10.5 Shutting down the server normally

Only a user logged on with Administrator privilage can use this command.

There will invariable be times when you need to shutdown the server, and let me say now the 'kill' command is not the best way.

There are two ways to shutdown the server, the bad way is covered in the next section.

The prefered way for the server to be shutdown is to use the command

```
SCHED SHUTDOWN
```

which brings the server down cleanly as follows
* no more jobs are permitted to start running (for worried operations staff the SCHED STATUS command will show the server is shutting down)
* all currently running jobs are allowed to complete, so joblogs and scheduler databases are all in sync.
* when the last running job completed the server will shutdown cleanly.

Considerations
This is the preferred method for shutting down the server, however it must be noted that as it waits for all jobs to complete running this could take a very long time, depending on how long your batch jobs run.
It should still be the only way you normally shutdown the server.
There is a faster way, but that should be reserved for system reboots (see below section).

10.6 Shutting down the server with a FORCEDOWN

Only a user logged on with Administrator privilage can use this command.

DON'T !. This function is reserved for use within system shutdown scripts.

This command is used by the supplied /etc/rc.d/init.d script to shutdown the job scheduler server when the runlevel is switching to single user, power down or reboot. It should never be manually entered.

This command is provided, and exists, simply because in a system shutdown or reboot situation the system is unlikely to hang around waiting for long running batch jobs to complete.

The command is designed to ensure the integrety of the job scheduler databases, it doesn't take into account any horrible effects on batch jobs that may occur simply because if the system is changing runlevels for a shutdown it's going to kill the jobs anyway.

```
What the FORCEDOWN option of the scheduler does is
* Issues a kill to all tasks (jobs) it spawned. This will not stop any
  child tasks started by scripts or programs the scheduler spawned but
  will kill the schedulers immediate child so it can be processed.
* The scheduler switches itself to a shutting down state
* The scheduler resumes normal shutdown operations which are
  - catch the completion of the child processes
  - process the completion of the child processes, these will all be
    placed into an alert event state of 'signal killed'
  - shutdown after the last child has reported it has been signal killed.
```

The happy outcome of this is that the scheduler knows all jobs it was managing were abnormally terminated, and best of all it gets to close its databases in a normal manner.

The unhappy side effect is that any subtasks spawned by child programs or scripts will continue running and unless it is a system shutdown sutuation will probably complete running quite happily, so this command should only be used in a real system shutdown, or runlevel change, and only from the supplied /etc/rc.d/init.d script. Please don't manually run it.

The command is (never to be manually used please)
```
SCHED SHUTDOWN FORCEDOWN
```

```
Considerations
* As all running jobs are flagged into alert state they must be MANUALLY
  checked when the system is restarted. Visual inspection is required to
  determine whether the appropriate action for the job will be an
  ALERT RESTART or an ALERT FORCEOK.
```

```
Notes: If the machine hangs/freezes or is fast booted etc, where the
       /etc/rc.d/init.d script is not run to shut the server down, a logic
       check at server startup time will check for jobs that are in the
       'executing' state, which as the server is only just starting up
       should not be possible as it has started no jobs, and
       immediately flag them into alert state. In that case the
       considerations above also apply.
```

11. Important Scheduler configuration commands

11.1 Adding a new license key

Only a user logged on with Administrator privilage can use this command.

The command syntax is (to be entered all on one command line)

```
SCHED LICENSE COMPANY companyname,SERVERNAME servername,
      EXPIRES YYYYMMDD,KEY hexchars
```

This has been obsoleted, the application is no longer shareware but free software.

However please note that if you copy all the files to another server and try to run the scheduler it will fail (the servername is set at install time and cannot be changed, so will not run on another server (or if you change the server name)). Just re-install to fix.
Yes there is a much easier way that takes seconds; but as it's now free there is also no longer any support provided, and if you get the easy way even slightly wrong you'll need support. So just re-install if you change the server name or copy the files to another server.

Any attempt to change the licence will fail.
It should allow company and servername to be changed but there is an outstanding bug to fix there.

11.2 Changing the scheduler job 'catchup' option

Only a user logged on with Administrator privilage can use this command.

The scheduler catchup flag is used to determine how the scheduler manages prolonged scheduler downtime, r jobs that have been deleted from processing for a given day.

Basically...

1) If the catchup flag is 'ALLDAYS', then the scheduler newday will repeatedly run for each days processing that has been missed, until eventually it reached the current days date.
2) If the catchup flag is 'NONE', the scheduler newday will run in the current batch day, then be normally rescheduled to the next time it would run after the current days date.

As the scheduler does not currently adjust system times (and does not intend to) the 'alldays' flag should only ever be used if ALL your jobs perform their own time management via control files. This is not actually too hard as instead of scripts doing a date request they could cat a file with a rundate in it, and have the last batch job in a daily stream change it; but I'm leaving that up to how you want to run it.

The default at install time is 'ALLDAYS'.
The recommendation for non-production servers is 'NONE'.

```
  To change the catchup flag value use one of the commands
    SCHED CATCHUP ALLDAYS
    SCHED CATCHUP NONE
```

11.3 Changing the scheduler newday time

Only a user logged on with Administrator privilage can use this command.

Once a day the system task SCHEDULER-NEWDAY will run to clean up data files and submit onto the active queue all jobs that are intended to run over the next 24hours. The time this job runs should be set to after the batch window is expected to complete.

```
SCHED NEWDAYTIME hh:mm
```

Considerations
The existing newday job will be deleted and a new one submitted.
* If the newday time selected is after the current time the newday job will be scheduled on for the current day, at the provided newday time.
* If the newday time selected is prior to the current time the newday job will be scheduled for the time selected on the following day.

11.4 Changing how the scheduler handles newday conflicts

Only a user logged on with Administrator privilage can use this command.

The scheduler internal job SCHEDULER-NEWDAY can only run after all jobs for the current batch day have completed. If there are still batch jobs on the active queue running or waiting to run at the time the newday is scheduled to run the newday job will do one of two things depending on how the newday 'pause action' flag is set.

1) alert, the scheduler-newday job will place itself into an alert status and wait for a manual 'alert restart' command when operations decide it can be restarted.
2) depwait, it will add one of the jobs waiting to run, or currently executing onto its dependency list and go into a dependency-wait state. When released it will check again and another job to its deoendency list as appropriate until the last job on the scheduler active queue has completed. This method is for a 'hands-off' site where the batch running late is not considered to be something that needs looking into.

The commands to change this setting are...

```
SCHED NEWDAYPAUSEACTION ALERT
SCHED NEWDAYPAUSEACTION DEPWAIT
```

The default when installed is DEPWAIT.

12. Alert Forwarding

12.1 Overview of function

```
Obviously we all monitor alerts from all our distributed servers at
one central point. So methods to provide any job scheduler alerts to
be passed to other applications can be configured.

The job scheduler will allow you to run external scripts to
process and forward alerts, so you can for example setup a
script to forward the alerts to Tivoli by running a script to
format a command to the Tivoli postemsg program, or any other
tool you are currently using for centralised alert monitoring.

Basically
  Forwarding off - no alerts are forwarded
  Forwarding on  - alerts are forwarded natively across tcpip sockets,
                   only usefull if you have some of my other products
                   to receive and process the raw data; don't use this.
                   The option has been removed from the shipped release.
  Forwarding externalcmd - will run the scripts you select on a raise
                   alert or cancel alert event, GIVING YOU FULL CONTROL
                   OF WHERE THEY GO.
```

12.2 Selecting the alert forwarding mode

This is done with the SCHED ALERT FORWARDING ACTION command.
The options to the command are

```
            ON - use my proprietary native tcp-ip protocol (most
                 efficent, but not portable; do not use this unless
                 on a site I support, it is also being phased out).
                 Functionality removed from the shipped release.
           OFF - do no alert forwarding (the default, more efficient
                 but no alert forwarding)
   EXTERNALCMD - run an external script to do the forwarding, allowing
                 you to use any vendors product to forward alerts.
                 This you would use if you already have centralised
                 monitoring in place (and your alert application has a
                 command line interface). This has the most overhead as
                 a fork must be done for each alert, however if your
                 batch jobs run correctly alerts will be a rare
                 occurence.
```

```
Syntax: SCHED ALERT FORWARDING ACTION ON|OFF|EXTERNALCMD
```

See section 12.5 on how to specify the actual scripts to be run when using the external command feature.

12.3 Using no alert forwarding

This is the default value when installed. With no alert forwarding in effect all the alerts are stored locally only. In small sites this is unlikely to be an issue as the supplied sample web pages show you how to view alerts across multiple systems, although they have a high overhead. Alerts are only viewable through the job scheduler command (or web page) interfaces.

If no alert forwarding is used you can skip this section, as none of the forwarding parameters will affect you.

To turn alert forwarding off (the default), from an administration user signon enter the command

```
    SCHED ALERT FORWARDING ACTION OFF
```

12.4 Using native alert forwarding

Disabled in this release. Being removed from the code as being too proprietory to be usefull to anybody else.

12.5 Forwarding alerts using external shell scripts

This is the prefered method of forwarding alerts as it allows you full control of where they are sent, and allows you to forward the alerts to existing alert monitoring systems you employ at your site.

You specify the shell script or command to be run when an alert is raised, and when an alert is cancelled. The shell scripts are passed the name of the job involved in the alert as $1 and any meaningfull text in $2.

This allows you to fully customise how you handle alerts from the scheduler, and allows you to forward on both alert events and alert cancellation events to any monitoring tools you have in place that allow a command line interface to be run.

Sample scripts are provided with the job scheduler that invoke the 'logger' program to write them to the syslog task (go to /var/log/messages). These will be in the directory you installed the job scheduler application to under the 'alert_tools' directory. You should customise these to use the alert management tool your site already uses.

If you chose to use my centralised alert collection tool scripts are provided with that specifically for interfacing with the job scheduler.

To use external scripts to forward alerts you need to define a script to be executed when an alert is raised, and a script to be executed when an alert, is cancelled; then turn them on.

The commands are as below

```
   SCHED ALERT FORWARDING EXECRAISECMD /full/program/path
   SCHED ALERT FORWARDING EXECCANCELCMD /full/program/path
   SCHED ALERT FORWARDING ACTION EXTERNALCMD
```

13. Miscellaneous Commands

13.1 OPEN new server command.

The OPEN command may be used to disconnect from the current server connection and connect to another job scheduler server.

While the jobsched_cmd program itself can be used to connect to remote servers this command is provided to allow connection to remote servers without having to exit the program and start another one; just easier when you are managing multiple servers.

If the optional port number is not provided then the connection will be attempted to the same port number as you initially connected on with the new ip address.

```
OPEN ipaddr [portnumber]

Examples:
  OPEN 127.0.0.1 9002
  OPEN 169.5.6.7
```

13.2 OPEN information.

If the OPEN command is used with the INFO option then the current server connection information will be displayed; justthe ipaddr and port number your current session is connected to.

```
OPEN INFO
```

note: “sched showsessions” is used to display all connected sessions.

14. Server Debugging Facility

14.1 When to debug the server

As a general rule you will only need to debug the server when you are experiencing problems, have raised a fault, and have been asked to turn debugging on while you try to recreate a problem you have exprienced.

As the debuging levels can be set specific to individual library modules it is best to seek advise before turning debugging on so the author can let you know which modules he wants additional information from, turning debugging on all modules will generally produce a trace too large to be of any use and is not recomended.

14.2 How the debugging levels work

The debug levels can be set individually on library components used to build the server, this allows debugging to be switched on for an individual function of the server when trying to narrow down a problem.

There are multiple levels that can be used from 0-9, 9 producing the most information, and 9 should never be used unless you have a spare 5-10gb of disk space for the log file. What each numeric level provides in the current release is shown in section 14.5.

The debug information is written out to the standard log file for the current day.

14.3 Changing the debugging levels at a library level

Only a user logged on with Administrator privilage can use these commands.

To enable debugging messages to be written use the command
DEBUG

The options are explained immediately below, the
values for the current release are documented in 14.5.

```
DEBUG SERVER n      ( server main code block )
DEBUG UTILS n       ( utility library )
DEBUG JOBSLIB n     ( job definition database functions )
DEBUG APILIB n      ( api calls )
DEBUG ALERTLIB n    ( alert queue management )
DEBUG CALENDAR n    ( calendar processing )
DEBUG SCHEDLIB n    ( active queue and dependency queue functions )
DEBUG BULLETPROOF n ( data structure checking library )
DEBUG USERS n       ( user record functions )
DEBUG MEMORY n      ( memory library functions )
```

Notes: The SCHED STATUS command will display what the current log level settings are for each component.

14.4 Globally changing debugging levels

Only a user logged on with Administrator privilage can use these commands.

It is possible to change the log level for all components with a single command, which is...

```
DEBUG ALL n
```
Don't use n=9 with this unless you have a couple of gigabytes free in the job schedulers logging file system.

Personally I find it most usefull to turn debugging off on all modules when I have finshed a selective debugging sesion. I'm not sure why you would ever want to globally turn on debugging, but it is possible.

n is as for section 14.3, a number in the range 0 to 9, with 0 being to turn off debuging.

14.5 Debug levels currently defined

The following debug levels are defined. The higher the number n the more debug information will be displayed, as all values of n below the selected level will also provide debug information.
ie: if loglevel 4 is used data for levels 4, 3, 2 and 1 are logged.
[2003/09/22-added *special* level 5 for memory library, see Note2]

0 = no debugging
1 = proc call level display
2 = IO information (record numbers etc)
4 = variable formatting information
5 = memory only, used only by the MEMORY module (*see note 2 below*)
6 = malloc/free information (***obsolete, see note1 below***)
7 = header dump information
8 = record compare information
9 is everything, including timer proc calls that occur about
  every 1.5 seconds in the server main code, so have a massive
  log file for level 9 (I have filled a 3GB file system in 10 minutes).

The default log level at server installation is 0 for all modules.

Considerations
These log levels are the levels defined at the time this document was written. They are subject to change.

Note1: level 6 has been obsoleted with the relocation of all inline server memory operation to the common memory library. Selecting a setting of 6 for other modules will produce no results as they no longer perform memory operations themselves.
Note2: level 5 is only used by the memory library, if used it will only display memory malloc/free events. This produces a cleaner log than using level 6 on the memory library as it suppresses the logging of proc, IO and variable debug events.

On memory. A "sched status mem" will dump to the scheduler log file the current memory allocation activity since the last newday job ran; logfile example

```
    Sat Mar 19 12:21:06 2011 INFO: MI001-Memory display requested
    Sat Mar 19 12:21:06 2011 INFO: MI005-No memory slots are currently in use.
    Sat Mar 19 12:21:06 2011 INFO: MI003-Using 0 slots, out of 20 total slots, currently malloced 0 bytes in total
    Sat Mar 19 12:21:06 2011 INFO: MI006-Since last newday - 1 malloc and 1 free calls from stack calls
    Sat Mar 19 12:21:06 2011 INFO: MI004-Memory display ended
```

14.6 Displaying why a scheduled job was deleted

Only a user logged on with Administrator privilage can change the default display method configuration entry. This change affects all users (it is a server setting, not a session setting).

Up until this release the command line interface always displayed in a SCHED LISTALL command why a job was considered complete. This may be because it had compleded normally, or usefull for me in testing displays if it was completed while still in dependency wait, in alert state, still in time wait etc.
There may be occasions when you still want to see this information, which is now not displayed by default.

This can be toggled by

```
  SCHED FULLDETAILS ON
  SCHED FULLDETAILS OFF
```

The supplied browser samples handle both as it has been coded to discard the full details info if present in the output. With the toggle ON jobs that have been completed or deleted will be displayed with the SCHED LISTALL command.
If the toggle is OFF only executing or waiting to execute jobs will be displayed.

15. Command Quick Reference

This section briefly lists the syntax for each of the commands. Any command with (*** READ THE MANUAL FIRST ***) should not be taken lightly, refer to the appropriate section of this manual for the impacts and considerations of using the command before actually using it.

```
JOB ADD jobname,OWNER ,TIME ,
        CMD unixcommand[,PARM "parmsforcommand"]
        [,DEP "JOB |FILE " total of five dependencies]
        [,REPEATEVERY nn]
        [,DAYS "MON,TUE...,SUN"][,DESC desc test with no commas]
JOB DELETE jobname     (*** READ THE MANUAL FIRST ***)
JOB INFO jobname
JOB LISTALL [mask]
JOB SUBMIT jobname     (*** READ THE MANUAL FIRST ***)

ALERT ACK jobname
ALERT FORCEOK jobname   (*** READ THE MANUAL FIRST ***)
ALERT INFO jobname
ALERT LISTALL
ALERT RESTART jobname

CAL ADD calname,TYPE NORMAL|HOLIDAY,YEAR YYYY[,HOLCAL name],format-string
CAL MERGE calname,TYPE JOB|HOLIDAY,YEAR YYYY,format-string
CAL SUBDELETE|UNMERGE calname,TYPE JOB|HOLIDAY,YEAR YYYY,format-string
CAL DELETE calname,TYPE NORMAL|HOLIDAY,YEAR YYYY
CAL LISTALL
CAL INFO calname,TYPE JOB|HOLIDAY,YEAR YYYY
      format-string may be one of...
           FORMAT MONTHSDAYS "MM/DD,MM/DD,MM/DD,MM/DD,MM/DD,MM/DD..."
           FORMAT DAYS ALL|JAN|FEB...DEC "DD,DD,DD,DD..."
           FORMAT DAYNAMES "SUN,MON,TUE,WED,THU,FRI,SAT"

DEBUG SERVER n      ( server main code block )
DEBUG UTILS n       ( utility library )
DEBUG JOBSLIB n     ( job definition database functions )
DEBUG APILIB n      ( api calls )
DEBUG ALERTLIB n    ( alert queue management )
DEBUG CALENDAR n    ( calendar processing )
DEBUG SCHEDLIB n    ( active queue and dependency queue functions )
DEBUG BULLETPROOF n ( data structure checking library )
DEBUG USERS n       ( user record functions )
DEBUG MEMORY n      ( memory management library )
DEBUG ALL n

DEP LISTALL
DEP LISTWAIT
DEP CLEARALL JOB jobname   (*** READ THE MANUAL FIRST ***)
DEP CLEARALL DEP depname   (*** READ THE MANUAL FIRST ***)

SCHED LISTALL [jobmask]
SCHED RUNNOW jobname   (*** READ THE MANUAL FIRST ***)
SCHED HOLD-ON jobname
SCHED HOLD-OFF jobname
SCHED DELETE jobname   (*** READ THE MANUAL FIRST ***)
SCHED STATUS
SCHED SHOWSESSIONS
SCHED EXECJOBS OFF
SCHED EXECJOBS ON
SCHED SHUTDOWN
SCHED SHUTDOWN FORCEDOWN (*** READ THE MANUAL FIRST ***)
SCHED LICENSE COMPANY company,SERVERNAME servername,EXPIRES yyyymmdd,KEY hex
SCHED CATCHUP ALLDAYS
SCHED CATCHUP NONE
SCHED NEWDAYTIME hh:mm
SCHED NEWDAYPAUSEACTION ALERT
SCHED NEWDAYPAUSEACTION DEPWAIT
SCHED FULLDETAILS ON
SCHED FULLDETAILS OFF

SCHED ALERT FORWARDING ACTION OFF | EXTERNALCMD
SCHED ALERT FORWARDING EXECRAISECMD /full/program/path
SCHED ALERT FORWARDING EXECCANCELCMD /full/program/path

USER ADD username, PASSWORD password, AUTH ADMIN|OPERATOR|JOB|SECURITY,
    AUTOAUTH YES|NO, SUBSETADDAUTH YES|NO|ADMIN,
    USERLIST "<1>,<2>...<10>",DESC "descriotion text"
USER DELETE username
USER PASSWORD username,newpassword
USER INFO username
```

16. Obtaining Support

Mark Dickinsons job scheduler is now released as free software. As such there is no longer any dedicated support for this application.

Refer to the site you downloaded the software from for user forum information; at the time of this documents creation only my personal forum exists. I am assuming you didn't download it from my website or you would already have access to my user forum. I have no intention of stopping other sites from making my software available; but I also have no intention of supporting anything provided to you by other sites (as it's obviously out of date if you have a problem with it).