Cron Converter

Job Scheduler

Documentation
February 2008

Software- und Organisations-Service GmbH Giesebrechtstr. 15 D-10629 Berlin Telephone +49 30 86 47 90-0 Telefax +49 30 861 33 35

Contact Information

Software- und Organisations-Service GmbH
Giesebrechtstr. 15

D-10629 Berlin

Telephone +49 30 86 47 90-0
Telefax +49 30 8 61 33 35
Mail info@sos-berlin.com
Web http://www.sos-berlin.com

Last Updated: February 2008

Table of Contents

1 Introduction 4
1.1 Requirements 4
2 The Cron Converter Command Line Tool 6
3 The Cron Adapter Job 8
3.1 Using Cron Adapter Job with Central Configuration 9
4 User crontab or System crontab? 11
Appendix A: Parameters 12
A.1 Customizing converted jobs 12
Appendix B: Limitations 13

1 Introduction

Why use the Job Scheduler, when Cron is available in the operating system as a scheduler?

One of the advantages of Cron is ist simple configuration and its use of Unix commands. The Job Scheduler, however, has several advantges over Cron:

- Cron does not write a job history
  Cron starts jobs but does not save any information about either when a job was started or finished, or about the result of the job. It is not possible to create a job history with Cron which provides relevant and updatable information.
  The open source Job Scheduler writes its job history in a database and has a web interface which can be used to provide information about the job history.
- Cron does not write a log.
  Cron ignores the output of a job (stdout/stderr). The saving of log information, therefore, must be built into the job itself and the resulting log files must then be seperately collected and centraly stored.
  The open source Job Scheduler automatically collects job output to stdout/stderr and save the resulting log file centrally in a database - if required, in compressed form.
- Cron works according to the fire & forget principle.
  It offers few possibilities, for reacting to the result of a job. It is possible to implement a job results check in Unix, but this involves considerable effort and is basically a reinventing the wheel process.
  Should an error occur whilst a job is running, the open source Job Scheduler can send an e-mail to the relevant authority and start follow-on jobs to correct the error. Furthermore, it offers a web interface in which allows manual intervention in the job program. The interface shows, for example, job start times and allows jobs to be started, paused and stopped.
- Cron does not recognise job chains.
  The linking of several commands in Cron is not equivalent to the building of a job chain. Job chains recognise dependancy in the individual job steps and can cause the folow-on job to be started to depend on the result of a job.
  The open source Job Scheduler recognises dependancies in job chains and writes the ongoing job states in a database, in order to be able to automatically restart jobs should a system error occur.

The open source Job Scheduler *Cron Converter* provides a migration path for Cron which:

- retains an existing Cron configuration through use of `crontab`
  The Job Scheduler automatically reads an already existing `crontab` and uses this to create a job definition. As with Cron jobs, the Job Scheduler reads `crontab` every 60 seconds.
  Such migrated jobs, do not use all of the features of the Job Schedulers - However, they are recorded in a history, are automatically entered in a central log and can be operated using a web interface.
- converts the `crontab` into the Job Scheduler configuration file format
  In this case `crontab` is used as the starting point for the migration. After migration, the job configuration can be continued using the Job Scheduler graphical interface.

The Job Scheduler provides two simple tools, to ease the conversion from Cron to the Job Scheduler:

- The *Cron Converter* command line tool
- The *Cron Adapter* job

The different purposes and methods of use of the tools is described in the following chapters.

1.1 Requirements

The Cron Adapter job and the Cron Converter commandline tool are both bundled in sos.scheduler.jar, which itself is included in the Job Scheduler distribution.

Note that a Java Runtime Environment version 1.4 or higher is required for both of these tools.

2 The Cron Converter Command Line Tool

The *Cron Converter* commandline tool is meant to be used for a once-and-for-all migration from a `crontab` file to a Job Scheduler XML configuration file. It analyses a `crontab` file and generates a corresponding XML file. This XML file can then be included in the Job Scheduler configuration file `./config/scheduler.xml` by using: `<base file = "my_cron_configuration.xml"/>`.

The *Cron Converter* is started with the `cronconverter.sh` script, which can be found in the Job Scheduler `./bin` installation directory.

The parameters which can be used in the command line tool are described in the appendix (page 12).

Example: Using the command line tool

The `/home/scheduler/scheduler/crontab` file contains:

```
5   *   * * *    /usr/bin/message.sh
00  */2 * * *    /usr/local/bin/mail_poll
59  23  * * 0    cp /var/log/messages /log/backup/messages
```

The `crontab` can be converted to an XML configuration file by calling:

```
bin/cronconverter.sh -crontab crontab -target config/my_scheduler_cron.xml
```

from the Job Scheduler's home directory (e.g. `/home/scheduler/scheduler`).

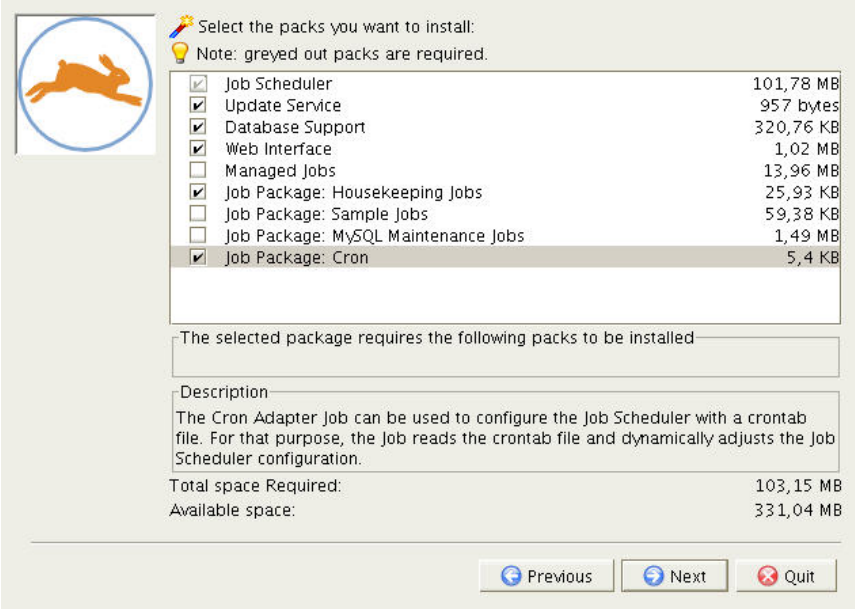
This results in the creation of a configuration file named `config/my_scheduler_cron.xml` with the following contents:

```
<?xml version="1.0" encoding="UTF-8"?>
<spooler>
 <config>
   <jobs>
     <job name="usr_bin_message.sh" timeout="600"
         title="Cron Job /usr/bin/message.sh">
       <script language="shell">/usr/bin/message.sh</script>
       <run_time>
         <period absolute_repeat="01:00" begin="00:05"/>
       </run_time>
     </job>
     <job name="usr_local_bin_mail_poll" timeout="600"
         title="Cron Job /usr/local/bin/mail_poll">
       <script language="shell">/usr/local/bin/mail_poll</script>
       <run_time>
         <period absolute_repeat="02:00" begin="00:00"/>
       </run_time>
     </job>
     <job name="var_log_messages" timeout="600"
         title="Cron Job cp /var/log/messages /log/backup/messages">
       <script language="shell">
         cp /var/log/messages /log/backup/messages
       </script>
       <run_time>
         <weekdays>
           <day day="0">
             <period single_start="23:59"/>
           </day>
         </weekdays>
       </run_time>
     </job>
   </jobs>
 </config>
</spooler>
```

3 The Cron Adapter Job

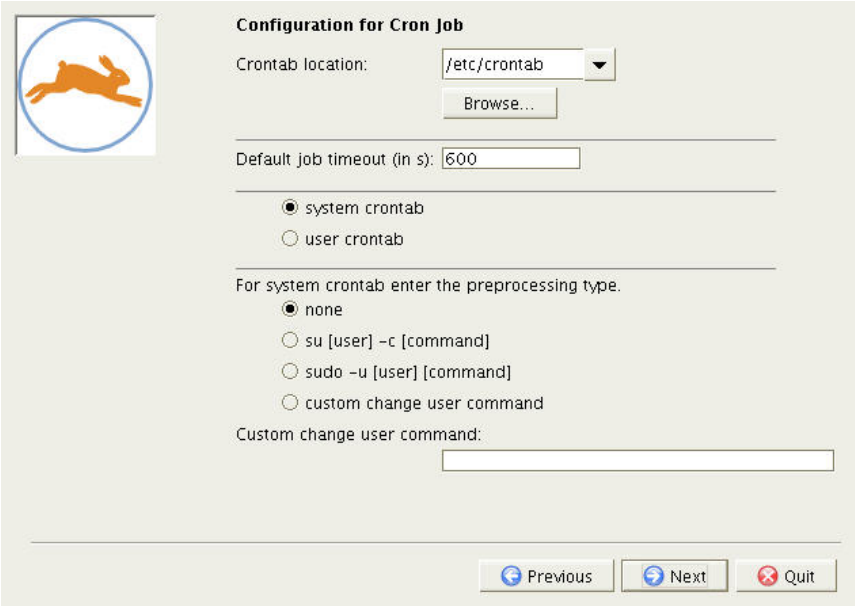
The *Cron Adapter* job can be used to configure the Job Scheduler with a `crontab` file. For that purpose, the job reads the `crontab` file and dynamically adjusts the Job Scheduler configuration. Note that scuh changes are not permanent - they will be lost after a restart of the Job Scheduler.

The Cron package has to be choosen during the Job Scheduler setup for the *Cron Adapter* job to be available:



Note that more detailed information about installing the Job Scheduler - in particular about the installation of new packages after the Job Scheduler itself has been installed - can be found in the Job Scheduler Installation document.

The setup queries the parameters for the job (path to `contrab`, type of `crontab`, ...) and configures the job in the `./config/scheduler_cron.xml` configuration file.



The setup will then the generate the following job configuration:

```
<job name           = "scheduler_cron_adapter"
     title          = "Read crontab"
     spooler_id     = "">

  <description>
    <include file = "jobs/JobSchedulerCronAdapter.xml"/>
  </description>

  <params>
    <param name="crontab" value="crontab"/>
    <!-- 0 for User crontab, 1 for system crontab -->
    <param name="systab" value="0"/>
    <!-- su, sudo or custom command, empty for none -->
    <param name="changeuser" value=""/>
    <param name="timeout" value="600"/>
  </params>

  <script language      = "java"
          java_class    = "sos.scheduler.cron.JobSchedulerCronAdapter"/>


  <run_time let_run = "yes"
            repeat  = "120"
            begin   = "00:00"
            end     = "24:00"
            once    = "yes"/>

</job>
```

The parameters which can be used for a *Cron Adapter* job are described in the appendix (page 12).

3.1 Using Cron Adapter Job with Central Configuration

The Cron Adapter Job may also be used to distribute cron jobs from a supervisor Job Scheduler to "Workload" Job Schedulers using central configuration.

This can be achieved with the follwing steps:

- One Job Scheduler must be installed as a central supervisor
- The "Workload" Job Schedulers have to be configured to register with the central supervisor (The "Workload" Job Schedulers don't need the cron package).
- Directories for the configuration of the workload Job Schedulers are created within the configuration directory of the supervising Job Scheduler. (See the "Central Configuration Using a Supervising Job Scheduler" chapter in the Job Scheduler reference documentation.
- The `scheduler_cron_remote.xml` element should be included alongside the `scheduler_cron.xml` in the `scheduler.xml` configuration file using `<base>`.

The chrontab files can now be added to the sub-directories of the host-specific configuration directories and the chrontab file jobs will be distributed to the corresponding workload Job Schedulers. Note that a special sub-directory (e.g. `cron`) should be created in each host directory (the directory corresponding to a workload Job Scheduler) for the configuration files, as files landing directly in the host directory will deleted.

Example: crontabs für Job Scheduler auf host2 und host3
A Job Scheduler on host1 should convert and distribute cron Jobs for host2 and host3. If all Job Schedulers are configured as described above then the workload Job Schedulers will be configured using crontab files saved in the following directories:

```
config/remote/host2#4444/cron/crontab - contains cron jobs for host2
config/remote/host3#4444/cron/crontab - contains cron jobs for host3
config/remote/_all/cron/crontab - contains cron jobs for all workload Job Schedulers
```

The `cron_adapter_dynamic_configuration_dir` parameter in `scheduler_cron_remote.xml` can be modified in order to activate the chron distribution only for individual workload Job Schedulers. Here, the directories are specified (separated by semi-colons) in which `crontab` files are to be sought. Note that sub-directories within directories specified here will also be automatically monitored for chrontab jobs.

4 User crontab or System crontab?

There are two kinds of `crontab` files: `user crontab` and `system crontab`.

A `user crontab` file has 5 columns for the configuration of the run time and one column for the command.

A `system crontab` file has 5 columns for the configuration of the run time, one column for the user who should execute the command and one column for the command itself.

It is necessary, for both the *Cron Converter* and the *Cron Adapter* tools, to specify whether the `crontab` file is to be configured for `user crontab` or for `system crontab` files. This is done using the `systab` parameter.

The commands of a `user crontab` file will be executed by the user running the Job Scheduler.

For a `system crontab` file, the user may be changed. By default, commands will be executed by the user running the Job Scheduler.

To execute commands with the users configured in the `system crontab` file, the `changeuser` parameter needs to be set to choose a command to change the user. If `changeuser` is set to `su` the Job Scheduler will execute the command as:

```
su $SCHEDULER_CRONTAB_USER -c command
```

(The $SCHEDULER_CRONTAB_USER environment variable contains the user name from the 6th column of the `system crontab`). Note that this only works if the Job Scheduler is running as `root`. If another user is running the Job Scheduler, the `su` command will open a prompt for a passwort, which cannot be answered by the Job Scheduler.

If the Job Scheduler needs to run as a user other than `root` and also needs to execute the `system crontab` commands as another user then sudo can be used. Therefore the `changeuser` parameter has to be set to `sudo`. In addition, `sudo` needs to be configured using the `/etc/sudoers` file so that the user running the Job Scheduler is allowed to execute scheduled commands as another user without a password.

Example: Scheduler user 'scheduler' may run ls as user 'test'
scheduler localhost = (test) /bin/ls NOPASSWD

If knowledge of `sudo` is not available, we strongly recommend reading a `sudo` tutorial or the *sudoers manpage* first.

Appendix A: Parameters

The *Cron Converter* commandline tool and the *Cron Adapter* Job are configured using the following paramters:

Converter parameter	Job parameter	Description
-crontab	crontab	Path and filename of the `crontab` file
-target	---	Target file (XML) for the conversion result
-systab	systab	Configures if the current `crontab` is a `system crontab` or `user crontab`: • *0*: `User crontab` • *1*: `System crontab` The default is *0*. If parameter crontab is "/etc/crontab" then the default is *1*.
-changeuser	changeuser	Sets the command to change the user when using a `system crontab`: • su: execute the command using su • sudo: execute the command using sudo • A custom command may be entered (using $SCHEDULER_CRONTAB_USER). If no command is set then the user will not be changed.
-timeout	timeout	Sets a timeout (in seconds) for the execution of the converted jobs. (Default: 600)
-v	---	loglevel [0=info] [1=debug1]...[9=debug9]

See the 'The Cron Adapter Job' chapter for an example configuration of this job.

A.1 Customizing converted jobs

The conversion result can be modified using `crontab` comments. Comments can be used to set job name, job title and job timeout parameters. These parameters are set using a comment containing one of the following patterns:

- # job_name = name of the job
- # job_title = title of the job
- # job_timeout = timeout (in seconds) of the job

needs to be placed before the cron job.

Example: set name of the job to my_cron_job

```
# job_name = my_cron_job
# other comment(s)
0 * * * * ls -la
```

Appendix B: Limitations

The 5 columns for runtime configuration in a `crontab` file offer a large number of combinations. Most of these can be converted by the *Cron Converter*. However, there are limitations for certain combinations which are listed below:

Cron Runtime	Comment
`*/2 */3 * * *`	A repeat interval is given for minutes and hours. So the job should run every 3 hours at 2 minute intervals. This combination is not supported, as it does not make any sense.
`0 10 */3 * *`	Start the job every 3 days at 10:00. Repeat intervals of several days are not supported. However, the converter will generate a runtime with fixed mothdays, which have the configured offset. In this case 1,4,7,10 ...
`0 10 1 */3 *`	Start the job on the 1st day of the month, every 3 months, at 10:00. The same is true for months as for days. Fixed months will be generated with the configured offset.
`0 10 * * */3`	Start the job every 3 weekdays at 10:00. Repeat intervals for weekdays are not supported. Repeat intervals in the 3rd column (monthdays) can be used instead.
`0 10 13 * 5`	Start the job every Friday the 13th at 10:00. The combination of a fixed weekday with a fixed monthday is not supported. The weekday column will only be evaluated if the monthday column is set to *.