CrontabFile Converter

Users Manual

July 2014

Contact Information

Software- und Organisations-Service GmbH

Giesebrechtstr. 15
D-10629 Berlin
Germany

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

Last Updated: 07/18/2014 12:01 PM

This documentation is based on JobScheduler Version 1.7.4169.

Copyright © 2005-2014 SOS GmbH Berlin.

All rights reserved. All trademarks or registered trademarks are the property of their respective holders. All information and materials in this book are provided "as is" and without warranty of any kind. All information in this document is subject to change without further notice.

This product includes software developed by the Apache Software Foundation (http://apache.org/)

We would appreciate any feedback you have, or suggestions for changes and improvements; please forward your comments to info@sos-berlin.com.

Table of Contents

1 Abstract

This document describes the ways to migrate a batch configuration based on Cron, into the world of JobScheduler. This is not a one-hundred-percent conversion. Some issues will be left and will require a manual justification. More Information about the Limitations can be found in ' ' . Basically, there are the possibilities:

With this conversion, the definitions in the crontab are migrated once in JobScheduler objects. Jobs, job chains and orders can be created. These object are persistend and should be stored in the live folder. The migration is carried out with the JobScheduler Crontab File Converter.

Using this type of migration the crontab file will be analyzed by a periodically running batch job, the JobScheduler Crontab Adapter Job job. The crontab job definitions are during the job run translated into JobScheduler objects that are transmitted to the JobScheduler dynamically. The objects are non-persistent, meaning that if you reboot JobScheduler these objects are no longer available until the JobScheduler Crontab Adapter Job job is running again.

On the one hand are the two basic options are presented and the other their advantages and disadvantages are demonstrated.

Because there is the possibility of using cron only in the Unix® and Linux® operating systems, the subject of this documentation only is this type of operating system.

2 Introduction

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

One of the advantages of Cron is the easy and simple configuration and its use of Unix® commands. The JobScheduler, however, has several advantages over Cron:

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 JobScheduler writes its job history in a database and has an user interface, JOC, which can be used to get information about the job history.
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 JobScheduler automatically collects job output to stdout/stderr and save the resulting log file centrally in a database - if required, in compressed form.
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 JobScheduler can send a notification, e.g. an e-mail, to the relevant authority and start follow-on jobs to handle the error. Furthermore, it offers an interface, JOC, in which allows manual intervention in the job. The interface shows, for example, job start times and allows jobs to be started, paused and stopped.
The concatenation 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 follow-on job to be started to depend on the result of a job.

The JobScheduler support dependencies in job chains and writes the ongoing job states in a database, in order to be able to automatically restart jobs should an error occur.

The JobScheduler Crontab File Converter provides a migration path for Cron which:

The JobScheduler automatically reads an already existing crontab and uses this to create a single JobScheduler object or more than one object. As with Cron jobs, the JobScheduler processes the crontab file every 60 seconds.

Such migrated objects do not use all of the features of the JobScheduler - however, they are recorded in a history, are entered in a central log and can be operated using JOC.
In this case the crontab file is used as the starting point for the migration. After migration, the configuration of objects (jobs, orders, job chains) can be continued using JOE.

The JobScheduler provides two tools, to ease the conversion from a job configuration defined in a Crontab to the configuration files which are used by JobScheduler:

The JobScheduler Crontab File Converter commandline tool is meant to be used for a once-and-for-all migration from a crontab file to JobScheduler configuration files.

This gives you an easy method of migrating all your existing Unix® jobs from various Unix® flavors into JobScheduler for a single point of management. All you need is your crontab configuration file, and JSJobScheduler Crontab File Converter.
The JobScheduler Crontab Adapter Job is used to configure the JobScheduler with a crontab file.

The different purposes and methods of use of the JobScheduler Crontab File Converter and JobScheduler Crontab Adapter Job is described in the following chapters.

3 Requirements

The JobScheduler Crontab Adapter Job and the JobScheduler Crontab File Converter are both bundled in sos.scheduler.jar, which itself is included in the JobScheduler distribution for Unix® or Linux® systems. The distribution of JobScheduler for other operating systems does not contain this components.

Note that a Java® Runtime Environment (JRE) version 1.7.x or higher is required for both of these tools.

4 The JobScheduler Crontab File Converter Command Line Tool

The JobScheduler Crontab File Converter commandline tool is meant to be used for a once-and-for-all migration from a `crontab` file to a JobScheduler configuration file. It analyses a `crontab` file and generates corresponding configuration files for the JobScheduler. This configuration files can then be stored in the live folder.

The JobScheduler Crontab File Converter is started with the `cronconverter.sh` script, which can be found in the JobScheduler `./bin` installation directory.

The parameters which can be used in the command line tool are described in the chapter parameters (page 19).

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

Example: the content of a simple crontab file

The /home/scheduler/scheduler/crontab file contains the lines shown above.

This crontab is converted to a JobScheduler configuration file by calling:

```
bin/JobScheduler Crontab File Converter.sh -crontab crontab -target config/my_scheduler_cron.xml
```

Example: starting the JobScheduler Crontab File Converter commandline tool

The tool is reading the crontab from the JobScheduler's home directory. The result is a configuration file named config/my_scheduler_cron.xml with the following content:

```
<?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>
```

Example: Result of a converted crontab

1 Description: SOSCrontabFileConverter

The SOSCrontabFileConverter can be used to migrate a crontab file in the objects of JobScheduler.

1 Parameters

List of parameters

Name	Title
base_directory	
create_sub_folder_structure	
create_job_chain_jobs	
create_a_mock	
mock_command	
crontab	
systab	
changeuser	
timeout	

1.1 Base_Directory - Base Directory

Mit dem Wert dieses Parameter wird ein Verzeichnis-Name festgelegt, der bei der Speicherung der migrierten Objekte als Basis-Verzeichnis angewendet werden soll.

Data-Type: SOSOptionFolderName

The default value for this parameter is .

1.2 Create_Sub_Folder_Structure - Create SubFolder Structure

```
00 21 * * 1-5 su - root -c "\${DIR_BIN}/batch/sel_memorybits.sh
```

Example: An Example using a subfolder structureEin Beispiel für ein Unterverzeichnis

The object "sel_memorybits.sh.job.xml" will be stored in the folder "batch". If the parameter base_directory is set to "prod" then the structure should be "prod/batch/".

Data-Type: SOSOptionBoolean

The default value for this parameter is **false**.

1.3 Create_Job_Chain_Jobs - Create JobChain Jobs

If the value of this parameter set to "true", so instead of "stand alone" jobs are generated those jobs in a job chain allowed to run. Additionally generated for each job is an order and a job chain, which can be started with the order.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<job order="yes" timeout="600"
     title="Cron Job su - root -c &#34;\${DIR_BIN}/batch/sel_memorybits.sh ">
   <extensions>
          <extension xmlns="www.sos-berlin.com/schema/joe">
            <generator date="2012-09-21 16:50:32" name="CronConverter" vendor="www.sos-berlin.com">
              <docu>00 21 * * 1-5 su - root -c "\${DIR_BIN}/batch/sel_memorybits.sh </docu>
            </generator>
            <comment/>
          </extension>
        </extensions>
   <script language="shell"><![CDATA[
echo created by CronConverter, at 2012-09-21 16:50:32
echo mock-mode: su - root -c "\${DIR_BIN}/batch/sel_memorybits.sh
exit 0
]]&lt;</script>
</job>
```

Example: An example for a job

```
<?xml version="1.0" encoding="iso-8859-1"?>
<job_chain>
   <extensions>
          <extension xmlns="www.sos-berlin.com/schema/joe">
            <generator date="2012-09-21 16:50:32" name="CronConverter" vendor="www.sos-berlin.com">
              <docu>00 21 * * 1-5 su - root -c "\${DIR_BIN}/batch/sel_memorybits.sh</docu>
            </generator>
            <comment/>
          </extension>
        </extensions>
   <job_chain_node error_state="!error" job="sel_memorybits.sh" next_state="success" state="100"/>
   <job_chain_node state="success"/>
   <job_chain_node state="!error"/>
</job_chain>
```

Example: An example for a job chain

```
<?xml version="1.0" encoding="iso-8859-1"?>
<order title="base/batch/sel_memorybits.sh">
   <extensions>
          <extension xmlns="www.sos-berlin.com/schema/joe">
            <generator date="2012-09-21 16:50:32" name="CronConverter" vendor="www.sos-berlin.com">
              <docu>00 21 * * 1-5 su - root -c "\${DIR_BIN}/batch/sel_memorybits.sh </docu>
            </generator>
            <comment/>
          </extension>
        </extensions>
   <run_time>
          <weekdays>
            <day day="1">
              <period single_start="21:00"/>
            </day>
            <day day="2">
              <period single_start="21:00"/>
            </day>
            <day day="3">
              <period single_start="21:00"/>
            </day>
            <day day="4">
              <period single_start="21:00"/>
            </day>
            <day day="5">
              <period single_start="21:00"/>
            </day>
          </weekdays>
        </run_time>
</order>
```

Example: An example for an order

Data-Type: SOSOptionBoolean

The default value for this parameter is **false**.

1.4 Create_A_Mock - Create A Mock: simulate the job execution

If the value of this parameter set to "true", so the script element of the job is filled with a command to be specified. The parameter mock_command has to be used to specify the MOCK-command. This command is then executed in the execution of the job, instead of the "real" command. The option in the crontab file command is also added as a comment in the script element.

With this setting, the execution of all created objects are simulated (MOCK-mode).

Data-Type: SOSOptionBoolean

The default value for this parameter is **false**.

Use together with parameter: Mock_Command

1.5 Mock_Command - Mock Command

With this parameter, the command is defined, which is to be performed in mock mode.

Data-Type: SOSOptionCommandString

The default value for this parameter is **ping -n 20 localhost**.

Use together with parameter: Create_A_Mock

1.6 crontab - Name of a crontab file to convert

Path and filename of the crontab file which has to be converted.

Data-Type: SOSOptionInFileName

The default value for this parameter is **crontab**.

This parameter is mandatory.

1.7 systab - Type of crontab file to convert

Sets if the current crontab is a system or user crontab:
0: User crontab 1: System crontab

The default is 0. If parameter crontab is "/etc/crontab" then the default is 1.

Data-Type: SOSOptionBoolean

The default value for this parameter is **false**.

1.8 changeuser - Command for switch user

Sets the command to change the user, if a system crontab is converted. See documentation.

Data-Type:

1.9 timeout - Maximum runtime of a job

Sets a timeout (in seconds) for the execution of the converted jobs.

Data-Type:

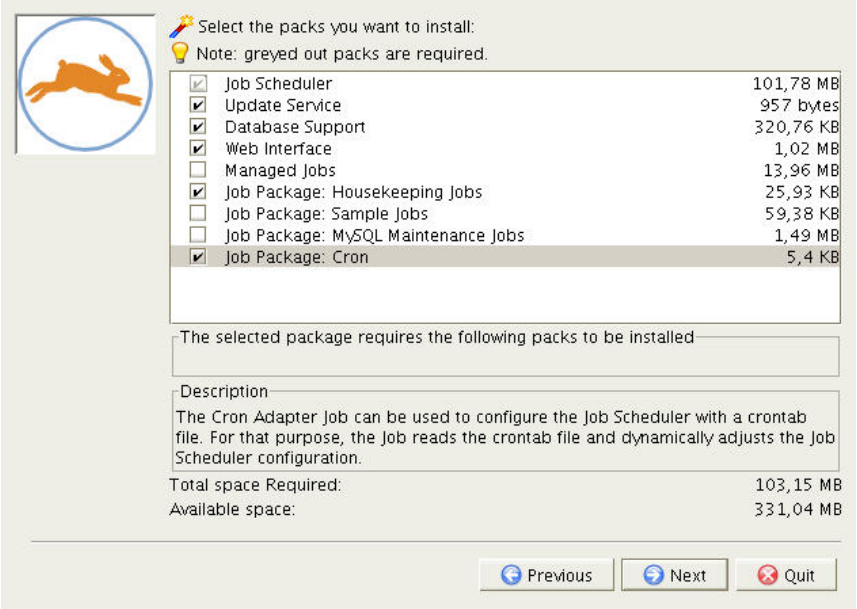
The default value for this parameter is **600**.

5 The JobScheduler Crontab Adapter Job

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

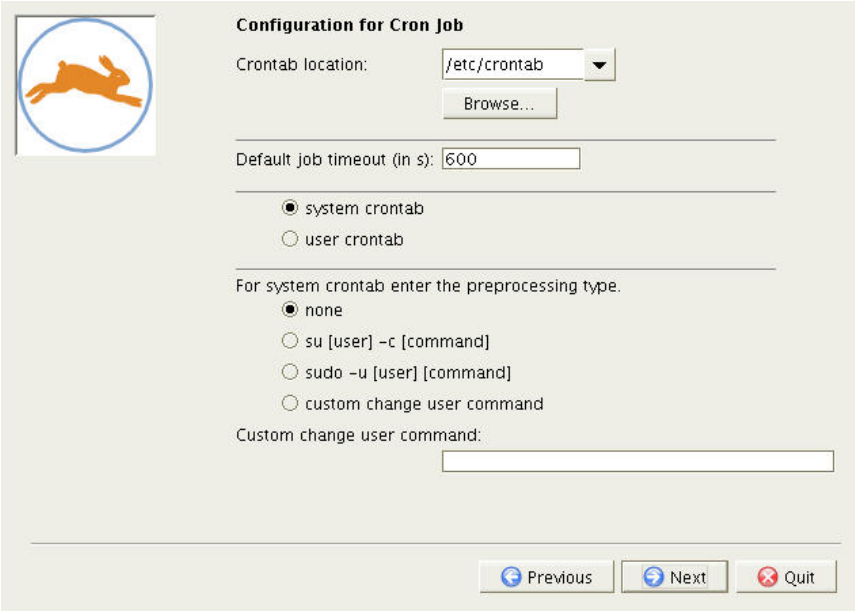
5.1 Install the JobScheduler Crontab Adapter Job

The "Job package: Cron" has to be choosen during the JobScheduler setup for the JobScheduler Crontab Adapter Job to be available:



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

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



The setup will then create the job configuration shown below:

```
<job title          = "Read crontab" >
  <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 repeat  = "120"
            begin   = "00:00"
            end     = "24:00"
            once    = "yes"/>
</job>
```

Example: Configuration of JobScheduler Crontab Adapter Job

The parameters which can be used for a JobScheduler Crontab Adapter Job are described in the chapter ' Parameters ' .

5.2 Using JobScheduler Crontab Adapter Job with remote Configuration

The JobScheduler Crontab Adapter Job may also be used to distribute cron jobs from a supervisor JobScheduler to "Workload" JobSchedulers using remote configuration.

This can be achieved with the following steps:

A JobScheduler must be installed as a central supervisor
The "Workload" JobSchedulers have to be configured to register with the central supervisor (The "Workload" JobSchedulers don't need the cron package).

Directories for the configuration of the workload JobSchedulers are created within the configuration directory of the supervising JobScheduler. (See the " Central Configuration Using a Supervisor JobScheduler " chapter in the JobScheduler 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 crontab files can now be added to the sub-directories of the host-specific configuration directories and the crontab file jobs will be distributed to the corresponding workload JobSchedulers. Note that a special sub-directory (e.g. `cron`) should be created in each host directory (the directory corresponding to a workload JobScheduler) for the configuration files, as files landing directly in the host directory will deleted.

```
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
JobSchedulers
```

Example: crontabs for remote hosts

A JobScheduler on host1 should convert and distribute cron Jobs for host2 and host3. If all JobSchedulers are configured as described above then the workload JobSchedulers will be configured using crontab files saved in the following directories:

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 JobSchedulers. 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 crontab jobs.

Example: crontabs für JobScheduler auf host2 und host3

6 Customizing the Conversion

The conversion result can be influenced using `crontab` comments. Comments can be used to set the job name, the job title and job timeout parameters. These are set using a comment line 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 in the line(s) before the line with the definition of the cron job.

Example: set name of the job to my_cron_job

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

7 User crontab or System crontab?

```
scheduler     localhost = (test) /bin/ls NOPASSWD
```

Example: Scheduler User 'scheduler' should execute 'ls' as user 'test'

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

A `user crontab` file has five columns for the configuration of the run time and an additional column for the command.

A `system crontab` file has five columns for the configuration of the run time, a column for the user who should execute the command and an extra column for the command itself.

It is necessary, for both the JobScheduler Crontab File 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 JobScheduler.

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

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 JobScheduler 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 file`). Note that this only works if the JobScheduler is running as `root`, what is not really recommended. If another user is running the JobScheduler, the `su` command will open a prompt for a passwort, which cannot be answered by the JobScheduler.

If the JobScheduler 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 JobScheduler 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.

8 Parameters

The JobScheduler Crontab File Converter commandline tool and the JobScheduler Crontab Adapter Job are configured using the following parameters:

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 ' ' chapter for an example configuration of this job.

9 Limitations

The five columns (sections) for runtime configuration in a `crontab` file offer a large number of combinations. Most of these can be converted by the JobScheduler Crontab File Converter. However, there are limitations for certain combinations which are listed below:

**/2 */3 * * * : repeat interval for minutes and hours*

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 * * : every 3 days at 10:00*

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 monthdays, which have the configured offset. In this case 1,4,7,10 ...

*0 10 1 */3 * : last DoM every n months*

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: repeat interval for weekdays*

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 : fixed weekday combinded with a fix monthday*

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 *.

10 Further readings

In this chapter we will give an overview for further readings regarding cron and **JobScheduler**.

More about Cron

The Debian Cron page ...

Cron on Cron

More about sudo

The sudo home

sudo on sudo

Index