



Quickstart

Introduction to Job Scheduling
February 2009

Contact Information

Software- und Organisations-Service GmbH
Giesebrechtstr. 15

D-10629 Berlin

Telephone +49 30 864790-0

Telefax +49 30 8613335

Mail info@sos-berlin.com

Web www.sos-berlin.com

Last Updated: December 2008

Table of Contents

1 Introduction	4
1.1 An Overview of the Job Scheduler Documentation	4
2 Creating and Configuring Jobs	5
3 Creating a Job in the "Hot Folder" Using the Job Editor	6
3.1 Pre-requirements for Configuration Using a "Hot Folder"	6
3.2 Carrying Out the Changes	6
4 Creating a Simple Job	12
4.1 Requirements	12
4.2 Procedure for the Configuration of a Job	12
5 Setting Up a Simple Managed Job	14
5.1 The Requirements for Managed Jobs	14
5.2 Procedure for the Configuration of a Managed Job	15
6 Examples for the Use of Job Chains	22
6.1 Starting a Series of Jobs One After the Other with Automatic Error Handling	22
6.2 Starting a Series of Jobs One After the Other with Manual Error Handling	23
6.3 Starting a Series of Jobs One After the Other with Automatic Repetition in the Event of an Error	24
6.4 Starting a Series of Jobs with Manual Repetition in the Event of an Error	25
6.5 Manual Skipping of a Job in a Job Chain	27
6.6 Manually Stopping a Job in a Job Chain	27
6.7 React to Changes in Directories	28
Appendix A: Example 1: Automatic Scheduled Execution of a Shell Script	30
Appendix B: Example 2: Directory Monitoring Based Execution of a Shell Script	31
Appendix C: Example 3: Execution of a PHP Script	32
Appendix D: Example 4: Program Execution	33
Glossary	34

1 Introduction

The Job Scheduler has been installed and can be started. What next?

The following examples should function as a quick introduction to the use of the Job Scheduler. Relative paths are used in this guide, based on the directory in which the Job Scheduler has been installed. Note that this installation directory is also the working directory

(see the chapter on Job Requirements (page 12)). The `[host]` part of the URL is the hostname of the computer on which the Job Scheduler has been installed. This name was specified during the installation of the Job Scheduler.

The Job Scheduler's own Web Server can be started by entering the `[host]` in the URL address field of a standard web browser, followed by the Job Scheduler's TCP port number.

For example, `http://localhost:4444`.

1.1 An Overview of the Job Scheduler Documentation

Installation and Configuration

- **Installation and Configuration (scheduler_installation)**
it is strongly recommended that this document is read *before* the Job Scheduler is installed and configured.
- **Quickstart Introduction to Job Scheduling (scheduler_quickstart)**
This document.
- **Reference Documentation (scheduler)**
A detailed description of job configuration and the Job Scheduler interfaces.

Job Implementation with the Job Scheduler API

These documents are intended for job developers - they are not required for setting up automatic programme and script starts.

- **API Documentation (scheduler_api)**
The Job Scheduler program interface documentation.
- **Job Implementation Tutorial (scheduler_tutorial)**
An introduction to the use of the Job Scheduler program interface.

Advanced Subjects

These documents describe more complex job scheduling scenarios.

- **Tutorial - Implementing Web Services (scheduler_webservices)**
Explains the configuration and implementation of web services for your jobs.
- **Managed Job Scheduling (scheduler_managed_jobs)**
Describes job administration using a database.
- **MySQL Job Scheduling (scheduler_managed_user_jobs)**
Describes the Job Scheduler's SQL statement and procedure interface, which can be used with a database in a similar way to the Oracle Job Scheduler.

2 Creating and Configuring Jobs

There are several ways to create and configure jobs:

- A job can be created and edited using the Job Scheduler's Job Editor, which creates and documents jobs at the same time.
Either an already existing configuration file is opened in the Job Editor, modified and then saved under a new name or a completely new file is created using the Job Editor before being configured.
- Any *text editor* can be used to create and edit a job.
In this case, the configuration file, which is in XML format, is modified "by hand".
- The job can be created and edited using the Managed Jobs interface (page 14).

Further, the configuration of the Job Scheduler itself can be changed in two ways:

- **Static Configuration:**
Here, the `scheduler.xml` file in the `[scheduler install path]/config` directory is modified as required along with any other optional configuration files which have been specified in the `scheduler.xml` file (see the Job Scheduler's Base Configuration).
After changes have been made to this type of configuration, the Job Scheduler must be restarted before the changes made take effect.
After the new start, the `scheduler.xml` file is reread, together with all dependant files.
- **Dynamic Configuration:**
With this configuration method, the configuration is changed by changing the files contained in "hot folder(s)".
The default "hot folder" is the `[scheduler install path]/config/live` directory.
Each file in this directory corresponds with a job, job chain or other element and contains the whole specification information for the element.

3 Creating a Job in the "Hot Folder" Using the Job Editor

The procedure for the configuration of a job for the Job Scheduler using it's Job Editor is described in this section. This procedure is the same for Windows or Unix systems.

As the procedure describes changes made to configuration files stored in the "hot folder", this means that the changes will become effective without the Job Scheduler having to be restarted.

3.1 Pre-requirements for Configuration Using a "Hot Folder"

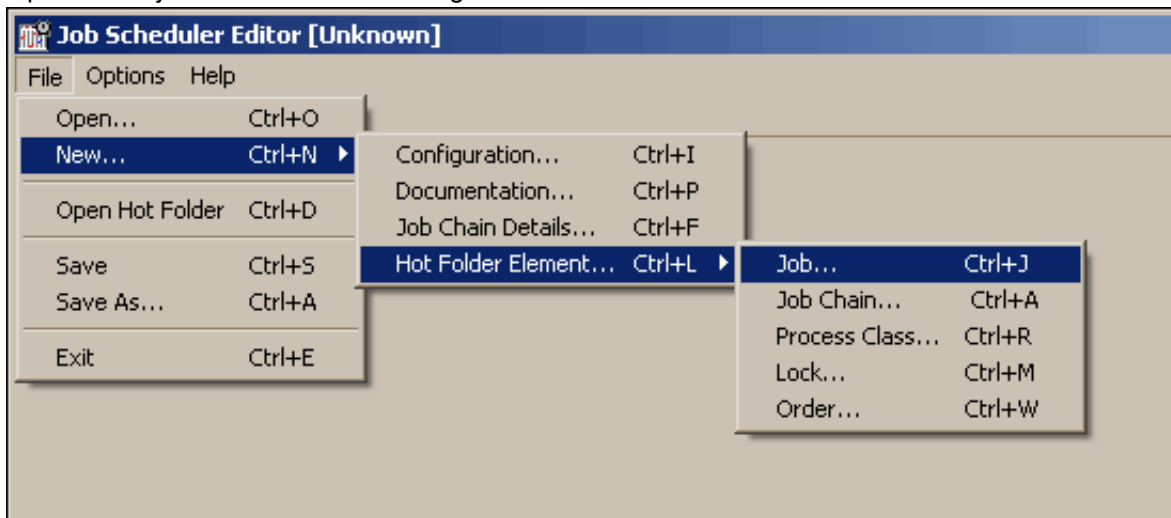
The Job Scheduler needs to be running - which means that the changes will be made "live".

3.2 Carrying Out the Changes

The Job Editor should be started on Windows systems using the `jobeditor.cmd` script and on Unix systems using the `jobeditor.sh` script.

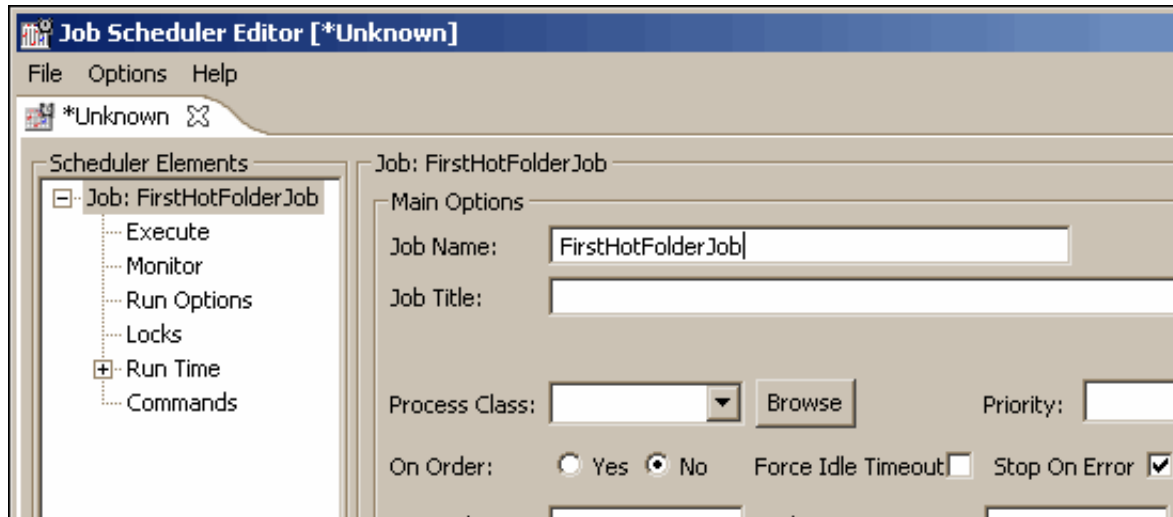
Both of these scripts are to be found in the `[scheduler install path]/bin` directory.

Open a new job in the Job Editor using `File->New...->Hot Folder Element...->Job...`



This creates an empty job configuration element.

Enter a name such as `FirstHotFolderJob` in the "Job Name" field in the main part of the Job Editor window.



Then select the "Execute" item in the Scheduler Elements menu, which is found in the left hand portion of the Job Editor window.

This opens a new configuration form, in which the program that is to be started by the job can be specified.

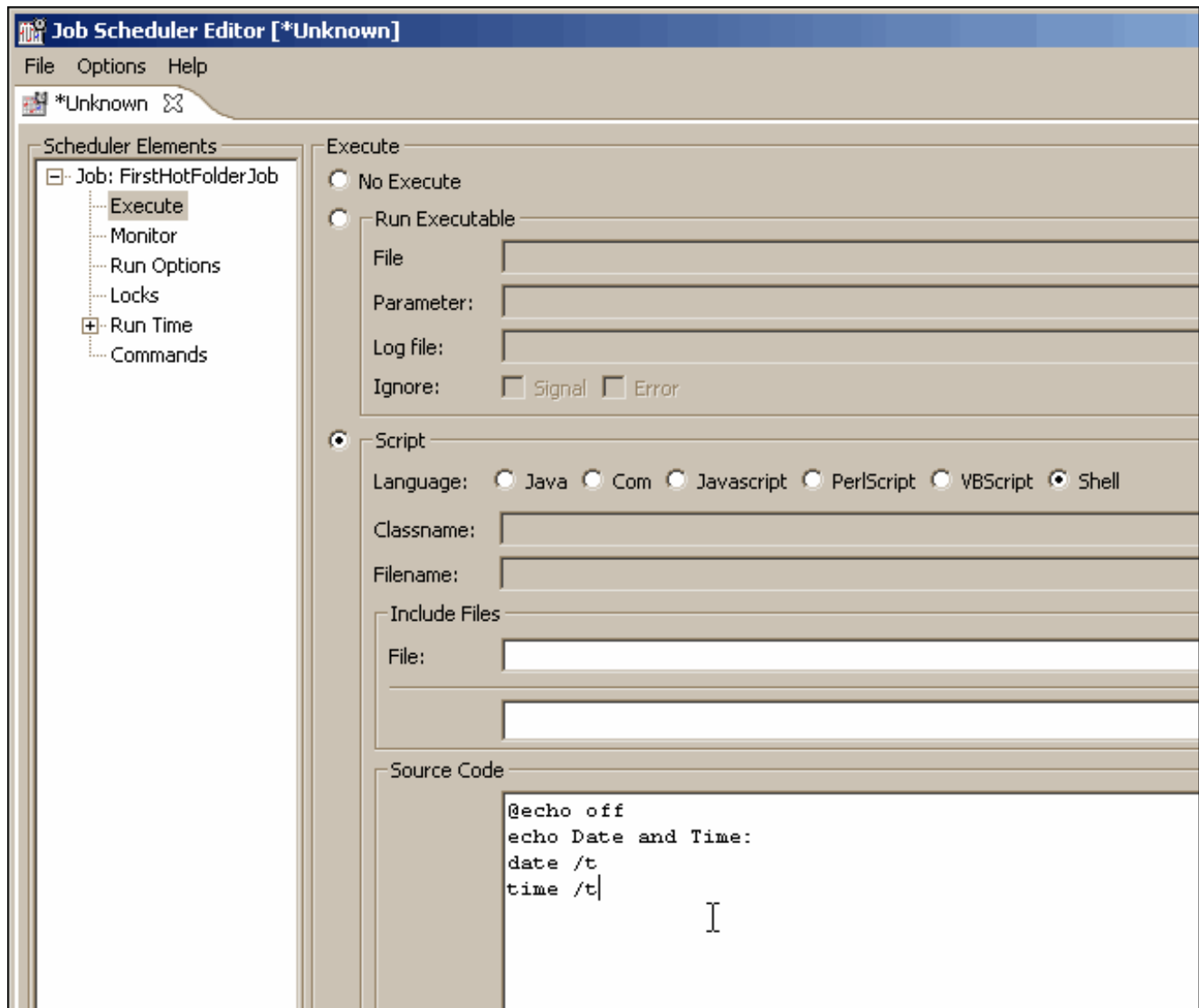
Here, the option of embedding program code directly in the job configuration will be used. To this end, first "Script" is selected and then "Shell" is selected as the language.

If the Job Scheduler is running on a Windows system, then the following should be entered in the "Source Code" field:

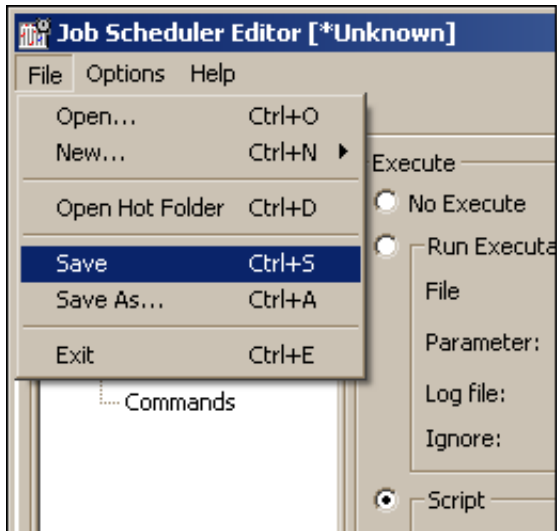
```
@echo off
echo Date and Time
date /t
time /t
```

For a Unix system, the following should be entered in the same field:

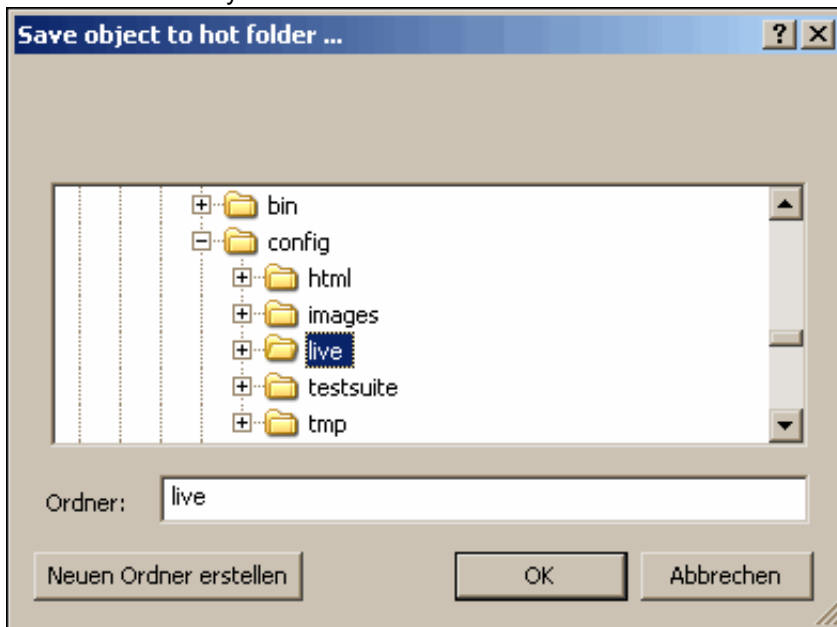
```
#!/bin/sh
echo Date and Time
date
```



This configuration should now be saved, which is done by clicking on **File->Save**. Shown in the screen shot below:



This now causes a dialog to open in which the "hot folder" in which the job is to be saved is specified. The default directory is `[scheduler install path]/config/live`. Select this directory and then click on "OK".



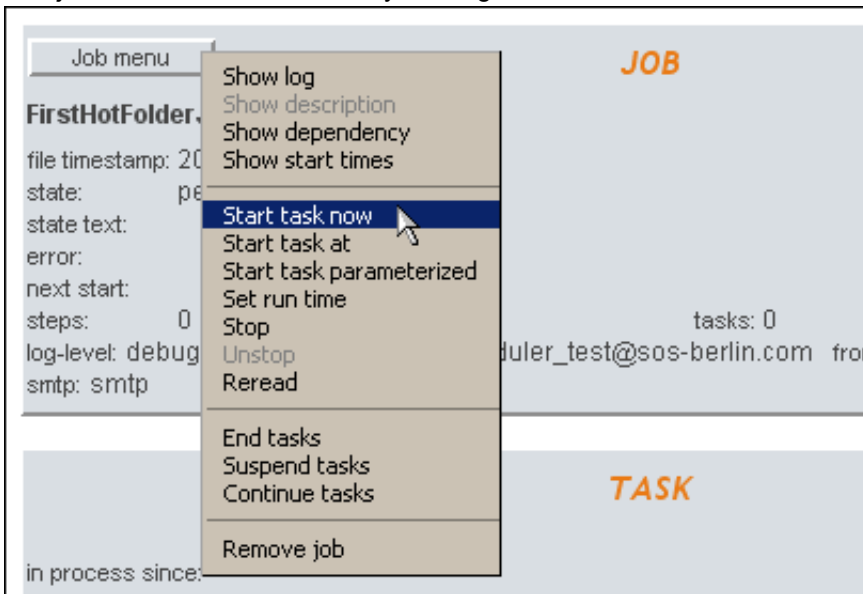
The Job Scheduler web interface should now be opened, in that the URL: `http://[host]:[port]/` is opened in a web browser.

Note that the values for `host` and `port` are as specified in the Job Scheduler installation.

The "FirstHotFolderJob" should now be visible in the Job Scheduler web interface as shown here:

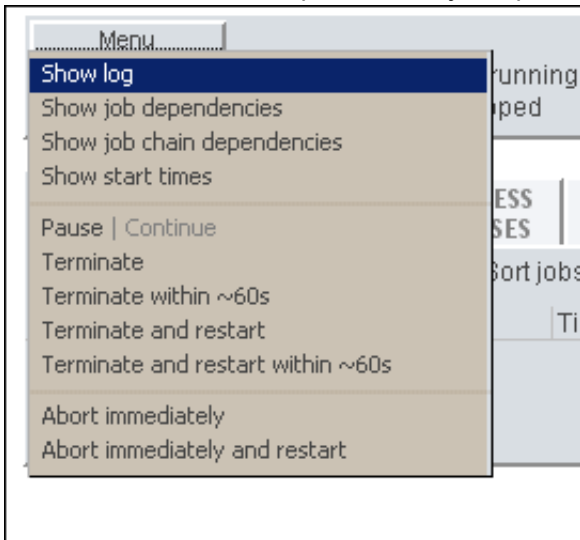


Clicking on the job name in the web interface will open the job menu in the right hand frame of the interface. The job should now be started by clicking on Job menu->Start task now as shown:



The task will now run.

In order to look at the output from the job, open the log file by clicking on Menu->Show log.



This will open a new web browser window containing the Job Scheduler log output, including the output from the "FirstHotFolderJob" task which will look like:

```
[info] (Job FirstHotFolderJob) SCHEDULER-919 Task 2594397 enqueued
[info] (Job FirstHotFolderJob) SCHEDULER-930 Task 2594397 started - cause: queue_at
[info] (Task FirstHotFolderJob:2594397) SCHEDULER-918 state=starting (at=2007-12-03 21:16:51.968)
[info] (Task FirstHotFolderJob:2594397) SCHEDULER-987 Starting process: "C:\WINDOWS\TEMP\sos42C17.cmd"
[info] (Task FirstHotFolderJob:2594397) SCHEDULER-915 Process event
[info] (Task FirstHotFolderJob:2594397) stdout:
[info] (Task FirstHotFolderJob:2594397) Date and Time:
[info] (Task FirstHotFolderJob:2594397) 03.12.2007
[info] (Task FirstHotFolderJob:2594397) 21:16
[info] (Task FirstHotFolderJob:2594397) SCHEDULER-918 state=closed
```

4 Creating a Simple Job

The configuration of an executable file as a job in the Job Scheduler is the subject of this section. Before this example can be followed it is necessary that the Job Scheduler has been successfully installed, either on a Windows or a UNIX system.

New jobs are normally entered in the `[scheduler install path]/config/scheduler.xml` configuration file. After a job has been configured the Job Scheduler must be restarted, in order that the `scheduler.xml` configuration file can be read.

4.1 Requirements

- Successful installation of the Job Scheduler.
- A simple executable file in the form of a Windows cmd or a UNIX sh script is required for the purposes of this example. This file should be saved in the `[scheduler install path]/jobs` directory.

Example scripts:

The following two scripts return on the Windows and UNIX command lines respectively the title "Date and Time:" and in the next two lines the current data and the time respectively.

Windows: Content of the `my_first_job.cmd` cmd Script:

```
@echo off
echo Datums- und Zeitausgabe:
date /t
time /t
```

Unix: Content of the `my_first_job.sh` Shell Script:

```
#!/bin/sh
echo Datums- und Zeitausgabe:
date
```

Note that with the shell script (i.e. on Unix systems), the Job Scheduler user must possess the permissions required to be able to execute the script and that the executable flag is set.

4.2 Procedure for the Configuration of a Job

Open the `[scheduler install path]/config/scheduler.xml` configuration file.

Should the `<jobs>` tag not be present in the file, then the following tags should be added before the closing `</config>` tag:

```
<jobs>
</jobs>
```

The simple script described above is included as a job by inserting the following code (including the `<job>` tags) within the `<jobs>` tags.

Windows Example:

```
<jobs>

  <job name="my_first_job">
    <script language="shell">
      <include file="jobs\my_first_job.cmd"/>
    </script>
    <run_time repeat="10"/>
  </job>

</jobs>
```

In this example the `jobs\my_first_job.cmd` script is included as a Windows job. On a UNIX system the `jobs/my_first_job.sh` would be included.

`<run_time repeat="10"/>` means that the job will be started with the Job Scheduler. After the job has ended, the Job Scheduler will wait 10 seconds before restarting it once more. This procedure will be repeated until the Job Scheduler is stopped.

The `scheduler.xml` file should now be saved in order to retain the changes made and the Job Scheduler restarted. (Restarting the Job Scheduler causes the configuration file to be reread.)

Note that it can take a few seconds for the Job Scheduler to restart, depending on the hardware the Job Scheduler is running on.

After restarting, the Job Scheduler's own web server is opened by entering the following Web Services URL:

`http://[host]:[port]/`

A list of all the installed jobs will then appear in the browser, including the newly created "my_first_job" job.

- "my_first_job" should now be clicked on.
- The browser window will now divide into two areas - a list of jobs and a summary of the selected job.
- The "Task history" tab in the selected job summary should now be clicked on.
- This opens a list showing the start and finishing times of the job runs since starting the Job Scheduler. The difference between a finish time of one job and the start time of the next should be 10 seconds, as set using `<run_time>`.

Congratulations!

The job has been successfully created.

5 Setting Up a Simple Managed Job

The procedure with which an executable file is integrated as a Managed Job in the Job Scheduler for Windows and for UNIX operating systems is described in this section. It is assumed here that the Job Scheduler has been successfully installed and configured for the use of Managed Jobs for the use of Managed Jobs.

New (normal - i.e. unmanaged) jobs are normally entered in the `[scheduler install path]/config/scheduler.xml` configuration file. So that such jobs can be recognised and administrated it is necessary that the Job Scheduler is restarted so that the `scheduler.xml` configuration file can be re-read.

Managed Jobs differ from "normal" jobs in that they are added and administrated using a web browser - the `scheduler.xml` does not need to be modified - and that the information about a job is stored in a database.

5.1 The Requirements for Managed Jobs

- The Job Scheduler has been successfully installed with the following packets:
Database Support ,
Web Interface and
Managed Jobs
 Should the Job Scheduler have been installed without one or more of these packets, then they can be added now. This is described in the Job Scheduler installation and configuration documentation.
- The database can be addressed by the Job Scheduler.
 The Job Scheduler requires a JDBC compatible driver when working with MySQL and SQL Server databases. We are not allowed to include these drivers with the Job Scheduler installation package and therefore they should be downloaded as follows:
 - for MySQL from <http://www.mysql.com>
 - for SQL Server from <http://www.microsoft.com/downloads/details.aspx?FamilyID=e22bc83b-32ff-4474-a44a-22b6ae2c4e17&DisplayLang=en>
- The Job Scheduler web interface can be opened with a browser (i.e. once the Job Scheduler's web server has been correctly configured).
 Example for the Apache Web Server:
 Assuming that the Job Scheduler has been installed in the `my_scheduler/` directory;
 then the following aliases should be added to the Apache Web Server `httpd.conf` file (found in the Apache `conf/` directory):>

```
Alias /my_scheduler/logs/      "C:/my_scheduler/logs/"
Alias /my_scheduler/          "C:/my_scheduler/web/"
```

for Windows - and

```
Alias /my_scheduler/logs/      /home/[ user]/my_scheduler/logs/
Alias /my_scheduler/          /home/[ user]/my_scheduler/web/
```

for Unix systems.

The Apache Web Server should now be restarted. The Job Scheduler web interface can then be reached by opening the following URL in a browser: `http://[hostname]/my_scheduler/index.htm` .

- An executable file such as a command line script for a windows system or a Unix shell script has been prepared.

This file should then be saved in the `[scheduler install path]/jobs/` directory.

Example scripts:

The following two scripts return on the Windows and UNIX command lines respectively the title "Date and Time:" and in the next two lines the current data and time.

Windows: Content of the `my_first_job.cmd` cmd Script:

```
@echo off
echo Datums- und Zeitausgabe:
date /t
time /t
```

Unix: Content of the `my_first_job.sh` Shell Script:

```
#!/bin/sh
echo Datums- und Zeitausgabe:
date
```

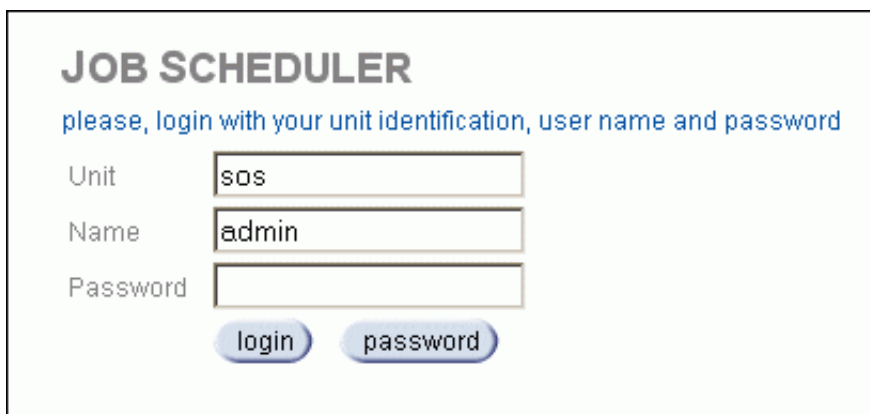
Note that with the shell script (i.e. on Unix systems), the Job Scheduler user must possess the permissions required to be able to execute the script and that the executable flag is set.

5.2 Procedure for the Configuration of a Managed Job

The Job Scheduler web interface should be opened in a browser using the `http://[hostname]/my_scheduler/index.htm` URL.

A log-in page appears.

The following should be entered: `sos` for Unit; `admin` for Name and the password field should be left empty. The "Login" button should now be clicked.



JOB SCHEDULER

please, login with your unit identification, user name and password

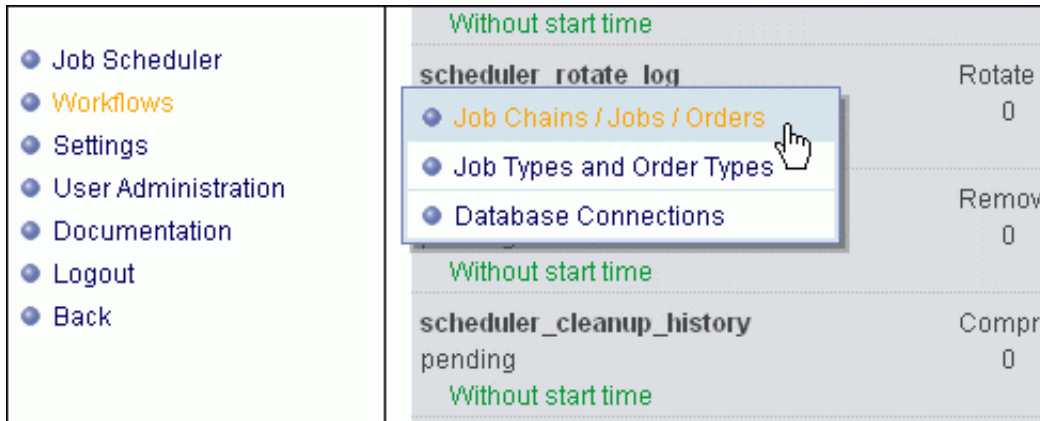
Unit:

Name:

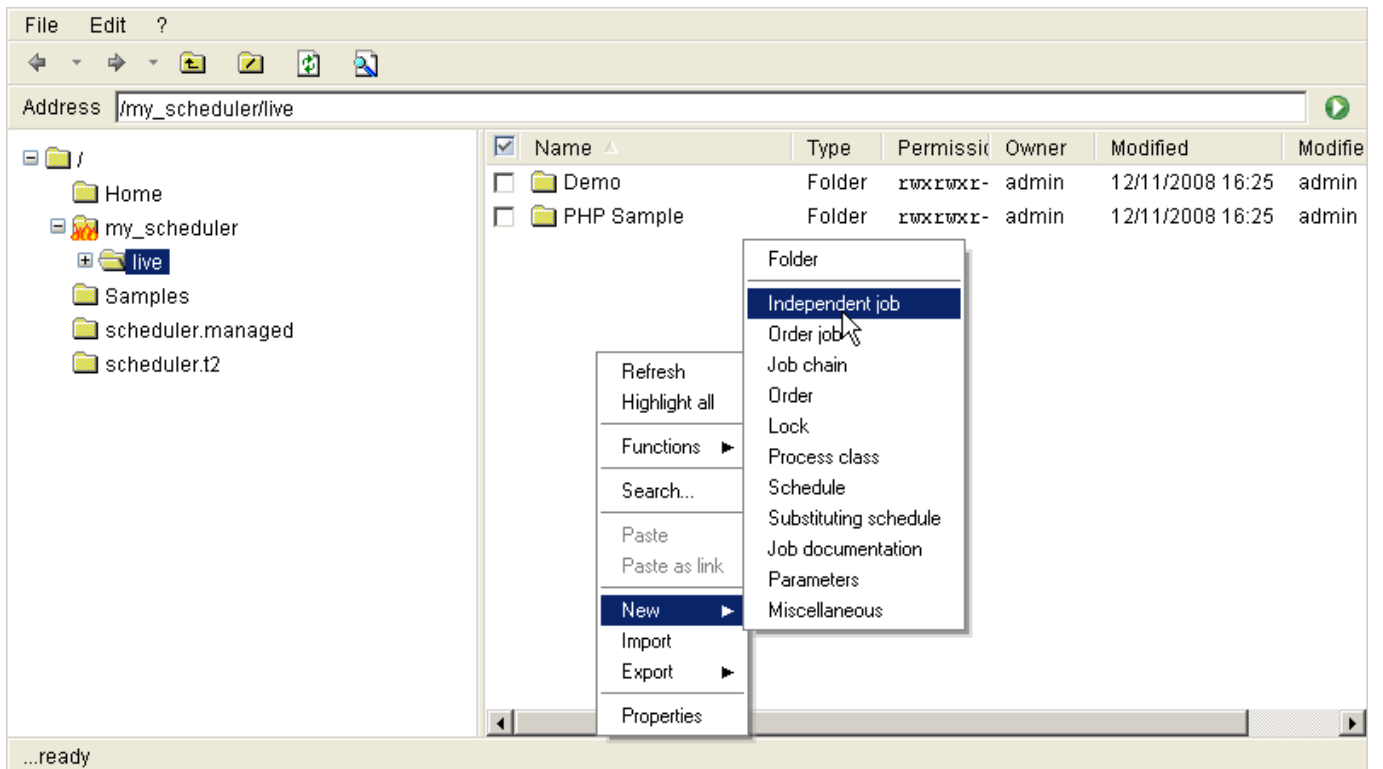
Password:

The main Managed Jobs administration page appears after successful registration. This page has a selection menu to the left and a list of different jobs on the right hand side.

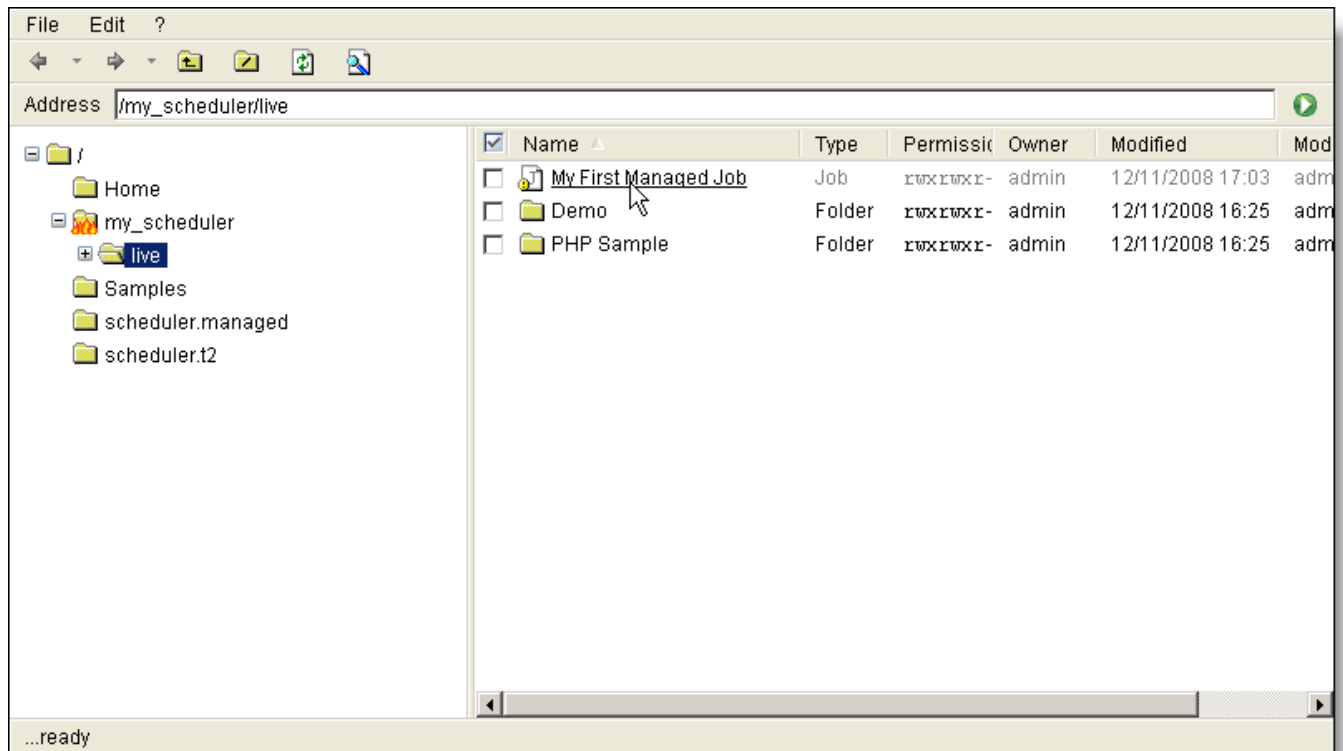
"Workflows"-> "Job-Chains / Jobs / Orders" should be selected from the menu on the left.



This opens Job Scheduler Explorer in the right side of the browser window. Switch to the `live` directory of your *Job Scheduler*. A new order-independent job should now be created, in that in the right hand part of the Explorer, "New"->"Independent Job" is selected in the context menu.



This job should now be given a name such as "My First Managed Job".



The Job Editor is now started, in that the newly created job is clicked once.

The Editor form is then completed as follows:

- enter a title for the job and
- select the "Executable File" job type from the Job configuration area and then click on the symbol to the right of the Job configuration area:



Independent job : My First Managed Job

XML

Job configuration

job type: Executable PHP File connection: -

accept this job configuration

job

- run_time

Title: Managed Jobs Demo

Process Class: [empty]

Tasks: [empty] numeric

Min Tasks: [empty] numeric

Timeout: [empty] seconds | hh:mm:ss

Priority: [dropdown]

Temporary: [dropdown]

Java Options: [empty]

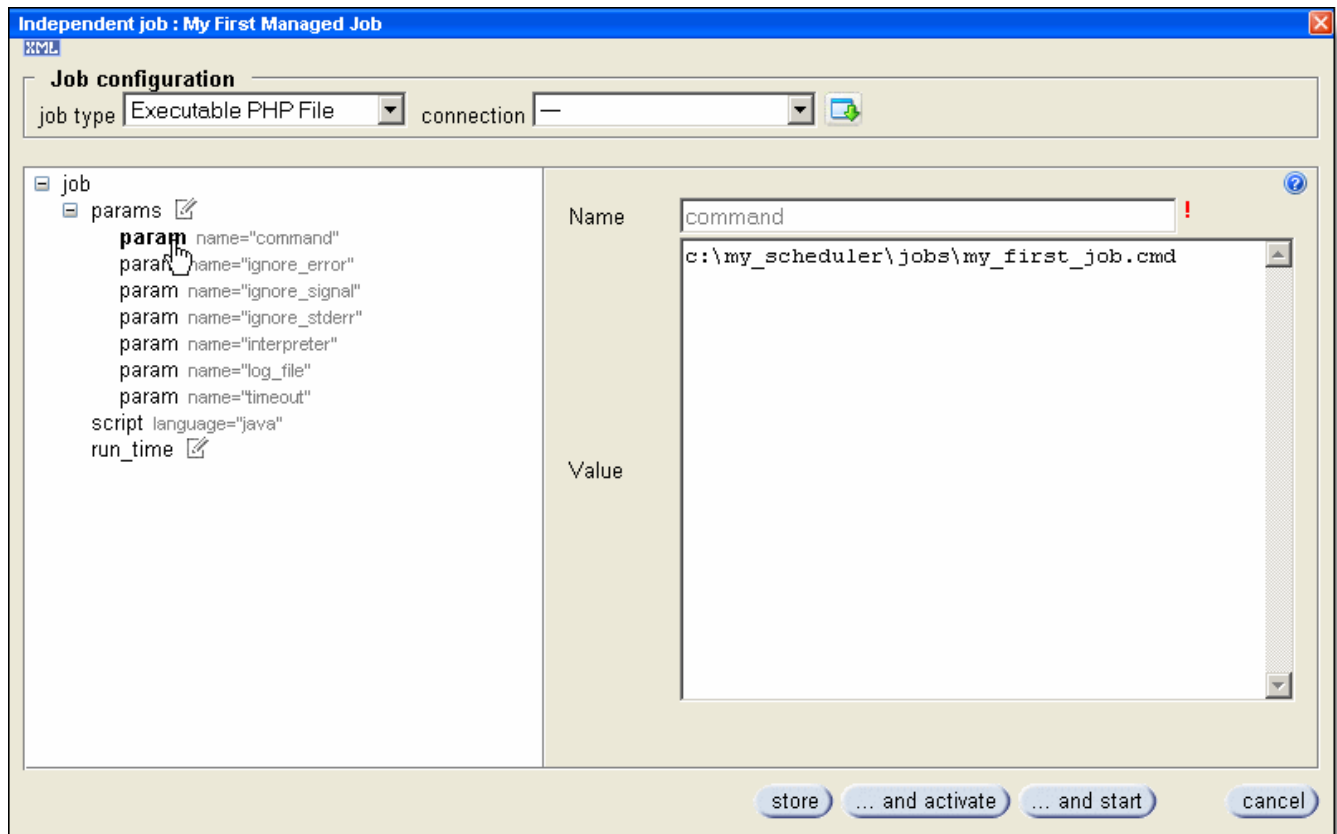
Visible: [dropdown]

Ignore Signals: [empty]

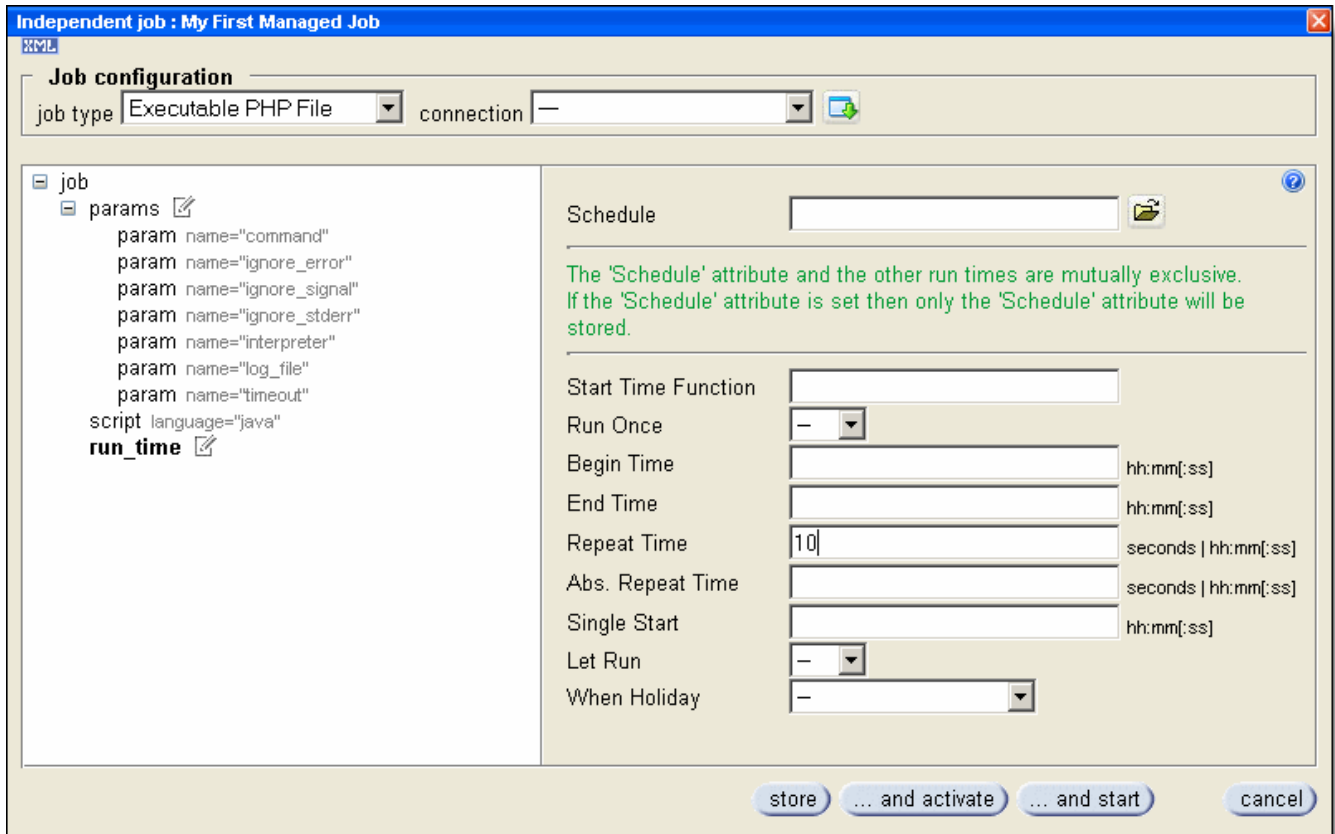
Stop On Error: [dropdown]

store ... and activate ... and start cancel

The "Executable File" parameter mask now opens for the job.
The path of the script to be executed is now entered as a value for the `command` parameter.

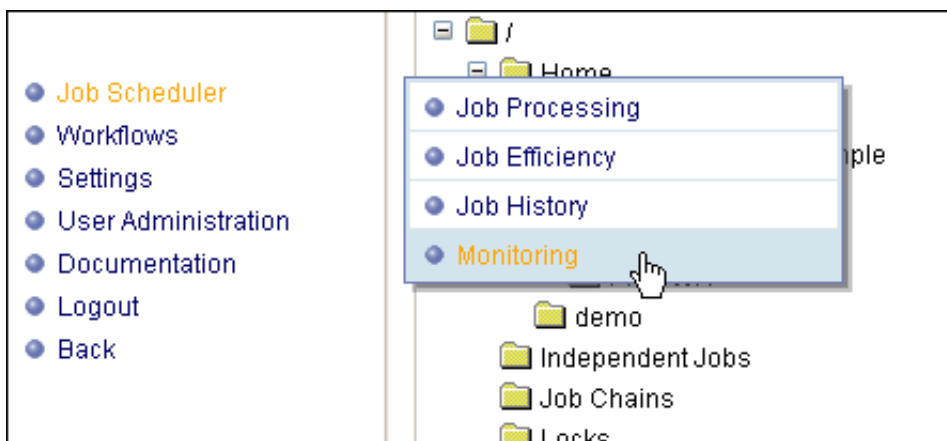


The job run-time(s) are now set in the `run_time` element.
10 seconds is entered for the `Repeat Time` attribute.
The "Store and activate" button is now clicked.



As a Repeat Time interval has been set for the job run-time, the Job Scheduler starts immediately after the job has been confirmed. The job will be repeatedly restarted, each restart taking place 10 seconds after the job was last completed.

"Job Scheduler"-> "Monitoring" should now be selected.



This opens a list of jobs in which "my_first_managed_job" can be found. This link should now be clicked on.

The right hand portion of the browser window now shows a list of all jobs and summary information about the selected job.

The "Task history" tab should now be selected.

This opens a list showing the start and finishing times of the job runs since starting the Job Scheduler.

The difference between a finish time of one job and the start time of the next should be 10 seconds, as set using `<run_time>`.

Congratulations!
Your first Managed Job has been successfully configured.

6 Examples for the Use of Job Chains

6.1 Starting a Series of Jobs One After the Other with Automatic Error Handling

- **Properties**
 - A job will only be started after its predecessor has been successfully completed.
 - In the event of an error, the `stop_on_error="no"` job attribute will cause the job chain to be set to the error state and the order in which it is running to be ended.
 - Neither the continued processing of the order nor its restarting is allowed.
 - Behavior on restarting the Job Scheduler: none, the order is entered in the Job Scheduler history.
- **Error Handling**
 - automatic: the order is stopped.
 - Starts its own error handling job.
 - An e-mail containing notification of the error is automatically sent out.
- **Field of Application**

This type of job configuration is used when errors in individual orders are to be automatically handled and other orders running on the same job chain are not to be stopped. This makes sense when it is most probable that errors occur in individual orders and do not automatically affect other orders. It is assumed that orders can be repeatedly created, either manually or by an external application. Generally, the orders used in this sort of configuration will not have start times specified.
- **Example**

```
<job name="job1" order="yes" stop_on_error="no">
  ...
</job>

<job name="job2" order="yes" stop_on_error="no">
  ...
</job>

<job_chain name="Chain_1">
  <job_chain_node state="1"
    job="job1"
    next_state="2"
    error_state="err"/>

  <job_chain_node state="2"
    job="job2"
    next_state="end"
    error_state="err"/>

  <job_chain_node state="end"/>
  <job_chain_node state="err"/>
</job_chain>
```

Use of the `Chain_1` job chain ensures that the `job2` job is only started after `job1` has been completed.

Should an error occur in either `job1` or `job2`, then an `err` error state will be set.

6.2 Starting a Series of Jobs One After the Other with Manual Error Handling

- **Properties**

- A job will only be started after its predecessor has been successfully completed.

- In the event of an error, the `stop_on_error="yes"` job attribute will cause the order to be added to the order queue, positioned ready to restart the job in which the error occurred.

- All other orders containing the job in question will also be stopped and added to the order queue once they reach the job in which the error occurred.

- This means that the orders will only run to the start of the job in which the occurred and that the job chain is stopped at this point.

- This interruption in the job chain by a stopped job can only be revoked manually.

- Behavior on restarting the Job Scheduler:

- when a database is being used and the `orders_recoverable="yes"` job chain attribute is set (the default setting), then, as the order queue is stored in the database, orders can be read from the order queue and be continued.

- should the job causing the error remain unchanged after the Job Scheduler is restarted, then the first order will cause an error again and the job causing the error will be stopped.

- should a database not be in use or the `orders_recoverable="no"` job chain attribute have been set, then new orders can be run on the job chain. However, the old orders waiting in the order queue will be lost after the Job Scheduler is restarted.

- **Error Handling**

- the job will be automatically stopped

- an e-mail containing notification of the error is automatically sent.

- manual intervention by the administrator is required:

- After the administrator has corrected the source of the error, the job can be restarted using "Unstop". All waiting orders, including the order in which the original error occurred, will then proceed normally in the job chain.

- **Field of Application**

- This type on job configuration is used when it is most likely that the causes of errors lie in jobs and not in the individual orders.

- That is, it is most likely that when a job error occurs in one order, then it will also occur in all subsequent orders.

- Use of a database ensures that orders are not lost if the Job Scheduler is restarted.

- This type of configuration is used when orders cannot be re-created.

- Generally, the orders used in this sort of configuration will not have start times specified.

- **Example**

```
<job name="job1" order="yes" stop_on_error="yes">
```

```
...
```

```
</job>
```

```
<job name="job2" order="yes" stop_on_error="yes">
```

```
...
```

```
</job>
```

```
<job_chain name="Chain_2" orders_recoverable="yes">
```

```
<job_chain_node state="1"
```

```

        job="job1"
        next_state="2"/>

<job_chain_node state="2"
    job="job2"
    next_state="end"/>

<job_chain_node state="end"/>
</job_chain>

```

Use of the `Chain_2` job chain ensures that the `job2` job is only started after `job1` has been completed. Should an error occur in either `job1` or `job2`, then the job in which the error occurs will be stopped. When `stop_on_error="yes"` is included in a job configuration, then definition of `error_state` is superfluous, as the order will not ever reach an error state. The Job Scheduler will repeatedly attempt to run order in a job until it has successfully completed the job. After this happens, the order is given the positive `next_state` successor state.

6.3 Starting a Series of Jobs One After the Other with Automatic Repetition in the Event of an Error

- **Properties**
 - A job will only be started after its predecessor has been successfully completed.
 - In the event of an error, the `stop_on_error="no"` job attribute and `on_error="setback"` job chain attribute ensure that the order is stopped and after a defined period restarted at the job in which the original error occurred.
 - Should the error no longer occur then the order will proceed along the job chain as normal.
 - Should, however, the error still occur after a predefined number of repetitions of the job, then the job chain will be set to the error state for the order in question.
 - Other orders will not be affected by the occurrence of an error in any one order.
 - Behavior on restarting the Job Scheduler: none, the order is entered in the Job Scheduler history.
- **Error Handling**
 - an e-mail containing notification of the error is automatically sent.
 - automatic: the order will be repeated n-times
 - starts its own error handling job.
- **Field of Application**

This type of job configuration is used when errors in individual orders are to be automatically handled and other orders running on the same job chain are not to be stopped. This makes sense when it is most probable that errors occur in individual orders and do not affect automatically extend to other orders. Here it is assumed that after a reasonable number of repetitions there is a good chance that the order can run the job successfully.

The errors most likely expected are such that they will resolve themselves, when enough time has passed. Orders can be repeatedly created, either manually or by an external application, should the error(s) occurring not be solved sufficiently quickly.

Generally, the orders used in this sort of configuration will not have start times specified.
- **Example**

```

<job name="job1" order="yes" stop_on_error="no">
  ...
  <delay_order_after_setback
    setback_count="2"

```

```

    delay="5"/>

<delay_order_after_setback
  setback_count="4"
  delay="10"/>

<delay_order_after_setback
  setback_count="5"
  is_maximum="yes"/>
</job>

<job name="job2" order="yes" stop_on_error="no">
  ...
</job>

<job_chain name="Chain_3">
  <job_chain_node state="1"
    job="job1"
    next_state="2"
    error_state="err"
    on_error="setback"/>

  <job_chain_node state="2"
    job="job2"
    next_state="end"
    error_state="err"/>

  <job_chain_node state="end"/>
  <job_chain_node state="err"/>
</job_chain>

```

Should an error occur in `job1` for the first time, then the order will be stopped and the job rerun immediately. Should the error occur once more (2nd time), then the `setback_count="2"` will cause the order to be delayed 5 seconds before being rerun.

The order will also be delayed for 5 seconds should the error occur a third time.

After the 4th occurrence of the error, the job will be delayed 10 seconds.

The order will also be delayed for 10 seconds should the error occur a fifth time.

The occurrence of a further (6th) error will cause the order to be set to the `err` state.

Should an error occur in the `job2` job, then an `err` error state will be directly set.

Note that `<delay_order_after_setback>` must be set in the job when `on_error="setback"` is set in the relevant order.

6.4 Starting a Series of Jobs with Manual Repetition in the Event of an Error

- **Properties**

- A job will only be started after its predecessor has been successfully completed.

- In the event of an error, the `stop_on_error="no"` job attribute and `on_error="suspend"` job chain attribute will cause the order to be delayed. The order will wait for the job in which the error occurred to be successfully completed outwith the order queue.

- The order can only be manually reactivated.

- Should the error no longer occur then the order will proceed along the job chain as normal.

- Other orders will not be affected by the occurrence of an error in any one order.
- Behavior on restarting the Job Scheduler:

When a database is being used and the `orders_recoverable="yes"` job chain attribute is set (the default setting), then suspended orders can be read from the database and wait to be reactivated.

Should a database not be in use or the `orders_recoverable="no"` job chain attribute have been set, then the original suspended orders will be lost on restarting the Job Scheduler.

- **Error Handling**

- an e-mail containing notification of the error is automatically sent.
- the order will be stopped at the job in which the error occurred
- manual intervention by the administrator is required:

The order can be restarted by the administrator after the reason for the error has been solved.

- **Field of Application**

It is assumed here that other orders will be able to successfully run through the job chain, even though a error has occurred on one order.

Should an error occur, then it is not expected that repetition of the order will lead to it being successfully being completed.

This configuration type is used when orders cannot be re-created and when it is not possible to continue processing an order after an error has occurred in one job.

Generally, the orders used in this sort of configuration will not have start times specified.

- **Example**

```
<job name="job1" order="yes" stop_on_error="no">
  ...
</job>

<job name="job2" order="yes" stop_on_error="no">
  ...
</job>

<job_chain name="Chain_4" orders_recoverable="yes">
  <job_chain_node state="1"
    job="job1"
    next_state="2"
    on_error="suspend"/>

  <job_chain_node state="2"
    job="job2"
    next_state="end"
    on_error="suspend"/>

  <job_chain_node state="end"/>
</job_chain>
```

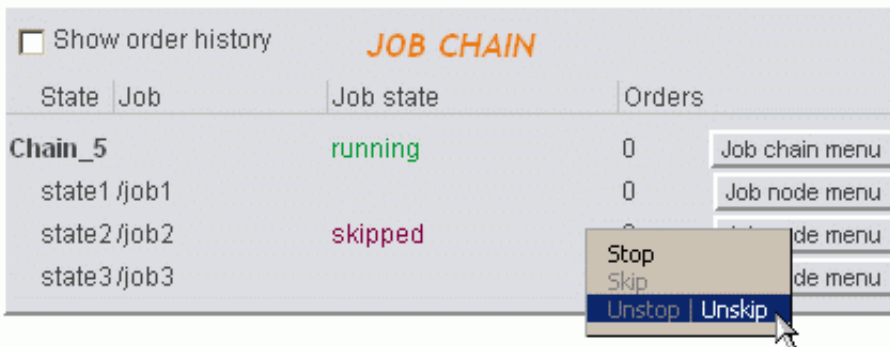
Should an error occur in either `job1` or `job2`, then the order will stop before the job in question and not be placed in the order queue.

Orders cannot be set to the `error_state` successor state.

6.5 Manual Skipping of a Job in a Job Chain

The administrator can manually configure a job chain node so that can be bypassed ('skipped') during order processing in the event of an error. This means that an order being processed along a job chain would miss out one node and proceed to the next job chain node and its corresponding job.

"Unskip" can be used to reactivate the node in a job chain.



6.6 Manually Stopping a Job in a Job Chain

The administrator can manually stop a job at any time using the "Stop" command. In this case, orders waiting to carry out this job are held in the order queue until the job is restarted. After the job in question has been restarted, the queuing orders then proceed along the job chain as normal. This procedure neither causes an error nor an error e-mail to be sent. The "Unstop" command is used to start the job once more.



JOB CHAIN			
State	Job	Job state	Orders
Chain_5		running	0
	state1 /job1		0
	state2 /job2	stopped	
	state3 /job3		

6.7 React to Changes in Directories

- **Task**

To create a unit-capable system which reacts to the arrival of files. The unit corresponding to the file can be identified from the file. Each file corresponds to an order for a unit. The same program procedure should be repeated for different units. Should problems occur with an order, the system should continue to process other orders.

- **Solution**

Directory monitoring combined with file orders offers a way of achieving the required system. Any directory can be used as the directory to be monitored. When a file is copied into this directory, the Job Scheduler's directory monitoring is activated and a order is created for the file.

- **Example**

```
<job name="job_1" order="yes" stop_on_error="no">
  ...
</job>

<job name="job_2" order="yes" stop_on_error="no">
  ...
</job>

<job name="job_3" order="yes" stop_on_error="no">
  ...
</job>

<job_chain name="Chain_A">

  <file_order_source directory="path"/>

  <job_chain_node state="state_1" job="job_1" error_state="error"/>
  <job_chain_node state="state_2" job="job_2" error_state="error"/>
  <job_chain_node state="state_3" job="job_3" error_state="error"/>

  <file_order_sink state="ok" remove="yes"/>
  <file_order_sink state="error" move_to="errorpath"/>

</job_chain>
```

When a file is copied into the `path` directory, then the directory monitoring is activated and an order is created.

Each order is processed by `job_1` then `job_2` before finally being processed by the `job_3` job.

Should an error occur during the processing of an order, then the order is removed from the order queue and its corresponding file is added to the `errorpath` directory. If the order successfully completes all three jobs then the file used which initiated the order is deleted.

Whether or not an error occurs when an order is being processed, the job chain remains open for further order files.

- **Note:**

It is also possible to configure directory monitoring for a stand-alone job.

See `<start_when_directory_changed>`

In comparison with the directory monitoring using file orders described in the previous section, in this solution, a similar order is created for each file arriving in the directory.

With the stand-alone directory monitoring job described here, only the directory as a whole is monitored - i.e. the monitoring job reacts to every new or deleted file. Further, the job itself must determine the type of change which has taken place in the directory (file added, deleted, renamed or re-saved).

In other words, the monitoring job only reacts to the number of files in the directory.

However, this simple monitoring solution provides a good service considering the small amount of code required.

Appendix A: Example 1: Automatic Scheduled Execution of a Shell Script

This example job for the `my_shell_script` script is configured as follows:

- for an automatic start every 30 minutes (the job will be run 1800 seconds after the previous run has been completed), weekdays between 9 and 12 o'clock
- and for manual starts between 8:00 & 20:00 every day.
- The job starts the `./jobs/my_shell_script.sh` shell script.
- Note that the scheduling can be combined with directory monitoring. This is described in Example 2 below.

The job is added to the `./config/scheduler.xml` configuration file in the form of a job element as follows:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<spooler>
  <config ...>
    <security ignore_unknown_hosts = "yes">
      <allowed_host host = "localhost" level = "all"/>
    </security>
    <process_classes>
      ...
    </process_classes>
    <jobs>
      <job name = "my_shell_script">
        <process file = "jobs/my_shell_script.sh"
          <!-- in Windows analog "jobs\my_shell_script.cmd" -->
            param = "">
        </process>
        <run_time begin = "08:00"
          end = "20:00">
          <weekdays>
            <day day="1">
              <period begin = "09:00" end = "12:00" repeat = "1800"/>
            </day>
            <day day="2">
              <period begin = "09:00" end = "12:00" repeat = "1800"/>
            </day>
            <day day="3">
              <period begin = "09:00" end = "12:00" repeat = "1800"/>
            </day>
            <day day="4">
              <period begin = "09:00" end = "12:00" repeat = "1800"/>
            </day>
            <day day="5">
              <period begin = "09:00" end = "12:00" repeat = "1800"/>
            </day>
          </weekdays>
        </run_time>
      </job>
    </jobs>
  </config>
</spooler>
```

Appendix B: Example 2: Directory Monitoring Based Execution of a Shell Script

In this example the `./notification_dir` directory is monitored.

- This job is configured so that the Job Scheduler starts the `my_shell_script` job each time a ".txt" file is either added to or deleted from the monitored directory. In addition, the job can be started manually.
- The job starts the `./jobs/my_shell_script.sh` shell script.
- Directory monitoring can also be combined with time-based starts (see Example 1), above.

The job element containing the job configuration information is added to the `./config/scheduler.xml` file as follows:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<spooler>
  <config ...>
    <security ignore_unknown_hosts = "yes">
      <allowed_host host = "localhost" level = "all"/>
    </security>
    <process_classes>
      ...
    </process_classes>
    <jobs>
      <job name = "my_shell_script">
        <process file = "jobs/my_shell_script.sh"
          <!-- in Windows analog "jobs\my_shell_script.cmd" -->
          param = "">
        </process>
        <start_when_directory_changed directory = "notification_dir"
          regex = "\.txt$"/>
        <run_time/>
      </job>
    </jobs>
  </config>
</spooler>
```

Appendix C: Example 3: Execution of a PHP Script

In this example the `my_php_script` script is allowed be started manually.

- The job executes the PHP command line interface, which allows the PHP-Skript to be handed over as a parameter.
- Note that should it be necessary to hand over parameters with the PHP script, then they should be listed after the script, seperated by spaces.
- See Example 1 and Example 2 for information about the automatic execution of jobs and directory monitoring.

The job is added to the `./config/scheduler.xml` configuration file in the form of a job element as follows:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<spooler>
  <config ...>
    <security ignore_unknown_hosts = "yes">
      <allowed_host host = "localhost" level = "all"/>
    </security>
    <process_classes>
      ...
    </process_classes>
    <jobs>
      <job name = "my_php_script">
        <process file = "/usr/src/php-4.3.3/sapi/cli/php"
          <!-- in Windows meist "c:\php\cli\php.exe" -->
          param = "-f jobs/my_php_script.php">
        </process>
        <run_time/>
      </job>
    </jobs>
  </config>
</spooler>
```

Appendix D: Example 4: Program Execution

In this example a MySQL database Windows Service will be restarted at 4:00 am by way of two jobs.

- The first `service_stop` job stops and the second `service_start` job starts the Windows service. The Windows own `net.exe` program is used for both jobs. This program lies in a sub-directory of the Windows `system32` directory. The Windows `%windir%` environment variable can be used for this path. Note that environment variables can also be designated on Windows systems using "\$".
- The MySQL database service name is "MySQL". Additionally, jobs cannot be manually started.
- Note also that in this example it would actually be better to use a shell script, in order to catch errors.

The job is added to the `./config/scheduler.xml` configuration file in the form of a job element as follows:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<spooler>
  <config ...>
    <security ignore_unknown_hosts = "yes">
      <allowed_host host = "localhost" level = "all"/>
    </security>
    <process_classes>
      ...
    </process_classes>
    <jobs>
      <job name = "service_stop">
        <process file = "$windir\system32\net.exe"
          param = "stop MySQL">
        </process>
        <run_time>
          <period single_start = "04:00"/>
        </run_time>
      </job>
      <job name = "service_start">
        <process file = "$windir\system32\net.exe"
          param = "start MySQL">
        </process>
        <run_time>
          <period single_start = "04:01"/>
        </run_time>
      </job>
    </jobs>
  </config>
</spooler>
```

Glossary

Job

A processing step of the Job Scheduler.

Programs and scripts which are to be executed by the Job Scheduler must be embedded in jobs. Jobs can either start (or stop) executable files or contain job scripts which use the Job Scheduler program interface. Jobs can be run in more than one process or task, when more throughput is required.

Job Chain

A series of jobs which are processed one after the other.

The Job Scheduler starts the jobs in a job chain automatically after an order is received. Depending on the payload, the jobs in a job chain can be run in parallel in order to carry out a larger number of orders.

Managed Jobs

Jobs which are administrated using a database and which are automatically allocated to one or more Job Schedulers. Note that one of the prerequisites for the use of Managed Jobs is the use of a database.

Order

An order activates the processing of a job chain and contains parameters for one or more jobs in a job chain. A persistent order contains a start time or a start interval for repeated starts. Should an order not be persistent, then it is deleted after it has been processed in the job chain.

An order proceeds along the jobs in a job chain one after the other. Should a processing error occur, then the order is stopped and removed from the job chain. Every job in a job chain has access to the parameters of an order.