Job Scheduling

JOBSCHEDULER

Technical Description
April 2016

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 www.sos-berlin.com

Last Updated: April 2016

Table of Contents

1 Configuring the JobScheduler

1.1 XML Configuration

The file containing the XML configuration should be specified when starting the JobScheduler.

1.1.1 Example

The following shows a simple configuration file with a shell job:

```
<?xml version="1.0">
<spooler>
    <config>
        <jobs>
            <job name="hello_world">
                <script language="shell"><![CDATA[
                    echo hello world
                ]]></script>

                <run_time repeat="10"/>
            </job>
        </jobs>
    </config>
</spooler>
```

This example shows the configuration of the `hello_world` job, which the JobScheduler repeats every 10 seconds.

When this configuration is saved in the `hello_world.xml` file, the JobScheduler could be started as follows (for Windows):

`…`*`scheduler installation path`*`…\bin\scheduler.exe -config=hello_world.xml`

Note that if the JobScheduler has been started at the command line, it can be stopped using the "Ctrl-C" keys.

Alternatively you could create individual configuration files per job, job chain etc. that would contain exclusively the elements required by these objects:

```
<?xml version="1.0">
<job>
    <script language="shell"><![CDATA[
        echo hello world
    ]]></script>

    <run_time repeat="10"/>
</job>
```

When this configuration is saved in a file *`scheduler installation path`*`/config/live/hello_world.job.xml`, then it would be automatically used by a running JobScheduler.

1.1.2 Coding the XML Configuration

`<?xml encoding="UTF-8"?>` and
`<?xml encoding="ISO-8859-1"?>` are allowed.

Note that the JobScheduler only processes 8 bit Characters (ISO-8859-1).

1.1.3 Configuration Schema

The JobScheduler verifies the XML configuration according to the JobScheduler XML Schema (Is only shown correctly in IE).

Clicking on an XML element in the schema list below leads to a description of the element:

```
<spooler
    <config central_configuration_directory = "…"
            configuration_add_event    = "…"
            configuration_delete_event = "…"
            configuration_directory    = "…"
            configuration_update_event = "…"
            include_path               = "…"
            ip_address                 = "…"
            log_dir                    = "…"
            mail_xslt_stylesheet       = "…"
            param                      = "…"
            port                       = "4444"
            priority_max               = "1000"
            spooler_id                 = ""
            supervisor                 = "…"
            tcp_port                   = "4444"
            udp_port                   = "4444">
            time_zone                  = "...">

        <base file="…"/>

        <params>
            <param name="…" value="…"/>
        </params>

        <security>
            <allowed_host name="…" level="…"/>
            …
        </security>

        <plugins>
            <plugin java_class="…"/>
            …
        </plugins>

        <cluster heart_beat_own_timeout  = "…"
                 heart_beat_timeout      = "…"
                 heart_beat_warn_timeout = "…"/>

        <process_classes ignore  = "no">
            <process_class max_processes = "…"
                           name          = "…"
                           spooler_id    = "…">
                <remote_schedulers ignore  = "no">
                    <remote_scheduler
                        remote_scheduler       = "…"
                        http_heartbeat_period  = "…"
                        http_heartbeat_timeout  = "…">
                    </remote_scheduler>
                </remote_schedulers>
            </process_class>
            …
        </process_classes>

        <schedules>
            <schedule name = "…"
                substitute = "…"
                valid_from = "…"
                valid_to   = "…"/>
            …
        </schedules>

        <locks>
            <lock name= "…" …/>
            …
        </locks>

        <script com_class   = "…"
                filename    = "…"
                java_class  = "…"
                language    = "…" >
            <include file="…"/>
            <![CDATA[
                program-code…
            ]]>
        </script>

        <http_server>
            <http.authentication scheme="basic">
                <http.users>
                    <http.user name="…" password_md5="…"/>
                    …
                </http.users>
            </http.authentication>

            <http_directory path="…" url_path="…"/>
            <web_service job_chain="…" url_path="…" …>
            <params>
                     <param name="…" value="…"/>
                     …
                 </params>
            </web_service>
            …
        </http_server>

        <holidays>
```

```
        <holiday date="…"/>
        <include file="…"/>
        <weekdays date="…"/>
    </holidays>

    <jobs>
        <job
            force_idle_timeout  = "yes| no"
            idle_timeout    = "…"
            ignore_signals  = "…"
            java_options    = "…"
            min_tasks       = "…"
            name            = "…"
            order           = "no"
            priority        = "…"
            process_class   = "…"
            spooler_id      = "…"
            stop_on_error   = "yes"
            tasks           = "1"
            temporary       = "no"
            timeout         = "…"
            title           = "…"
            visible         = "yes"
        >

            <description> … </description>

            <lock.use lock="…"  …/>

            <environment>
                <variable name="…"  value="…"/>
                …
            </environment>

            <params>
                <param name="…"  value="…"/>
                …
            </params>

            <script com_class   = "…"
                    filename    = "…"
                    java_class  = "…"
                    language    = "…" >
                <include file="…"/>
                <![CDATA[  program-code… ]]>
            </script>



            <monitor name           = "…"
                     ordering       = "…">
               <script language="…"  …>…<script/>
            </monitor>

            <start_when_directory_changed  directory="…"  regex="…" />
```

```
<delay_after_error  delay="…"  error_count="…" />

<delay_order_after_setback  delay="…"  is_maximum="yes| no"
                            setback_count="…" />

<run_time let_run="no">
    <period begin        = "00:00"
            end          = "24:00"
            let_run      = "…"
            repeat       = "…"
            single_start = "…"
            when_holiday = "…"     />

    <date date="yyyy-mm-dd"/>
    …

    <weekdays>
        <day day="…">
           <period …/>
           …
        </day>
        …
    </weekdays>

    <monthdays>
        <day day="…">
            <period …/>
            …
        </day>
        <weekday weekday="…"  which="…">
            <period …/>
            …
        </weekday>
        …
    </monthdays>

    <ultimos>
        <day day="…">
            <period …/>
            …
        </day>
        …
    </ultimos>

    <month month="…">
        <period begin        = "00:00"
                end          = "24:00"
                let_run      = "…"
                repeat       = "…"
                single_start = "…"     />
        <weekdays>
            <day day="…">
               <period …/>
               …
            </day>
            …
```

```
                    </weekdays>
                    <monthdays>
                        <day day="…">
                            <period …/>
                            …
                        </day>
                        <weekday weekday="…" which="…">
                            <period …/>
                            …
                        </weekday>
                        …
                    </monthdays>

                    <ultimos>
                        <day day="…">
                            <period …/>
                            …
                        </day>
                        …
                    </ultimos>
                    …
                </month>

                <holidays>
                    <holiday date="yyyy-mm-dd"/>
                    <include file="…"/>
                    …
                </holidays>

                <at at="yyyy-mm-dd HH:MM:SS"/>
            </run_time>

            <commands on_exit_code="…"
                <start_job job="…" …>
                    <params>
                        <param name="…" value="…"/>
                        <copy_params from="task"/>
                        <copy_params from="order"/>
                        …
                    </params>
                </start_job>

                <add_order job_chain="…" …>
                    <params>
                        <param name="…" value="…"/>
                        <copy_params from="task"/>
                        <copy_params from="order"/>
                        …
                    </params>
                </add_order>
                …
            </commands>
        </job>
        …
    </jobs>
```

```
        <job_chains>
            <job_chain
                    distributed        = "no"
                    name               = "…"
                    orders_recoverable= "yes"
                    title              = "…"
                    visible            = "yes"
                >
                <file_order_source
                    delay_after_error = "…"
                    directory         = "…"
                    max               = "…"
                    next_state        = "…"
                    regex             = "…"
                    repeat            = "…"
                />
                …
                <job_chain_node
                    delay       = "…"
                    error_state = "…"
                    job         = "…"
                    next_state  = "…"
                    on_error    = "suspend| resume"
                    state       = "…"
                >

                    <on_return_codes
                    >

                        <on_return_code
                            return_code    = "…"
                        >
                            <add_order
                                xmlns        =
"https://jobscheduler-plugins.sos-berlin.com/NodeOrderPlugin"
                                job_chain    = "…"
                                id           = "…"
                            >
                                <params>
                                    <param name="…"  value="…"/>
                                    …
                                </params>
                            </add_order>
                            …
                            <to_state state="…" />
                        </on_return_code>
                        …
                    </on_return_codes>

                </job_chain_node>
                …
                <file_order_sink
                    move_to     = "…"
                    remove      = "yes"
                    state       = "…"
                />
                …
            </job_chain>
            <job_chain name="…">
                <job_chain_node.job_chain
                    job_chain   = "…"
                    error_state = "…"
                    next_state  = "…"
                    state       = "…"
                />
                …
                <job_chain_node.end
                    state       = "…"
                />
                …
            </job_chain>
            …
        </job_chains>

        <commands>
            …
        </commands>

    </config>
</spooler>
```

XML Elements

XML Element `<add_order>`

```
<add_order
   job_chain                    = "name"
   id                           = "id"
   replace                      = "yes|no"
   priority                     = "number"
   title                        = "text"
   state                        = "text"
   web_service                  = "name"
   at                           = "timestamp"   Order Starting Time
   end_state                    = "text"        State before which the order should be successfully
                                                completed and should leave the job chain
>
   params                   Parameters
   run_time                 The Job Run Time
   xml_payload
</add_order>
```

Adds a new order.

When the `<params>` element is specified, then the JobScheduler creates a `Variable_set` and makes it available in `Order.payload()`.

Example:

```
    <add_order job_chain="job_chain" id="1234" title="My First Order" state="100
at="now+3:00">

    <params>
        <param name="a_parameter" value="a value"/>

    </params>
</add_order>
```

Parent Elements

<commands> - XML Commands

Attributes

`job_chain`=*"name"*

The name of the job chain in which the order is being processed.

`id`=*"id"*

The alphanumerical identification of the order. (*Note that this parameter may not be set to* `id` *- which is an XML reserved word.*)

`replace`=*"yes|no"* (Initial value:yes)

`replace="no"`: `Job_chain.add_order()` will be called.
`replace="yes"`: `Job_chain.add_or_replace_order()` will be called.

`priority`=*"number"*

If two orders should be started at the same time then orders with a higher priority are processed first.

`title`=*"text"*

The title of the order.

`state`=*"text"*

`web_service`=*"name"*

When an order has been completed and the end of the job chain reached, it is then transformed with a style sheet and forwarded to a Web Service.

See `<web_service>` (page 83).

`at`=*"timestamp"* (Initial value:now) Order Starting Time `"now"`, `"yyyy-mm-dd HH:MM[:SS]"`, `"now + HH:MM[:SS]"` and `"now + SECONDS"` are possible.

See also `Order.at`.

`end_state`=*"text"* State before which the order should be successfully completed and should leave the job chain

See `Order.end_state`.

XML Element `<add_order>`

```
<add_order
    xmlns                         = "namespace"   Must be set to
                                                  https://jobscheduler-plugins.sos-berlin.com/NodeO
                                                  rderPlugin
    job_chain                     = "name"
    id                            = "id"
>
    params-node_order_plugi
    n
</add_order>
```

Adds a new order for a job chain. This order will be started immediately.

This element is derived from a different namespace https://www.sos-berlin.com/repository/scheduler/1.9/scheduler.xsd that includes support for the NodeOrderPlugin.

The `<add_order>` element is used with the `<on_return_code>` element to add an order in the event of a job task ending with a return code matching one of those specified in the `<on_return_code>` element's `return_code` attribute.

Note that the execution of an `<add_order>` element is not logged in the log file of the current current (i.e. originating) order.
Requirements

- The *NodeOrderPlugin* has been delivered as part of all JobScheduler releases since 1.9.0.
- The *NodeOrderPlugin* must also be activated in the `scheduler.xml` file as follows:

```
    <plugins>
        <plugin
java_class="com.sos.scheduler.engine.plugins.nodeorder.NodeOrderPlugin"/>
        …
    </plugins>
```

Example:

```
<on_return_codes>
    <on_return_code  return_code="0">
        <add_order id="myOrderId"
xmlns="https://jobscheduler-plugins.sos-berlin.com/NodeOrderPlugin" job_chain="Chain_B"
>
            <params >
                <param  name="parameterName" value="parameterValue"/>
            </params>
        </add_order>
    </on_return_code>
</on_return_codes>
```

Parent Elements

<on_return_code> - On Return Code

Attributes

`xmlns`=*"namespace"* Must be set to https://jobscheduler-plugins.sos-berlin.com/NodeOrderPlugin

The namespace of the plugin used. This attribute must be specified.

`job_chain`=*"name"*

The name of the job chain in which the order is being processed.

`id`=*"id"*

The identification for orders generated by the `<add_order>` element.

(*Note that this parameter should not be set to `id`, which is a reserved word in XML.*)

Example:

- `<add_order id="myOrderId" job_chain=""/>`

If the id attribute is not specified then orders generated by the `<add_order>` element inherit the `id` of the originating order.

Orders generated by the `<add_order>` element that are to be executed on job chains other than the originating job chain do not have to have an `id` attribute.

Orders generated by the `<add_order>` element for the originating job chain (i.e. creating a loop) have to have an `id` attribute.

Orders `id` attributes can be generated using variables. For example:

- `<add_order id="${ORDER_ID}.1" job_chain=""/>`

XML Element `<allowed_host>`

```
<allowed_host
    host                                = "host"
    level                          = "level"
> </allowed_host>
```

`<allowed_host>` - the name or IP address of a computer which is allowed to communicate with the JobScheduler.

The IP number may be a network address (class A, B or C), in which case all computers belonging to the network are allowed by default. A network address can be recognized in that the last part of the IP number is 0. Note that the JobScheduler handles the permissions for exact IP numbers with higher priority than network addresses.

Example:

```
<security>
    <allowed_host host="127.0.0.1"         level="all"/>
    <allowed_host host="admin.company.com" level="all"/>
    <allowed_host host="192.168.1.0"       level="info"/>
</security>
```

The IP addresses 127.0.0.1 and admin.company.com and the addresses in the Class-C-Network 192.168.1 are allowed to connect with the JobScheduler. The last entry, however, restricts the commands which may be carried out to those which deliver information.

Example:

```
<security>
    <allowed_host host="0.0.0.0" level="all"/>
</security>
```

Any computer has full access to the JobScheduler.

Behavior with `<base>`

Supplements the `<allowed_host>` element in the corresponding node of the basic XML configuration with the attribute `host=` . Attributes specified here have precedence over those entered in the basic XML configuration.

Parent Elements

<security> - Access Protection for TCP, HTTP and UDP

Attributes

`host`=*"host"*

host is the name or the IP address of the computer for which the access protection should be set. It can also be the IP address of a network.

An entry with the `host="0.0.0.0"` will be used for all hosts which are not specified here.

`level`=*"level"*

level="none"

The computer has no access.

level="signal"

The computer may carry out signalizing commands.

level="info"

The computer may carry out commands which convey information, but which do not change the JobScheduler's state.

level="no_add"

The computer has full access, with the exception of `<add_jobs>`, `<job>` and `<add_order>`.

level="all"

The computer has full access.

XML Element `<at>`

```
<at
   at                          = "yyyy-mm-dd hh:mm[:ss]"
> </at>
```

Parent Elements

<run_time> - The Job Run Time

Attributes

`at`=*"yyyy-mm-dd hh:mm[:ss]"*

Defines a starting point with date and time.

XML Element `<base>`

```
<base
   file                        = "filename"
> </base>
```

The element `<base>` references to a basic configuration. A basic configuration lies in a separate file and has the same structure as a XML-Konfiguration (page 6) (that is to say that it starts with `<spooler>`). Settings can be made in a basic configuration, and then either extended or overwritten by those of a higher level configuration. (A higher level configuration is that containing the `<base>` element which references a lower level configuration.)

The JobScheduler processes the basic configuration first.

For example, jobs can be defined in a basic configuration, and the higher level configuration can specify the job run times (`<run_time>`).

A basic configuration can call up another basic configuration. However, note that the JobScheduler does not test whether a basic configuration refers to itself.

`<base>` can be called repeatedly. The JobScheduler works through base configurations in the order in which they are called. A second basic configuration can either supplement or overwrite a first.

Parent Elements

<config> - Configuration

Attributes

`file`=*"filename"*

filename is the name of the file in which the basic configuration is held.

When the file name is not given absolutely, then the JobScheduler assumes that the file is to be found in the directory in which the `<base>` element is to be found.

The XML elements are executed in the basic configuration with the rights `<allowed_host level="all">`.

Environment variables (e.g. `$HOME`) are replaced by this attribute (see Settings which Allow Environment Variables to be Called (page 106)).

XML Element `<cluster>`

```
<cluster
    heart_beat_timeout          = "seconds"
    heart_beat_own_timeout      = "seconds"
    heart_beat_warn_timeout     = "seconds"
> </cluster>
```

This element can only be used in conjunction with the `-exclusive` or `-distributed-orders` options.

Behavior with `<base>`

Supplements the `<cluster>` element in the corresponding node of the basic XML configuration . Attributes specified here have precedence over those entered in the basic XML configuration.

Parent Elements

<config> - Configuration

Attributes

`heart_beat_timeout`=*"seconds"* (Initial value:60)

The time limit allowed for the heartbeat of another active JobScheduler in a cluster to be delayed before the JobScheduler is declared dead.

When a JobScheduler is inactive and operated with the `-exclusive` option set, then it may then become active.

Messages

[warn]	SCHEDULER-836	Deactivating that dead Scheduler
[warn]	SCHEDULER-996	No heart beat for seconds (time) - JobScheduler seems to be dead

`heart_beat_own_timeout`=*"seconds"* (Initial value:55)

This setting is used for self-regulation of a JobScheduler. When a JobScheduler notices that its own heartbeat has been delayed longer than the time specified in this parameter, then it immediately stops and restarts itself. This means that a JobScheduler preempts being deactivated by another Scheduler and avoids the short-term possibility of parallel operation because of a delayed heartbeat.

Messages

`[ERROR]`	`SCHEDULER-386`	Last heart beat was time, seconds ago. Something is delaying JobScheduler execution, the JobScheduler is aborted immediately

`heart_beat_warn_timeout`=*"seconds"* (Initial value:10)

The time allowed for a delay in the heartbeat of a JobScheduler before a warning is given out.

Messages

`[warn]`	`SCHEDULER-994`	No heart beat for seconds (time), expecting heart beat within seconds
`[warn]`	`SCHEDULER-995`	No heart beat for seconds (time), ignored for seconds because of recent database reconnect

XML Element `<commands>`

```
<commands
    on_exit_code                  = "exitcodes"
>
    add_jobs
    add_order                   Add an order
    modify_job
    modify_order
    modify_spooler
    scheduler_log.log_categ
    ories.reset
    scheduler_log.log_categ
    ories.set
    scheduler_log.log_categ
    ories.show
    show_history
    show_job
    show_jobs
    show_job_chains
    show_state
    start_job                   start a Task
    terminate
</commands>
```

This element allows a series of commands to be grouped together. The starting point for these commands then depends upon the relevant parent elements.

Parent Elements

<config> - Configuration

<job> - Definition of jobs

Attributes

`on_exit_code`=*"exitcodes"*

This attribute is mandatory within `<job>` - it cannot be used anywhere else.

Defines the exit codes which are to cause the commands listed here to be carried out. The following values can be specified here:

- a list of exit codes, separated by blanks.
- `on_exit_code="success"` is the same as `on_exit_code="0"`.
- `on_exit_code="error"` is valid for all exit codes with the exception of 0 and the exit codes already specified in other `<commands exit_code="…">`.
- Only for Unix systems: A task ending with a signal (either caused by the operating system command `kill` or a program error) has an exit code with the negative signal value. For example, for `kill` this would be -15.
- The following signal names may be used: `SIGHUP`, `SIGINT`, `SIGQUIT`, `SIGILL`, `SIGTRAP`, `SIGABRT`, `SIGIOT`, `SIGBUS`, `SIGFPE`, `SIGKILL`, `SIGUSR1`, `SIGSEGV`, `SIGUSR2`, `SIGPIPE`, `SIGALRM`, `SIGTERM`, `SIGSTKFLT`, `SIGCHLD`, `SIGCONT`, `SIGSTOP`, `SIGTSTP`, `SIGTTIN`, `SIGTTOU`, `SIGURG`, `SIGXCPU`, `SIGXFSZ`, `SIGVTALRM`, `SIGPROF`, `SIGWINCH`, `SIGPOLL`, `SIGIO`, `SIGPWR` and `SIGSYS`. Signal names which are not recognised by the operating system are ignored and a warning is given out.

See also `<job ignore_signals="…">` (page 42).

Example:

```
on_exit_code="0"
on_exit_code="1 2 3 99"
on_exit_code="error"
on_exit_code="SIGTERM SIGKILL"
```

Messages

[ERROR]	SCHEDULER-324	Invalid value for attribute exit_code="" in <commands>
[ERROR]	SCHEDULER-325	Attribute exit_code is not applicable here
[ERROR]	SCHEDULER-326	<commands on_exit_code="">: <commands> for exit code is already defined
[ERROR]	SCHEDULER-327	Last error occurred when executing command: xml_command
[warn]	SCHEDULER-337	Signal is unknown on this operating system and is ignored
[info]	SCHEDULER-328	Executing <commands on_exit_code="">:

XML Element `<config>`

```
<config
   central_configuration_direc  = "path"
   tory
   configuration_directory      = "path"         Path to the Configuration Directory
   configuration_add_event      = "job_path"     Job for Creating a New Configuration
                                                 File
   configuration_modify_event   = "job_path"     Job for Modifying a Configuration File
   configuration_delete_event   = "job_path"     Job for Deleting a Configuration File
   supervisor                   = "host:port"
   spooler_id                   = "spooler_id"
   port                         = "number"       Port Number for TCP and UDP
   tcp_port                     = "number"       Port for HTTP and TCP commands for
                                                 the JobScheduler
```

```
    udp_port                    = "number"          Port for UDP commands for the
                                                    JobScheduler
    param                       = "text"            For free use
    log_dir                     = "directory"       Protocol directory
    time_zone                   = "text"            JobScheduler time zone
    include_path                = "directory"       Directory path for <include>
    mail_xslt_stylesheet        = "path"            The path to the XSLT style sheet used
                                                    for the preparation of e-mails
    ip_address                  = "ip_number"       The interface IP address for TCP and
                                                    UDP
>
    base                    Basic Configuration
    params                  Parameters
    security                Access Protection for TCP, HTTP and UDP
    cluster                 Settings for cluster operation
    process_classes         Process Classes
    script                  Program code
    scheduler_script
    http_server             HTTP server
    holidays                Holidays
    jobs                    Jobs
    job_chains              Job Chains
    commands                XML Commands
</config>
```

`<config>` contains the JobScheduler configuration information - in particular, information related to the configuration of jobs. The `<config>` element can be repeated when the the `spooler_id` attribute changes. Should the `-id=` parameter not be specified on starting the JobScheduler, then the first specified `<config>` will be used. Otherwise the `spooler_id` attribute with the same value will be used.

Behavior with `<base>`

Supplements the `<config>` element in the corresponding node of the basic XML configuration with the attribute `spooler_id=` . Attributes specified here have precedence over those entered in the basic XML configuration.

Parent Elements

<spooler> - Root Element of the XML Document

Attributes

`central_configuration_directory`=*"path"* (Initial value:remote)

The default directory is `remote` in the configuration file directory - see `-config`.

This directory is for operation as a supervisory JobScheduler. Other Workload JobSchedulers register with the supervisory JobScheduler using `<config main_scheduler="…">`. This directory can contain a sub-directory, also containing configuration files, which the supervisory JobScheduler replicates when in operation.

Schedulers in a `-exclusive` or `-distributed-orders` cluster obtain their configuration from a directory with the name of the JobScheduler ID (`-id`).

JobSchedulers that do not belong to a cluster obtain their configuration from a directory whose name is made up from the computer's name and TCP port number (`<config tcp_port="…">`): *`hostname#tcpport`* .

In both cases, additional configuration files will be forwarded by the supervisory JobScheduler from its `_all` directory . These files are of lesser priority.

It is possible that no directory is specified for a Workload JobScheduler. In this case, however, the configuration from the `_all` directory, if available, will still be transmitted.

Messages

[warn]	SCHEDULER-454	Remote configuration directories '' and '' refer to the same IP number
[warn]	SCHEDULER-457	Remote JobScheduler '' has not been registered
[info]	SCHEDULER-455	No configuration directory for ''

`configuration_directory`=*"path"* (Initial value:live) Path to the Configuration Directory

The default setting is the `live` directory, which itself is to be found in the same directory as the configuration file - see `-config` .

Use of this directory takes the definitions of jobs, job chains and other objects away from the JobScheduler - see (page 88).

`configuration_add_event`=*"job_path"* Job for Creating a New Configuration File

The job in question is started when the JobScheduler loads a *new* file from its configuration directory.

The job is started when allowed by its `<run_time>` parameter. It should be defined in the JobScheduler configuration file, so that it is recognised when the JobScheduler reads the configuration directory.

The task which is started by this job has three parameters (which can be reached using `Task.params`):

The following task parameters are available as environment variables `SCHEDULER_LIVE_EVENT`, `SCHEDULER_LIVE_FILEPATH` and `SCHEDULER_LIVE_FILEBASE`.

`configuration_modify_event`=*"job_path"* Job for Modifying a Configuration File

The job in question is started when the JobScheduler loads a *modified* file from its configuration directory.

The job is started when allowed by its `<run_time>` parameter. It should be defined in the JobScheduler configuration file, so that it is recognised when the JobScheduler reads the configuration directory.

Parameters and environment variables are set as for `<config configuration_add_event="…">` .

`configuration_delete_event`=*"job_path"* Job for Deleting a Configuration File

The job in question is started when a file which has been loaded by the JobScheduler is removed from its configuration directory.

The job is started when allowed by its `<run_time>` parameter. It should be defined in the JobScheduler configuration file, so that it is recognised when the JobScheduler reads the configuration directory.

Parameters and environment variables are set as for `<config configuration_add_event="…">` .

`supervisor`=*"host:port"*

The supervisory JobScheduler, at which the current JobScheduler should log on and off. This takes place asynchronously and errors do not affect operation.

No more than 4 other (Workload) Schedulers should log on to a supervisory JobScheduler running on Windows.

The supervisor can set the configuration for the Workload (secondary) JobSchedulers, see. `<config central_configuration_directory="…">` (page 21). The JobScheduler retains this new configuration in its `cache` directory.

`spooler_id`=*"spooler_id"*

This element is only effective when its attribute is identical to the `-id=` parameter which was set as the JobScheduler was started, or when the `-id=` parameter was not set as the JobScheduler was started.

`port`=*"number"* (Initial value:0) Port Number for TCP and UDP

Combines the `tcp_port` and `udp_port` settings.

See also `<config tcp_port="…">` (page 21). and `<config udp_port="…">` (page 21).

Example:

```
<config port="4444">
```

The `-port` option has precedence over this parameter.

`tcp_port`=*"number"* (Initial value:0) Port for HTTP and TCP commands for the JobScheduler

The JobScheduler can accept commands via a TCP port whilst it is running. The number of this port is set here - depending on the operating system - with a number between 2048 and 65535. The default value is 4444.

The JobScheduler operates a HTTP/HTML server on the same port, enabling it to be reached using a web browser - e.g. via http://localhost:4444.

The JobScheduler does not respond to the `tcp_port=0` default setting either with TCP or HTTP protocols. This setting can therefore be used to block a JobScheduler from being accessed - for example via TCP.

See also `<config port="…">` (page 21).

Example:

```
<config tcp_port="4444">
```

The `-tcp-port` option has precedence over this parameter.

`udp_port`=*"number"* (Initial value:0) Port for UDP commands for the JobScheduler

The JobScheduler can also accept UDP commands addressed to the port specified in this setting. Note that a UDP command must fit in a message and that the JobScheduler does not answer UDP commands.

The default value of `udp_port=0` does not allow the JobScheduler to open a UDP port.

See also `<config port="…">` (page 21).

Example:

```
<config udp_port="4444">
```

The `-udp-port` option has precedence over this parameter.

`param`=*"text"* For free use

Sets the value of spooler.param (Object spooler, property param). For free use in scripts.

`log_dir`=*"directory"* Protocol directory

The directory in which the JobScheduler writes protocols.

`log_dir="*stderr"` allows the JobScheduler to write the main protocol to the standard error output (stderr).

`time_zone`=*"text"* JobScheduler time zone

Specifies the time zone in which a job or order is to start. Time zones are to be specified as defined in the tz database. A List of time zones is available in the Joda API, which is used in JobScheduler for the time functions.

The JobScheduler uses its local time if a time zone is not specified.

Example:

```
<config time_zone="Europe/Berlin">
```

The `-time-zone` option has precedence over this parameter.

`include_path`=*"directory"* Directory path for <include>

The directory of the files which are to be included by the `<include>` element.

Environment variables (e.g. `$HOME`) are replaced by this attribute (see Settings which Allow Environment Variables to be Called (page 106)).

The `-include-path` option has precedence over this parameter.

The `factory.ini` (section `[spooler]`, entry `include_path= …`) setting is overwritten by this parameter.

`mail_xslt_stylesheet`=*"path"* The path to the XSLT style sheet used for the preparation of e-mails

The path to the XSLT style sheet. XSLT style sheets are used by the JobScheduler for the preparation of e-mails. At the time of writing (April 2006) this subject is not documented.

`ip_address`=*"ip_number"* (Initial value:0.0.0.0) The interface IP address for TCP and UDP

The IP address to which the TCP and UDP ports are bound. The JobScheduler can then only be reached by way of this address.

A host name can also be specified.

The default setting is 0.0.0.0, which stands for all IP addresses.

When another IP address as 127.0.0.1 or localhost is given, then the JobScheduler cannot be reached by way of localhost.

The `-ip-address` option has precedence over this parameter.

XML Element `<content>`

```
<content > </content>
```

Parent Elements

<service_request> - Web-Dienst-Anforderung

XML Element `<copy_params>`

```
<copy_params
    from                                = ""                    Parameter Sources
> </copy_params>
```

The element hands over the parameters from a task or order. This can be used in `<start_job>` and `<add_order>` according to the following hierarchy:

```
<job>
...
<commands on_exit_value="…">
...
<start_job>
or
<add_order>
...

<params>
...
<copy_params from="order">
```

Should the same parameter name occur twice then the parameter value will be set according to the order in which the parameters with `<copy_params>` and `<param>` are specified. When the same parameter name occurs more than once, the last value will be used (any subsequent parameter overrides the value of a predecessor having the same name).

Parent Elements

<params> - Parameters

Attributes

`from`="" Parameter Sources

`from="task"`

Task parameters (`Task.params`) will be handed over.

from="order"

The parameters of the last order executed (`Order.params`) will be handed over. Note that the order parameters are available after a task has been carried out when:

- the job has been implemented with `<process>` or `<script language="shell">`;
- the `Task.end()` method has been called or
- an error has occurred during the execution.

An error (`SCHEDULER-329`) occurs when the job is not order driven (`<job order="no">`).

Messages

[ERROR]	SCHEDULER-329	<copy_params from=""/>: requested parameters are not available

XML Element `<copy_params>`

```
<copy_params
    from                        = ""              Parameter Sources
> </copy_params>
```

The element hands over the parameters from a task or order. This can be used in `<start_job>` and `<add_order>` according to the following hierarchy:

```
<job>
...
<commands on_exit_value="…">
...
<start_job>
or
<add_order>
...

<params>
...
<copy_params from="order">
```

Should the same parameter name occur twice then the parameter value will be set according to the order in which the parameters with `<copy_params>` and `<param>` are specified. When the same parameter name occurs more than once, the last value will be used (any subsequent parameter overrides the value of a predecessor having the same name).

Parent Elements

<params> - Parameters

Attributes

`from`="" Parameter Sources

`from="task"`

Task parameters (`Task.params`) will be handed over.

`from="order"`

The parameters of the last order executed (`Order.params`) will be handed over. Note that the order parameters are available after a task has been carried out when:

- the job has been implemented with `<process>` or `<script language="shell">`;
- the `Task.end()` method has been called or
- an error has occurred during the execution.

An error (`SCHEDULER-329`) occurs when the job is not order driven (`<job order="no">`).

Messages

[ERROR]	SCHEDULER-329	<copy_params from=""/>: requested parameters are not available

XML Element `<copy_params>`

```
<copy_params
    from                          = ""                    Parameter Sources
> </copy_params>
```

The element hands over the parameters from a task or order. This can be used in `<start_job>` and `<add_order>` according to the following hierarchy:

```
<job>
...
<commands on_exit_value="…">
...
<start_job>
or
<add_order>
...

<params>
...
<copy_params from="order">
```

Should the same parameter name occur twice then the parameter value will be set according to the order in which the parameters with `<copy_params>` and `<param>` are specified. When the same parameter name occurs more than once, the last value will be used (any subsequent parameter overrides the value of a predecessor having the same name).

Parent Elements

<params> - Parameters

Attributes

`from`="" Parameter Sources

`from="task"`

Task parameters (`Task.params`) will be handed over.

`from="order"`

The parameters of the last order executed (`Order.params`) will be handed over. Note that the order parameters are available after a task has been carried out when:

- the job has been implemented with `<process>` or `<script language="shell">`;
- the `Task.end()` method has been called or
- an error has occurred during the execution.

An error (`SCHEDULER-329`) occurs when the job is not order driven (`<job order="no">`).

Messages

[ERROR]	SCHEDULER-329	<copy_params from=""/>: requested parameters are not available

XML Element `<copy_params>`

```
<copy_params
    from                  = ""              Parameter Sources
> </copy_params>
```

The element hands over the parameters from a task or order. This can be used in `<start_job>` and `<add_order>` according to the following hierarchy:

```
<job>
...
<commands on_exit_value="…">
...
<start_job>
or
<add_order>
...

<params>
...
<copy_params from="order">
```

Should the same parameter name occur twice then the parameter value will be set according to the order in which the parameters with `<copy_params>` and `<param>` are specified. When the same parameter name occurs more than once, the last value will be used (any subsequent parameter overrides the value of a predecessor having the same name).

Parent Elements

<params> - Parameters

Attributes

`from=""` Parameter Sources

`from="task"`

Task parameters (`Task.params`) will be handed over.

`from="order"`

The parameters of the last order executed (`Order.params`) will be handed over. Note that the order parameters are available after a task has been carried out when:

- the job has been implemented with `<process>` or `<script language="shell">`;
- the `Task.end()` method has been called or
- an error has occurred during the execution.

An error (`SCHEDULER-329`) occurs when the job is not order driven (`<job order="no">`).

Messages

[ERROR]	SCHEDULER-329	<copy_params from=""/>: requested parameters are not available

XML Element `<date>`

```
<date
    date                              = "yyyy-mm-dd"
>
    period                            Operating period
</date>
```

defines the operating times for a particular day.

See also `<at>` (page 18).

Example:

```
<date date="2004-08-22">
    <period begin="10:00" end="12:00"/>
    <period begin="16:00" end="22:00"/>
</date>
```

Example:

```
<date date="2004-09-02" begin="14:00" end="18:00"/>
```

Behavior with <base>

Replaces the `<date>` element in the corresponding node of the basic XML configuration with the attribute `date=` .

Parent Elements

<run_time> - The Job Run Time

Attributes

`date`=*"yyyy-mm-dd"*

The Date.

XML Element `<day>`

```
<day
    day                                 = "number"
>
    period                           Operating period
</day>
```

Defines the periods for a particular day.

Behavior with <base>

Replaces the `<day>` element in the corresponding node of the basic XML configuration with the attribute `day=` .

Parent Elements

<weekdays> - Operating Periods for Weekdays

<monthdays> - Operating periods on particular days of the month

<ultimos> - Ultimos - Operating Periods for Particular Days of the Month - Counted from the End of the Month

Attributes

`day`=*"number"*

defines the day number, which is dependant on the context of the parent element. In the case of days of the week, the English names can be entered here. Note that the names must begin with a small letter.

Mehrere Tage können durch Leerzeichen getrennt angegeben werden.

XML Element `<delay_after_error>`

```
<delay_after_error
    error_count                 = "integer"
    delay                       =
                                "seconds|HH:MM|HH:MM:SS|stop"
> </delay_after_error>
```

See `Job.delay_after_error`.

Example:

```
<job …>
    <script …>…</script>
    <delay_after_error error_count= "2" delay="10"    />   <!-- 10 Seconds -->
    <delay_after_error error_count= "5" delay="00:01" />   <!-- One Minute -->
    <delay_after_error error_count="10" delay="24:00" />   <!-- A Day -->
    <delay_after_error error_count="20" delay="STOP"  />
</job>
```

The job is immediately restarted after the first error.
The job is restarted after a delay of 10 seconds after the 2nd, 3rd & 4th consecutive errors,

after between 5 and 9 errors the job is delayed each time by a minute,

after between 10 and 19 errors the job is delayed 24 hours,

after 20 consecutive errors the JobScheduler stops the job immediately.

Parent Elements

<job> - Definition of jobs

Attributes

`error_count`=*"integer"*

The number of consecutively occurring errors before which a job will be delayed.

`delay`=*"seconds|HH:MM|HH:MM:SS|stop"*

Delay before the job will be rerun.

`delay="stop"` or `delay="STOP"` stops a job after the specified number of consecutive errors.

XML Element `<delay_order_after_setback>`

```
<delay_order_after_setback
    setback_count               = "integer"
    delay                       = "seconds|HH:MM|HH:MM:SS"
    is_maximum                  = "yes|no"
> </delay_order_after_setback>
```

See `Job.delay_order_after_setback`, `Job.max_order_setbacks` and `Order.setback()`.
- after the 1st attempt is failed the job will be executed again after 10min.
- after the 5th attempt is failed the job will be executed again after 30min.
- after the 10th attempt the job will run one time again and goes into the error state if it fails.

Example:

```
<delay_order_after_setback setback_count="1" is_maximum="no" delay="00:10"/>
<delay_order_after_setback setback_count="5" is_maximum="no" delay="00:30"/>
<delay_order_after_setback setback_count="10" is_maximum="yes" />
```

Parent Elements

<job> - Definition of jobs

Attributes

`setback_count`=*"integer"*

The number of successive setbacks occurring for an order. Different delays can be set for each setback - e.g. 1st setback, 1 second; 2nd setback, 10 seconds; etc.
Specifies the number of sebacks after which an XML element applies.

For example, where `setback_count=5`, the element applies after the 5th setback.

`delay`=*"seconds|HH:MM|HH:MM:SS"*

The period an order waits after a setback before being restarted in a job.

`is_maximum`=*"yes|no"* (Initial value:no)

`setback_count=` specifies the maximum number of sequential setbacks allowed. A further setback occurring after this number of setbacks has been reached (`Order.setback()`) causes the JobScheduler to give the order the error state `Job_chain_node.error_state` .

See `Job.max_order_setbacks` .

XML Element `<description>`

`<description >`	
`include`	Includes text from a file
`</description>`	

A description of a job which will be shown in the HTML interface. The text should be coded in HTML. (This is only possible if `<![CDATA[ ...]]>` or `<include>` is used, because of the strict DTD.)

Example:

```
<job name="my_job">

    <description>

This is the description of my job:
<include file="description_of_my_job.txt"/>


    </description>
    ...
</job>
```

Behavior with <base>

Replaces the `<description>` element in the corresponding node of the basic XML configuration .

Parent Elements

<job> - Definition of jobs

XML Element `<environment>`

```
<environment >
   variable                      A Variable
</environment>
```

Defines additional environment variables for a process.

Parent Elements

<job> - Definition of jobs

<process> -

<start_job> - start a Task

XML Element `<file_order_sink>`

```
<file_order_sink
   state                           = "string"
   remove                          = "yes|no"
   move_to                         = "directory_path"
> </file_order_sink>
```

`<file_order_sink>` is implemented using the internal `scheduler_file_order_sink` job.

The order is completed after the operation.

Should it not be possible to move or remove a file, then the order will be added to the blacklist. This ensures that the still existing file can result in a new order being started. Blacklisted orders can still be removed using <

`remove_order>` . Once the JobScheduler realises that a blacklisted file has been removed, then the file will be deleted from the list. The file can then be re-added to the monitored directory if required.

See also Directory Monitoring with File Orders (page 165) and `<file_order_source>` (page 35).

Parent Elements

<job_chain> - Job chain

Attributes

`state`=*"string"*

The state valid for a job chain node. This state is an end state.

`remove`=*"yes|no"*

`remove="yes"` removes the file.

Messages

[info]	SCHEDULER-979	Removing file

`move_to`=*"directory_path"*

The file will be moved to the directory specified. An already existing file of the same name will be overwritten.

On Unix systems the file can only be moved within the same file system.

Environment variables (e.g. `$HOME`) are replaced by this attribute (see Settings which Allow Environment Variables to be Called (page 106)).

Messages

[info]	SCHEDULER-980	Moving file to

XML Element `<file_order_source>`

```
<file_order_source
    directory                    = "directory_path"
    regex                        = "regex"
    delay_after_error            = "seconds"
    repeat                       = "no|seconds"
    max                          = "integer"
    next_state                   = "string"
    alert_when_directory_miss    = "boolean"
    ing
> </file_order_source>
```

Adds a file order source to a job chain. Every file in the source directory with a name corresponding to a regular expression can be added to the job chain as a file order.

See also Directory Monitoring with File Orders (page 165) and `<file_order_sink>` (page 34).

Example:

```
<job_chain name="my_job_chain">
    <file_order_source directory="/tmp/input"/>
    <job_chain_node state="first" job="process_file" error_state="ERROR"/>
    <file_order_sink state="remove" remove="yes"/>
    <file_order_sink state="ERROR" move_to="/tmp/input.error"/>
</job_chain>
```

Parent Elements

<job_chain> - Job chain

Attributes

`directory`=*"directory_path"*

the path to the directory containing the files.

Environment variables (e.g. `$HOME`) are replaced by this attribute (see Settings which Allow Environment Variables to be Called (page 106)).

`regex`=*"regex"*

A regular expression used to select files according to their names.

`delay_after_error`=*"seconds"* (Initial value:repeat)

The default setting is the `repeat="…"` attribute value.

Should the directory not be readable, then the JobScheduler will first send an e-mail and then repeatedly try to read the directory until it is successful. The JobScheduler will then send a further mail.

Messages

`[info]`	`SCHEDULER-984`	Recovered from previous error in directory

`repeat`=*"no|seconds"*

The JobScheduler checks for changes in the directory on a regular basis. . The length of time between these checks can be set here.

The default setting on Windows systems is `repeat="60"`. Further, the JobScheduler also uses the Windows system directory monitoring, in order to be able to react immediately to a change in a directory. This is renewed regularly after the repeat interval has elapsed.

On Unix systems the default value is `repeat="10"`. This means that the directory is checked every 10 seconds for changes.

Messages

`[info]`	`SCHEDULER-984`	Recovered from previous error in directory

`max`=*"integer"* (Initial value:100)

The maximum number of files to be taken as orders. Should more files be present, then these extra files are taken on as soon as the first job in the job chain can take on a new order.

Messages

```
[info]    SCHEDULER-985
[info]    SCHEDULER-986
```

`next_state`=*"string"*

Should it not be possible to start the orders in the first job of the job chain, then the initial state of the orders can be specified using this attribute.

`alert_when_directory_missing`=*"boolean"*

The warning for a missing directory can be activated/deactivated using this attribute.

XML Element `<holiday>`

```
<holiday
    date                              = "yyyy-mm-dd"
> </holiday>
```

Defines a holiday - a day on which the JobScheduler should not run a job.

Example:

```
<holiday date="2004-12-24"/>
```

Behavior with `<base>`

Replaces the `<holiday>` element in the corresponding node of the basic XML configuration with the attribute `date=` .

Parent Elements

<holidays> - Holidays

Attributes

`date`=*"yyyy-mm-dd"*

The date of the holiday.

XML Element `<holidays>`

```
<holidays >
    weekdays                    Operating Periods for Weekdays
    holiday                     Holidays on Which a Job Should not Run
    include                         Includes text from a file
</holidays>
```

Example:

```
<holidays>
    <holiday date="2004-12-24"/>
    <holiday date="2004-12-25"/>
    <holiday date="2004-12-26"/>
    <holiday date="2004-12-31"/>
    <include file="holidays-2007.xml"/>
    <include file="holidays-2008.xml"/>
</holidays>
```

Parent Elements

<config> - Configuration

<run_time> - The Job Run Time

XML Element `<http.authentication>`

```
<http.authentication
    scheme                          = "scheme"                    Authentication
>
    http.users                      HTTP User Authentication
</http.authentication>
```

The JobScheduler usess this element to obtain authentication from the HTTP client (browser) according to the »Basic« RFC 2617 scheme.

The authentication also applies for the `<web_service>`.

Example:

```
<http_server>
    <http.authentication>
        <http.users>
            <http.user name="Rose Kemp"
password_md5="701d051b67bc5fc7c7c919d01f0aa7cb"/>
            <http.user name="Jeff Beck"
password_md5="eb6801a466d5376639e29cd1d11ecb9f"/>
        </http.users>
    </http.authentication>
    …
</http_server>
```

Behavior with `<base>`

Supplements the `<http.authentication>` element in the corresponding node of the basic XML configuration . Attributes specified here have precedence over those entered in the basic XML configuration.

Parent Elements

<http_server> - HTTP server

Attributes

`scheme`=*"scheme"* (Initial value:basic) Authentication

Only the `scheme="basic"` authentication scheme is implemented.

XML Element `<http.user>`

```
<http.user
    name                         = "name"             User Authentication
    password_md5                 = "string"           The Password MD5 Sum
> </http.user>
```

See `<http.authentication>` (page 38).

Behavior with `<base>`

Replaces the `<http.user>` element in the corresponding node of the basic XML configuration with the attribute `name=` .

Parent Elements

<http.users> - HTTP User Authentication

Attributes

`name`=*"name"* User Authentication

`password_md5`=*"string"* The Password MD5 Sum

On Unix systems the MD5 password check-sum can generally be obtained using the `md5sum` command:

```
echo -n 'password' | md5sum
```

The MD5 sum comprises only the hexadecimal characters 0-9, a-f and A-F and is 32 characters long.

XML Element `<http.users>`

```
<http.users >
    http.user                 HTTP User Authentication
</http.users>
```

See `<http.authentication>` (page 38).

Behavior with `<base>`

Supplements the `<http.users>` element in the corresponding node of the basic XML configuration . Attributes specified here have precedence over those entered in the basic XML configuration.

Parent Elements

<http.authentication> - HTTP Authentication

XML Element `<http_directory>`

```
<http_directory
    url_path                        = "url_path"          The first directory in a URL path
    path                            = "path"              File system path
> </http_directory>
```

Specifies a directory in the file system which is to be mapped to a directory in a URL path,

The example returns the URL `http://host:port/doc/xml/http_directory.html`
and the file `c:\pub\html\doc\xml/http_directory.html`.

Behavior with `<base>`

Supplements the `<http_directory>` element in the corresponding node of the basic XML configuration . Attributes specified here have precedence over those entered in the basic XML configuration.

Parent Elements

<http_server> - HTTP server

Attributes

`url_path`=*"url_path"* The first directory in a URL path

Specifies the first directory in a URL path, which is be mapped from a file system directory.

The directory specified should start with a forward stroke (/).

Example:

```
<http_directory url_path="/doc/" path="c:/html/my_doc/" />
```
Creates the `c:/html/my_doc/` directory on the URL starting with `/doc/`.

`path`=*"path"* File system path

The file system path to be mapped to the directory specified in `url_path=""`.

Example:

```
path="c:\pub\html\doc\"
```

Environment variables (e.g. `$HOME`) are replaced by this attribute (see Settings which Allow Environment Variables to be Called (page 106)).

XML Element `<http_server>`

```
<http_server >
   http_directory          HTTP File Directory
   web_service             Web Service
</http_server>
```

Behavior with `<base>`

Supplements the `<http_server>` element in the corresponding node of the basic XML configuration . Attributes specified here have precedence over those entered in the basic XML configuration.

Parent Elements

<config> - Configuration

XML Element `<include>`

```
<include
   file                          = "filename"      Path to file to be included
   live_file                     = "path"          Path to the file to be added from the
                                                   configuration directory
   node                          = "xpath"         XPath expression
> </include>
```

This element may be included in the text (outside of `<![CDATA[` and `]]>`), in order to include source code which is to be found in other files.

The file is read on a remote computer when such a computer is specified using `<process_class remote_scheduler="…">`.

Should the JobScheduler not be able to read a file:

- in the `<description>` element: the JobScheduler ignores errors but adds the error message to the `<description>`.
- in the `<script>` element: the JobScheduler puts the job in the `read_error` state. The `<modify_job cmd="reread">` command allows the JobScheduler to reread a script.

Changes in include-files have no impact on a running JobScheduler.

Parent Elements

<script> - Program code

<description> - Description

<holidays> - Holidays

<params> - Parameters

Attributes

`file`=*"filename"* Path to file to be included

The name of the file whose content is to be included. Should the name of the file not be absolute, then the JobScheduler assumes different directories, independent of surrounding XML elements:

Environment variables (e.g. `$HOME`) are replaced by this attribute (see Settings which Allow Environment Variables to be Called (page 106)).

`live_file`=*"path"* Path to the file to be added from the configuration directory

This attribute can be used directly in:

- `<job><description>`
- `<job><params>`
- `<holidays>`
- `<script>`

and specifies, for the file whose content is to be included, the file path relative to the directory of the file which the `<include>` came from. This file must lie within the configuration directory tree. "`/`" at the beginning of this path denotes the configuration directory root. Windows drive letters cannot be used.

When the file containing the `<include>` does not come from a configuration directory, the JobScheduler assumes the configuration root directory of the installation.

A change in the file for a file-based job or order under `<job><params>` or `<order><params>` to be re-read when the job or order comes from a configuration directory.

[ERROR]	SCHEDULER-461	Path reaches beyond root (too many '..'):
[ERROR]	SCHEDULER-417	Invalid name: ''

Environment variables (e.g. `$HOME`) are replaced by this attribute (see Settings which Allow Environment Variables to be Called (page 106)).

`node`=*"xpath"* XPath expression

Only applies for `<include>` in `<params>` and selects the `<param>` elements. The default setting is `xpath="params/*"`.

XML Element `<job>`

```
<job
    spooler_id              = ""
    name                    = "jobname"
    title                   = "text"
    order                   = "yes_no"              Order Controlled Job
    process_class           = "process_class"
    tasks                   = "number"              The maximum number of tasks
    min_tasks               = "number"              The minimum number of tasks
                                                    kept running
    timeout                 = "duration"            The time allowed for an
                                                    operation
    idle_timeout            = "duration"            Limit for the waiting_for_order
                                                    state
    force_idle_timeout      = "yes_no"              Task ended by idle_timeout
                                                    despite min_task
    priority                = "process_priority"
    temporary               = "yes_no"
    java_options            = "string"
    visible                 = "yes|no|never"
    ignore_signals          = "all|signalnames"
    stop_on_error               = "yes|no"
    replace                     = "yes|no"
    warn_if_shorter_than        =
                                "HH:MM:SS|seconds|percentage%"
    warn_if_longer_than         =
                                "HH:MM:SS|seconds|percentage%"
    enabled                     = "yes|no"                        Disable a Job.
>
    description             Description
    lock. use               Lock declaration
    environment             Environment Variables
    params                  Parameters
    script                  Program code
    monitor                 Job Monitor
    start_when_directory_ch Directory Monitoring
    anged
    delay_after_error       Job Delay after an Error
    delay_order_after_setba Delay Order after Setback
    ck
    run_time                The Job Run Time
    commands                XML Commands
</job>
```

Defines a job with program code, runtime, etc.

Behavior with `<base>`

Supplements the `<job>` element in the corresponding node of the basic XML configuration with the attribute `name=` . Attributes specified here have precedence over those entered in the basic XML configuration.

Parent Elements

<jobs> - Jobs

Attributes

`spooler_id`=""

`name`=*"jobname"*

Every job has a unique name.

Should a job with the same name be defined in one of the basic configurations, then this parameter can be used to change or supplement the settings made in that job.

`title`=*"text"*

A description of the job (max. 1 line).

`order`=*"yes_no"* Order Controlled Job

`order="yes"` defines a job as being order controlled. The JobScheduler will only start an order controlled job when an order for the job exists.

A script can check this attribute using the `Job.order_queue` method.

`process_class`=*"process_class"*

Defines the name of the process class in which the job should run. Note that process classes are defined with `<process_classes>`.

`tasks`=*"number"* (Initial value:1) The maximum number of tasks

A number of tasks can run in parallel from one job. This attribute specifies the maximum number of tasks for a job.

`<lock.use>`: only `tasks="1"`, which only allows one task, makes sense in combination with an exclusive lock.

`min_tasks`=*"number"* (Initial value:0) The minimum number of tasks kept running

The JobScheduler keeps this minimum number of tasks running. This allows order controlled tasks, which require a long time to initialize, to be held in waiting.

Note that the `<job tasks="…">` element must be large enough.

The JobScheduler will start additional tasks when:

- it is starting
- a task has ended
- at the start of a new Period (`<run_time>`)
- or the job is stopped by `<modify_job cmd="unstop">`
- or all job tasks are continued with `<modify_job cmd="continue">`
- or the job is wakened by `<modify_job cmd="wake">`

- and fewer tasks than specified in `min_tasks` are running
- and the start is allowed according to the `<run_time>` period
- and the job has either the `pending` or `running` states (that is, it is not being stopped or has not already stopped).

In order to prevent an overload, the JobScheduler does not start any new tasks immediately after a task has been completed. The completion of a task causes a new one to be started only in the following situations:

- `spooler_process()` is called
- The task has been waiting some time on an order (the state `running_waiting_for_order`, when `<job idle_timeout="0">`)
- The task could start but is delayed by the `Task.delay_spooler_process` method
- The task is a process (`<script language="shell">`) and the process has not terminated itself immediately after starting

Messages

[ERROR]	SCHEDULER-322	min_tasks= is greater than max_tasks=
[warn]	SCHEDULER-970	task ended immediately after start, so min_tasks= doesn't lead to new tasks
[debug3]	SCHEDULER-969	Less than min_tasks= are running. New tasks will be started. Reason:

`timeout`=*"duration"* The time allowed for an operation

Limits the duration of a task operation (spooler_open, spooler_process etc.) or the whole task for a non-API task (`<script language="shell">`). An error does not occur when the priority of a job has not been set. Should a task exceed the time allowed, then the JobScheduler aborts the task.

duration can be specified in seconds or in the `HH:MM` or `HH:MM:SS` formats.

`idle_timeout`=*"duration"* (Initial value:5) Limit for the waiting_for_order state

Limits the idle time of an order controlled job (`order="yes"`). When a task is waiting on the next order and this idle time is exceeded, then the JobScheduler ends the task.

The *duration* can be specified in seconds or in the `HH:MM` or `HH:MM:SS` formats.

`idle_timeout="never"` allows a job to run indefinitely, only limited by `<run_time>`.

See also `<job force_idle_timeout="…">` (page 42).

`force_idle_timeout`=*"yes_no"* (Initial value:no) Task ended by idle_timeout despite min_task

Note that this is only effective with `<job min_tasks ≥ "0">` and `<job idle_timeout="…">`.

`force_idle_timeout="yes"` ends a task after `idle_timeout` has expired, even when fewer tasks than specified in `min_tasks` are running. `min_tasks` only starts new tasks after the task termination.

In this way tasks can be ended which may not take up a resource such as a database too long when idle.

`priority`=*"process_priority"*

Sets the priority of a task.

This attribute can be given the following values: `idle`, `below_normal`, `normal`, `above_normal` and `high` or the numerical values allowed for the operating system being used.

An error does not occur when the priority of a job is not set.

A task with a higher priority can block the computer.

See `Task.priority_class`.

`temporary`=*"yes_no"*

`temporary="yes"` defines a job as being temporary. This setting is only for `<add_jobs>`. A job will then be deleted after being carried out and will no longer be recognized.

`java_options`=*"string"*

Is only effective when a job runs as its own process (see `<process_classes>` (page 70)) and either the job or monitor is implemented in Java. The options are handed over together with the commandline option `-job-java-options` Java options. The interpretation of these options depends on Java.

`visible`=*"yes|no|never"* (Initial value:yes)

`visible="no"` and `visible="never"` make a job invisible in the results of `<show_jobs>` and `<show_state>`.

The JobScheduler makes a job visible as soon as a task has been loaded when `visible="no"` is set. When `visible="never"` is set, then this job will never be returned when querying using the `<show_state>` command.

`ignore_signals`=*"all|signalnames"* (Initial value:no)

Is only relevant for UNIX systems.

A job whose task process ends with a signal, causes the job to be stopped. Signals are sent when the task ends either by way of the `kill` system command or by way of a program being aborted.

If `ignore_signals` has not been specified, then a task ending with a signal stops the job (with the message `SCHEDULER-279`).

`ignore_signals="all"` means that a job will not be stopped by a signal.

A list of signal names (separated by blanks) can be specified instead of `"all"`. The following signal names are recognized, depending on the operating system: `SIGHUP`, `SIGINT`, `SIGQUIT`, `SIGILL`, `SIGTRAP`, `SIGABRT`, `SIGIOT`, `SIGBUS`, `SIGFPE`, `SIGKILL`, `SIGUSR1`, `SIGSEGV`, `SIGUSR2`, `SIGPIPE`, `SIGALRM`, `SIGTERM`, `SIGSTKFLT`, `SIGCHLD`, `SIGCONT`, `SIGSTOP`, `SIGTSTP`, `SIGTTIN`, `SIGTTOU`, `SIGURG`, `SIGXCPU`, `SIGXFSZ`, `SIGVTALRM`, `SIGPROF`, `SIGWINCH`, `SIGPOLL`, `SIGIO`, `SIGPWR` und `SIGSYS`. Signal names which are not recognized by an operating system are ignored and a warning given.

Note that because a task ending with a signal which may be ignored can cause a TCP connection error (`ECONNRESET`), the JobScheduler is so configured that TCP connection errors only lead to a job being stopped when `ignore_signals="…"` does not apply. The JobScheduler reacts to this situation with the `SCHEDULER-974` message.

See also `<commands on_exit_code="SIGTERM">` (page 20) and the Gnu `man 7 signal` command.

Example:

```
<job name="my_job" ignore_signals="SIGTERM SIGKILL">
```

Messages

[warn]	SCHEDULER-279	Process terminated with signal (name)
[warn]	SCHEDULER-337	Signal is unknown on this operating system and is ignored
[info]	SCHEDULER-974	Last error does not stop the job if the task aborts (after kill or crash) with any signal listed in ignore_signals="". In this case, expect warning SCHEDULER-279

`stop_on_error`=*"yes|no"* (Initial value:yes)

The default `stop_on_error="yes"` setting stops a job when a task ends with an exception or has an "`[ ERROR]` " message in its log file. An error message can then be written, for example with `Log.error()` .

Specifying `stop_on_error="no"` does not allow a job to be stopped in this situation. Should `spooler_process()` end with an exception, then the JobScheduler sets the order in the error state (`<job_chain_node error_state="…">`).

This setting is not effective, when either `<delay_after_error>` or `Job.delay_after_error` are used.

Messages

[warn]	SCHEDULER-846	After task exception and due to stop_on_error='no', the order has been moved to error_state=''
[debug3]	SCHEDULER-977	Job is not stopping because of <job stop_on_error="no">. Task error was:
[debug3]	SCHEDULER-978	Job is stopping because of <job stop_on_error="yes">. Task error was:

`replace`=*"yes|no"* (Initial value:yes)

`replace="yes"` causes any already existing job definition to be replaced. Should this not be possible - for example because a task is running - then the JobScheduler will replace the job later.

An order controlled job (`<job order="yes">`) cannot be replaced.

`warn_if_shorter_than`=*"HH:MM:SS|seconds|percentage%"*

If a job is completed in less than the specified time allowed then the JobScheduler will give out the `SCHEDULER-711` warning.

The time can be specified in `HH:MM` or `HH:MM:SS` formats, as a number of seconds or as a percentage. A percentage value is calculated from the average length of the job steps carried out up to this point as specified in the `SCHEDULER_HISTORY` database table.

`warn_if_longer_than`=*"HH:MM:SS|seconds|percentage%"*

If a job is completed in more than the specified time allowed then the JobScheduler will give out the `SCHEDULER-711` warning.

The time can be specified in `HH:MM` or `HH:MM:SS` formats, as a number of seconds or as a percentage. A percentage value is calculated from the average length of the job steps carried out up to this point as specified in the `SCHEDULER_HISTORY` database table.

`enabled`=*"yes|no"* (Initial value:yes) Disable a Job.

With this attribute a Job can be disabled. It has the same effect as the stop command but is different displayed in the Operations GUI.

XML Element `<job_chain>`

```
<job_chain
    name                        = "name"
    visible                     = "yes|no|never"
    orders_recoverable          = "yes|no"
    distributed                 = "yes|no"
    title                       = "String"
    max_orders                  = "postive integer"
    process_class               = "String"
    file_watching_process_cla   = "String"
    ss
>
    file_order_source        File Order Source
    job_chain_node           Job Chain Nodes
    file_order_sink          File Order Sink
    job_chain_node.job_chai  Job Chain Nodes
    n
    job_chain_node.end       Job chain ends
</job_chain>
```

Adds a new job chain (see the `Job_chain` class).

See Orders (page 163), `Job_chain`, `Spooler.create_job_chain()` and `Spooler.add_job_chain()`

Simple Job Chains - a Chain of Jobs

Simple job chains contain jobs and are described using the `<job_chain_node>`, `<file_order_source>` and `<file_order_sink>` XML elements.

Superordinate Job Chains - a Chain of Job Chains

Superordinate job chains refer to other job chains and are described using the `<job_chain_node.job_chain>`. `<job_chain_node.end>` XML elements. A superordinate job chain can only contain simple job chains.

Superordinate job chains cannot be used in combination with distributed orders.

Example:

```
<job_chains>
    <job_chain name="Chain_A">
        <job_chain_node state=  "1" job="job_a" next_state=  "2" error_state="999" />

        <job_chain_node state=  "2" job="job_b" next_state="100" error_state="999" />

        <job_chain_node state="100" />
        <job_chain_node state="999" />
    </job_chain>
</job_chains>
```

This is the same as the following JobScheduler script:

```
<script language="javascript"><![CDATA[
    var job_chain = spooler.create_job_chain();
        job_chain.name = "Chain_A";
        job_chain.add_job( "job_a", 1,   2, 999 );
        job_chain.add_job( "job_b", 2, 100, 999 );
        job_chain.add_end_state( 100 );
        job_chain.add_end_state( 999 );
    spooler.add_job_chain( job_chain );
]]></script>
```

Example:

```
<job_chains>
    <job_chain name="superchain">
        <job_chain_node.job_chain state="A" job_chain="job_chain_a"
                                  next_state="B" error_state="ERROR" />
        <job_chain_node.job_chain state="B" job_chain="job_chain_b"
                                  next_state="OK" error_state="ERROR" />
        <job_chain_node.end state="OK" />
        <job_chain_node.end state="ERROR" />
    </job_chain>
</job_chains>
```

Behavior with `<base>`

This element may not be specified here when it has already been specified in the basic XML configuration.

`<job_chain name="`Name`">` can be added with unique name.

Parent Elements

<job_chains> - Job Chains

Attributes

`name`=*"name"*

The name of the job chain. Note that a job chain can only be defined once.

`visible`=*"yes|no|never"* (Initial value:yes)

`visible="no"` and `visible="never"` make a job chain invisible in the results of `<show_job_chains>` and `<show_state>`.

The JobScheduler makes a job chain visible as soon as an order has been added to the chain.

`orders_recoverable`=*"yes|no"* (Initial value:yes)

`orders_recoverable="yes"`

When the JobScheduler has been configured to store orders in the database, as soon as an order is added to a job's order queue it will be also be stored in the database. After the order has completed the job chain, it will be deleted from the database

The JobScheduler loads orders from the database on starting and setting up the job chains. See `Spooler.add_job_chain()`.

This attribute does not function when the JobScheduler has been configured to work without a database - see `factory.ini` (section `[spooler]`, entry `db= ...`) (page 98).

`orders_recoverable="no"`

The JobScheduler does not store orders in or load orders from a database.

See Database (page 136).

`distributed`=*"yes|no"* (Initial value:no)

Only works when specified in conjunction with `-distributed-orders` and causes orders to be distributed over more than one JobScheduler.

`distributed="no"` prevents a job chain from being processed by more than one JobScheduler. Instead, causes a job chain to be processed on the one JobScheduler, as if it were in a non-distributed environment. Note that in this situation, the name of the job chain *must* be unique in the cluster (Note that this is not checked by the JobScheduler).

`title`=*"String"*

A job chain can also be given a title.

See also `Job_chain.title`.

`max_orders`=*"postive integer"* (Initial value:99999)

In general an unlimited number of orders for a job_chain could ran simultaneously. To declare the attribute max_orders it is possible to limit this number. E.g. *max_orders='1'* let run the job_chain exclusively for one order. A new order for this job_chain can only run, if the first order was finished.

The number of simultaneous orders for a job_chain is not limited, if this attribute is not set.

This attribute will take only effect if the order starts with the first node of the job chain.

`process_class`=*"String"*

A `<process_class>` can be used to to specify the default Agent or Remote JobScheduler to be used for processing the job chain.

See How to execute Jobs and Job Chains with Agents for more information.

`file_watching_process_class`=*"String"*

A `<process_class>` can be used to to specify an Agent that is to be used to watch for files arriving at the Agent host. This Agent triggers file orders for incoming files.

This process class is used to define both the `<file_order_source>` and `<file_order_sink>`.

If the `file_watching_process_class` attribute is empty, it defaults to the process class of the job chain.

The process class must denote only one agent.

Changing the process class at runtime will have no effect on the `file_order_source` or `file_order_sink`.

When the process contains both the `remote_scheduler` attribute and the `<remote_schedulers>` element, the `remote_schedulers` element will be ignored when the process class is used for the `file_watching_process_class` attribute or `file_watching_process_class` in the `<job_chain>` element.

See the JobScheduler Universal Agent - Remote File Watching article for more information.

XML Element `<job_chain_node>`

```
<job_chain_node
    state                        = "string"
    job                          = "job_name"
    next_state                   = "string"
    error_state                  = "string"
    delay                        = "seconds"
    on_error                     = "suspend|setback"
>
    on_return_codes            On Return Codes
</job_chain_node>
```

Adds a new job chain node to a job chain (see the `Job_chain_node` class).

The XML elements

```
<job_chain_node state="STATE"       job="JOB" next_state="NEXT_STATE"
error_state="ERROR_STATE"/>
<job_chain_node state="ERROR_STATE" />
```

correspond to the following API calls

```
job_chain.add_job( "JOB", "STATE", "NEXT_STATE", "ERROR_STATE" );
job_chain.add_end_state( "ERROR_STATE" );
```

See `Job_chain_node` , `Job_chain.add_job()` and `Job_chain.add_end_state()` .

Parent Elements

<job_chain> - Job chain

Attributes

`state`=*"string"*

The state valid for a job chain node.

`job`=*"job_name"*

The name of the job to be called when an order reaches the state specified.

This attribute should not be specified for the end state.

Example:

```
<job_chain_node state="1" job="my_job"/>

<job_chain_node state="2" job="../job_in_parent_folder"/>
```

`next_state`=*"string"*

An order is given the next state when the `spooler_process()` returns `return true` for the order.

The default setting is the `state=` attribute of the following `<job_chain_node>`.

`error_state`=*"string"*

When `return false` is returned by a job's `spooler_process()` method, then the order state is changed to error.

`delay`=*"seconds"* (Initial value:0)

Delays an order before handing it over to a job.

`on_error`=*"suspend|setback"*

After an order processing step, which the order designates as containing an error, the JobScheduler normally allocates the order the `error_state` state. The `on_error` attribute can, however, be used to define another behaviour.

`on_error="suspend"` has the same effect as `Order.suspended` `=true`: the JobScheduler leaves the order in its current state, however, the next processing step is not carried out and the order is suspended.

`on_error="setback"` has the same effect as `Order.setback()` : the JobScheduler leaves the order in its current state, the next processing step is not carried out and the order is treated in the same way as `<delay_order_after_setback>` .

XML Element `<job_chain_node.end>`

```
<job_chain_node.end
    state                         = "string"                        End state
> </job_chain_node.end>
```

Sets a chain of job chains in the end state.

Parent Elements

<job_chain> - Job chain

Attributes

`state`=*"string"* End state

The state valid for this job chain node.

XML Element `<job_chain_node.job_chain>`

```
<job_chain_node.job_chain
    state                         = "string"
    job_chain                     = "job_chain_name"
    next_state                    = "string"
    error_state                   = "string"
> </job_chain_node.job_chain>
```

Adds a new job chain node to a chain of job chains - i.e. in nested job chains.

Job chains which contain nested job chains cannot be nested in other job chains.

Job chains grouped together through nesting build a common *Order_id_space*, in which the uniqueness of an order identifier is tested when the order is submitted.

The JobScheduler does not accept the submission of an order in a job chain, when the order ID has already been allocated in that Order_id_space.

When an order within an Order_id_space is replaced in one job chain in the ID space, then to will also be replaced in other job chains in the same Order_id_space.

Parent Elements

<job_chain> - Job chain

Attributes

`state`=*"string"*

The state valid for this job chain node.

`job_chain`=*"job_chain_name"*

The job chain to which the order is to be handed over when it reaches this state.

`next_state`=*"string"*

`spooler_process()` with `return true` from the last completed job, sets an order in this state.

The default value is the value of the `state=` attribute of the next job chain node.

`error_state`=*"string"*

`spooler_process()` with `return false` from the last completed job, sets an order in this error state..

XML Element `<job_chains>`

```
<job_chains >
    job_chain                      Job chain
</job_chains>
```

Example:
See `<job_chain>` (page 47).

Behavior with `<base>`

Supplements the `<job_chains>` element in the corresponding node of the basic XML configuration . Attributes specified here have precedence over those entered in the basic XML configuration.

Parent Elements

<config> - Configuration

XML Element `<jobs>`

```
<jobs >
    job                            Definition of jobs
</jobs>
```

Behavior with `<base>`

Supplements the `<jobs>` element in the corresponding node of the basic XML configuration . Attributes specified here have precedence over those entered in the basic XML configuration.

Parent Elements

<config> - Configuration

XML Element `<lock>`

`<lock`		
`name`	= *"name"*	The lock name
`max_non_exclusive`	= *"integer"*	Restricting Non-Exclusive Use
`> </lock>`		

A lock can stop two tasks from running at the same time.

See `<lock.use>` (page 54) for information about the use of locks.

See also `<lock.remove>` (page 192)

Example:

```
<locks>
    <lock name="switching_database"/>
    <lock name="only_three_tasks" max_non_exclusive="3"/>
</locks>
```

Behavior with `<base>`

Supplements the `<lock>` element in the corresponding node of the basic XML configuration . Attributes specified here have precedence over those entered in the basic XML configuration.

Parent Elements

<locks> - Declaration of locks

Attributes

`name`=*"name"* The lock name

`max_non_exclusive`=*"integer"* Restricting Non-Exclusive Use

The default setting is unlimited - which means that with `<lock.use exclusive="no">` an unlimited number of non-exclusive tasks can be started (but only one exclusive).

Messages

[ERROR]	SCHEDULER-887	More lock holders than new max_non_exclusive=: holders

XML Element `<lock.use>`

`<lock.use`		
`lock`	= *"name"*	The name of the lock
`exclusive`	= *"yes|no"*	The lock can be made exclusive or non-exclusive
`> </lock.use>`		

See also `<locks>` (page 56) and `<lock>` (page 54).

Example:

```
<locks>
    <lock name="my_file"/>
</lock>

<job name="my_file_reader" tasks="3">
    <lock.use lock="my_file" exclusive="no"/>
    …
</job>

<job name="my_other_file_reader">
    <lock.use lock="my_file" exclusive="no"/>
    …
</job>

<job name="my_file_writer">
    <lock.use lock="my_file"/>
    …
</job>
```

The `my_file_reader` and `my_other_file_reads` jobs do not use the locks exclusively and can run at the same time as other jobs.

The `my_file_writer` job has an exclusive lock and can only run when no other job is running. This job can change the data set and can be used with the certainty that no other jobs will attempt to read the data while this job is running.

See also `Task.try_hold_lock()` .

Behavior with `<base>`

Supplements the `<lock.use>` element in the corresponding node of the basic XML configuration . Attributes specified here have precedence over those entered in the basic XML configuration.

Parent Elements

<job> - Definition of jobs

Attributes

`lock`=*"name"* The name of the lock

The lock itself must have been declared using `<lock>` .

`exclusive`=*"yes|no"* (Initial value:yes) The lock can be made exclusive or non-exclusive

`exclusive="yes"` is the default setting. This means that the lock is made exclusive and that only one task can acquire the lock and start a task. All other jobs with the same lock are added to a queue. Jobs with `exclusive="yes"` are started before jobs with `exclusive="no"`.

`exclusive="no"` only blocks exclusive use (`exclusive="yes")` . A task with `exclusive="no"` only exclude tasks with `exclusive="yes"`. The total number of tasks with `exclusive="no"` can be limited using `<lock max_non_exclusive="…">` .

XML Element `<locks>`

```
<locks >
    lock                        Lock Declaration
</locks>
```

Behavior with `<base>`

Supplements the `<locks>` element in the corresponding node of the basic XML configuration . Attributes specified here have precedence over those entered in the basic XML configuration.

Parent Elements

<config> - Configuration

XML Element `<monitor>`

```
<monitor
    name                          = "Name"
    ordering                      = "Number"
>
    script                      Program code
</monitor>
```

A monitor makes functions available which can be called before and after a class and before and after `spooler_process()` .

A monitor can start the task or stop the execution of `spooler_process()` .

See the `Monitor_impl` superclass whose methods can be implemented by a monitor.

Example:

```
<monitor>

    <script java_class="spooler_job.Java_monitor"><![CDATA[
        package spooler_job;
        import sos.spooler.*;

        public class Java_monitor  extends sos.spooler.Monitor_impl
        {
            public boolean spooler_task_before()  throws Exception
            {
                spooler_log.info( "SPOOLER_TASK_BEFORE()" );
                return true;
            }

            public void spooler_task_after()  throws Exception
            {
                spooler_log.info( "SPOOLER_TASK_AFTER()" );
            }

            public boolean spooler_process_before()  throws Exception
            {
                spooler_log.info( "SPOOLER_PROCESS_BEFORE()" );
                return true;
            }

            public boolean spooler_process_after( boolean spooler_process_result )
throws Exception
            {
                spooler_log.info( "SPOOLER_PROCESS_AFTER(" + spooler_process_result +
")" );
                return spooler_process_result;
            }
        }
    ]]></script>
</monitor>
```

Behavior with `<base>`

This element may not be specified here when it has already been specified in the basic XML configuration.

Parent Elements

<job> - Definition of jobs

Attributes

`name`=*"Name"*

The name of a monitor is an unique identifier for a monitor. More than one monitor can be specified, as long as they have unique names.

The monitors are started in ascending sequence as specified using the `ordering` attribute. The monitor-methods `spooler_process_after()` and `spooler_task_after()` are called in descending sequence.

`ordering`=*"Number"*

Several monitors, if required implemented using different languages, can be specified.

The monitors are started in the order specified in the `ordering` attribute. `spooler_process_after()` and `spooler_task_after()` are called in reverse order.

XML Element `<month>`

```
<month
    month                       = "month"
>
    period                     Operating period
    monthdays                  Operating periods on particular days of the month
    ultimos                    Ultimos - Operating Periods for Particular Days of the Month -
                               Counted from the End of the Month
    weekdays                   Operating Periods for Weekdays
</month>
```

Sets the periods for a particular day of the month.

In contrast to other elements, `<month>` does not take over the attributes from `<run_time>` or the default `<period>` settings.

When `<month>` is set, then values of `<weekdays>`, `<monthdays>` or `<ultimos>` directly set under `<run_time>` do not apply.

Parent Elements

<run_time> - The Job Run Time

Attributes

`month`=*"month"*

One of more names of months, seperated by empty spaces (" "): `"january"`, `"february"`, `"march"`, `"april"`, `"may"`, `"june"`, `"july"`, `"august"`, `"september"`, `"october"`, `"november"`, `"december"`.

XML Element `<monthdays>`

```
<monthdays >
    day                        Periods for Particular Days
    weekday                    Periods for a Particular Week Day
</monthdays>
```

Sets the operating period for a particular day of the month.

Example:

```
<monthdays>
    <day day="1">
        <period begin="10:00" end="12:00"/>
    </day>
    <day day="2">
        <period begin="08:00" end="12:00"/>
        <period begin="15:00" end="18:00"/>
    </day>
    <weekday day="monday" which="1">
        <period single_start="02:00"/>
    </weekday>
</day>
```

The example defines the operating periods as being the first day of the month, from 10:00 to 12:00 and the second day of the month from 08:00 to 12:00 and 15:00 to 18:00. In addition, the job should be run on the first Monday of the month at 02:00.

Example:

```
<monthdays>
    <day day="5">
        <period single_start="16:00"/>
    </day>
</monthdays>
```

This example starts a job on the 5th of each month at 16 Uhr.

Behavior with `<base>`

Supplements the `<monthdays>` element in the corresponding node of the basic XML configuration . Attributes specified here have precedence over those entered in the basic XML configuration.

Parent Elements

<run_time> - The Job Run Time

XML Element `<on_return_code>`

```
<on_return_code
    return_code                    = "integer"            The return code from the job task.
>
    to_state                  To State
    add_order-node_order_pl
    ugin
</on_return_code>
```

Specifies the behavior to be carried out at a node when the node job task exits with a specific return code.

The `<on_return_code>` element can be used to specify the following operations:

- Proceed to next state:

`<to_state>` specifies the state that is to be set for the current order if the task for the job attached to the current node exits with the return code specified.

- Add order for a job chain:

  <add_order> specifies an order that is to be started if the job at the current node exits with the return code specified.

Return code specification Syntax

The `<on_return_code>` element can be used to specify the following operations:

- Individual values, separated by a blank space:

  ```
  <on_return_code return_code="1 2 3">
  ```
- Value ranges:

  ```
  <on_return_code return_code="1..3 7..9">
  ```

  "Less than" (`"..3"`) and "Greater than" (`"7.."`) are not allowed.
- Individual values and ranges can be combined:

  ```
  <on_return_code return_code="1 7..9">
  ```

Example

Using a range of return code values with `<to_state>`:

Example:

```
<on_return_code return_code="1..5">
    <to_state  state="6"/>
</on_return_code>
```

Example

Example with add_order and a single return code value:

Example:

```
<on_return_code  return_code="0">
    <add_order  xmlns="https://jobscheduler-plugins.sos-berlin.com/NodeOrderPlugin"
job_chain="Chain_B">
        <params>
            <param  name="parameterJob3"  value="Job3-RC-0"/>
        </params>
    </add_order>
</on_return_code>
```

Comments

- The `NodeOrderPlugin` listed in the above example has to be installed and specified in the `scheduler.xml` file. See the <add_order> element.
- Return codes can only be specified once within an `<on_return_code>` element.
- Return codes may not overlap within an `<on_return_code>` element.
- Behavior with non-zero `return_code` values depends on the behavior specified in the `<on_return_code>` child element(s):
  - With a `<to_state>` element:
    - An error will be logged for the job task.
    - Processing of the current order will continue from the state specified for `<to_state>` element.

    - The order will end with the *success* state if no further errors occur during its processing.
- With an `<add_order>` element:
    - An error will be logged for the job task.
    - The order specified in the `<add_order>` element will be generated and started immediately.

      Note that the execution of an `<add_order>` element is not logged in the log file of the current current (i.e. originating) order.
    - The current order will end with the *error* state. Processing of the order will not be continued.
- With both `<to_state>` and `<add_order>` elements:
    - An error will be logged for the job task.
    - `<to_state>` elements will be executed before any `<add_order>` elements, regardless of the order in which the `<to_state>` and `<add_order>` elements are specified in the `<on_return_code>` element.

      This moves the current order to a different state and prevents any `<add_order>` elements being executed.
    - The order will end with the *success* state if no further errors occur during its processing.
    - **Caution**: There will be no mention in the log file of the current order that any `<add_order>` elements were not executed.

Usage notes Proceed to next state (`<to_state>`)

- Specification of `<to_state>` for a return code other than 0 (zero) will cause an error to be logged for the job. However, the `<to_state>` element can be used to continue processing of the current job chain from a desired state under the same order ID and with the same order parameters. In this case, the error will still be logged for the job ending in error but the order itself will end with the success state.

  If `<to_state>` is not specified JobScheduler will revert to its default behavior and the order will proceed to the `error_state`.
- **Example:**

```
<job_chain_node  state="3" job="Job_3" next_state="4" error_state="Error">
    <on_return_codes >
        ...
        <on_return_code  return_code="1">
            <add_order
xmlns="https://jobscheduler-plugins.sos-berlin.com/NodeOrderPlugin"
job_chain="Chain_B"/>
            <to_state  state="6"/>
        </on_return_code>
        ...
    </on_return_codes>
</job_chain_node>
```

Add order for a job chain (`<add_order>`)

- Absolute and relative paths can be specified for job chains. Relative paths are based on the location of the current job chain.
- The job chain path of the new order can be used relative to the current job chain.

  The path should start with "./".

•

Example:

```
<job_chain_node  state="3" job="Job_3" next_state="4" error_state="Error">
    <on_return_codes >
        ...
        <on_return_code  return_code="2">
            <add_order
xmlns="https://jobscheduler-plugins.sos-berlin.com/NodeOrderPlugin"
job_chain="Chain_B"/>
            <add_order
xmlns="https://jobscheduler-plugins.sos-berlin.com/NodeOrderPlugin"
job_chain="Chain_C"/>
        </on_return_code>
        ...
    </on_return_codes>
</job_chain_node>
```

- Add order can be specificied together with proceed to next state using the same return code at the same node.

  See the example above under "Proceed to next state".

Further information

- A description of a working example together with a downloadable configuration file is avaliable from the:

  How to configure workflow control by return code handling article.

Behavior with `<base>`

Supplements the `<on_return_code>` element in the corresponding node of the basic XML configuration with the attribute `name=` . Attributes specified here have precedence over those entered in the basic XML configuration.

Parent Elements

<on_return_codes> - On Return Codes

Attributes

`return_code`=*"integer"* The return code from the job task.

This attribute specifies a return code value or range of values.
The operation specified as a child node of the `on_return_code` element will be carried out if the exit code returned by a task executed by the job at the current node matches this value.

Messages

XML Element `<on_return_codes>`

```
<on_return_codes >
    on_return_code            On Return Code
</on_return_codes>
```

The `<on_return_codes>` element functions as a container for one or more `<on_return_code>` elements. These in turn specify an behavior to be carried out if a job task ends with a return code that matches one of the return codes specified in the `<on_return_code>` element's `return_code` attribute.

An example showing use of `<on_return_codes>` can be found in the `<on_return_code>` element page.

Behavior with `<base>`

Supplements the `<on_return_codes>` element in the corresponding node of the basic XML configuration . Attributes specified here have precedence over those entered in the basic XML configuration.

Parent Elements

<job_chain_node> - Job Chain Nodes

XML Element `<param>`

```
<param
    name                    = ""                    Unique Names
    value                   = ""
> </param>
```

See `<params>` (page 66).

Defines the individual parameters for the JobScheduler, a Job or an Order. In general all parameters are available with the API calls `Spooler.variables()`, `Task.params()` respectively `Order.payload()` or in shell Jobs as environment variable (with leading **SCHEDULER_PARAM_**)

Correspondent parameter in the JobScheduler configuration, at the Job and at the order are valid in the following sequence:
- order
- job
- scheduler

The use of individual parameters beginning with **scheduler.** is not recommended as this name space is reserved for the JobScheduler configuration settings.

The parameters can be overwritten and extended during run time.

See also `Variable_set` class.
JobScheduler Parameters

The following parameters can be used to configure the JobScheduler:

- **SCHEDULER_VARIABLE_NAME_PREFIX**
- **scheduler.max_kbyte_of_db_log_entry**
- **scheduler.order.keep_order_content_on_reschedule**
- **scheduler.order.distributed.balanced**
- **scheduler.agent.keep_alive**

  The JobScheduler Master and Classic Agent can be configured to prevent connections from timing out by adding a `scheduler.agent.keep_alive` parameter to the `<params>` section of the Master's scheduler.xml file. This file is located in the `$SCHEDULER_DATA/config` folder, where `$SCHEDULER_DATA` is the directory used for JobScheduler's configuration and log files.

Example:

```
<params>
   <param name="scheduler.agent.keep_alive" value="300" />
</params>
```

- The value attribute sets the interval in seconds between keep-alive packets.

  A duration lower than 30s will be silently replaced by 30s.
- Keep-alive packets will not be sent if the parameter is not set or if the value attribute is empty.
- The keep-alive parameter will be forwarded to the Agent along with other task configuration parameters for use when the Agent initiates a connection.
- Keep-alive packets will be sent across the network by the JobScheduler (either Master or Agent) that initiates a task.

JobScheduler Master

- The Master sends keep-alive packets to Classic Agents (up to and including Classic Agent release 1.9) via TCP connections.
- The Master log will show a `SCHEDULER-711` message at the info level stating that a keep-alive packet has been successfully sent.

Connection Keep-Alive for Master and Agent

- See the Connection Keep-Alive for Master and Agent article for more detailed information.

Delimitation

- Keep-alive packets are not created if Remote File Watching is performed by the Agent.
- Keep-alive packets are only sent for running jobs.

Important

- The JobScheduler Universal Agent (available with JobScheduler 1.10 and later) does not use keep-alive packets. Instead, the Universal Agent sends so-called heartbeats using a secure HTTP connection. See the `<remote_scheduler>` (page 71) element for information about the configuration of heartbeats.

Job Parameters

Job parameters can be called using the `Task.params()` method.

Order Parameters

Order parameters can be called using the `Order.params()` method.

Behavior with `<base>`

Supplements the `<param>` element in the corresponding node of the basic XML configuration with the attribute `name=` . Attributes specified here have precedence over those entered in the basic XML configuration.

Parent Elements

<params> - Parameters

Attributes

`name`="" Unique Names

`value`=""

Environment variables (e.g. `$HOME`) are replaced by this attribute (see Settings which Allow Environment Variables to be Called (page 106)).

XML Element `<param>`

```
<param
    name                    = ""                    Unique Names
    value                   = ""
> </param>
```

See params.

Defines the individual parameters for an order when an <add_order> element is used in the correct namespace.

Functions as a container for the `<param>` elements which specify the parameters for an order.

Note that any order parameters that were set for the originating order will also be forwarded to the new order.

Parameters that were set for the originating order will be overwritten by parameters set using the `<param>` element that have the same name.

Correspondent parameter in the JobScheduler configuration, at the Job and at the order are valid in the following sequence:

- order
- job
- scheduler

The use of individual parameters beginning with **scheduler.** is not recommended as this name space is reserved for the JobScheduler configuration settings.

The parameters can be overwritten and extended during run time.

- **SCHEDULER_VARIABLE_NAME_PREFIX**
- **scheduler.max_kbyte_of_db_log_entry**
- **scheduler.order.keep_order_content_on_reschedule**
- **scheduler.order.distributed.balanced**

Behavior with `<base>`

Supplements the `<param>` element in the corresponding node of the basic XML configuration with the attribute `name=` . Attributes specified here have precedence over those entered in the basic XML configuration.

Parent Elements

<params-node_order_plugin> -

Attributes

`name`="" Unique Names

`value`=""

Environment variables (e.g. `$HOME`) are replaced by this attribute (see Settings which Allow Environment Variables to be Called (page 106)).

XML Element `<params>`

```
<params >
    param                          Individual Parameters
    copy_params                    Passing Parameters
    include                        Includes text from a file
</params>
```

Specifies the parameters for the JobScheduler, a job or an order. The parameters can be overwritten and extended whilst the JobScheduler is running.

JobScheduler parameters can be called up using the `Spooler.variables()` method.

Job parameters are called using the `Task.params()` method.

The parameters for an order can be called using the `Order.payload()` method.

See also the `Variable_set` and `<sos.spooler.variable_set>` (page 79) classes.

Behavior with `<base>`

Supplements the `<params>` element in the corresponding node of the basic XML configuration . Attributes specified here have precedence over those entered in the basic XML configuration.

Parent Elements

<job> - Definition of jobs

<add_order> - Add an order

<config> - Configuration

<modify_order> -

<payload> -

<queued_task> -

<web_service> - Web Service

XML Element `<params>`

```
<params >
    param-node_order_plugin
</params>
```

This element is derived from a different namespace `https://www.sos-berlin.com/repository/scheduler/1.9/scheduler.xsd` that includes support for the NodeOrderPlugin.

This element can only be used when its parent <add_order> element is used in the correct namespace.

The `<params>` element functions as a container for `<param>` elements used to specify the parameters for an order.

See also <param>.

Behavior with `<base>`

Supplements the `<params>` element in the corresponding node of the basic XML configuration . Attributes specified here have precedence over those entered in the basic XML configuration.

Parent Elements

<add_order-node_order_plugin> -

XML Element `<period>`

```
<period
    begin                    = "hh:mm[:ss]"
    end                      = "hh:mm[:ss]"
    repeat                   = "hh:mm[:ss] or seconds"
    absolute_repeat          = "hh:mm[:ss] | seconds"
    single_start             = "hh:mm[:ss]"
    let_run                  = "yes_no"
    when_holiday             = ""                        Treatment of Holidays
> </period>
```

An operating period defines when a job may run. This can be a period of time within a day (with the `begin` and `end` attributes) or a start time (with the `single_start` attribute).

Parent Elements

<run_time> - The Job Run Time

Attributes

`begin`=*"hh:mm[:ss]"* (Initial value:00:00)

The start of the operating period for the job.

`end`=*"hh:mm[:ss]"* (Initial value:24:00)

The end of the operating period. Should `let_run="no"` have been set and no further operating period is designated, then the JobScheduler ends all tasks which are running (using `spooler_close()`).

`repeat`=*"hh:mm[:ss] or seconds"*

Should a job not already be running, then it will be started at the start of the operating period. After the job has ended, it will be restarted after the time specified, as far as allowed by the `<run_time>` attribute. This repeat interval can be specified in *hh:mm*, in *hh:mm:ss* or in seconds.

Cannot be combined with the `single_start=` attribute.

The job will not be repeated, if `repeat="0"` (the default value) is set.

`absolute_repeat`=*"hh:mm[:ss] | seconds"*

Similar to `repeat` but allows the `begin` and `end` times to be specified independently of a JobScheduler's operating period.

Starts a job, should it not already be running, at the beginning of a specified time period. Thereafter, the job will be restarted at regular intervals. The job starting times then result from the `begin` time plus a multiple of the `absolute_repeat` interval. This repeat interval can be specified in *hh:mm*, in *hh:mm:ss* or in seconds.

Cannot be combined with the `single_start=` attribute.

`single_start`=*"hh:mm[:ss]"*

The job should start at the time given.

Cannot be used in combination with the `begin=`, `end=` or `repeat=` attributes.

`let_run`=*"yes_no"*

This attribute can only be used for jobs and not for orders. The `let_run="no"` setting should be made for order controlled jobs.

`let_run="yes"` allows the JobScheduler to let a task continue running, even though this is not allowed by the `<run_time>` attribute.

`let_run="no"` causes the JobScheduler to end a task (spooler_close is evoked instead of spooler_process), as soon as the `<run_time>` is no longer valid.

`when_holiday`=*""* Treatment of Holidays

A period landing on a holiday `<holidays>` is usually suppressed. Other settings are however possible.

`when_holiday="suppress"`

The default setting. A period landing on a holiday is suppressed.

`when_holiday="ignore_holiday"`

A period landing on a holiday is not suppressed.

`when_holiday="previous_non_holiday"`

When a period occurs on a holiday, it will be brought forward to the last preceding non-holiday.

`when_holiday="next_non_holiday"`

When a period occurs on a holiday, it will be postponed to the next non-holiday.

XML Element `<process_class>`

```
<process_class
    spooler_id              = "scheduler_id"
    name                    = "name"
    max_processes           = "number"
    remote_scheduler        = "host:port"          Task execution on remote
                                                   computers
    replace                 = "yes|no"
```

```
>
   remote_schedulers        Remote Schedulers
</process_class>
```

Defines or modifies a process class.

See also <process_class.remove> (page 202).

Behavior with <base>

Supplements the <process_class> element in the corresponding node of the basic XML configuration . Attributes specified here have precedence over those entered in the basic XML configuration.

Parent Elements

<process_classes> - Process Classes

Attributes

spooler_id=*"scheduler_id"*

An element having this attribute is only active when the attribute is either:

- empty
- set to the -id= JobScheduler start parameter
- or when the JobScheduler -id option is not specified when starting the JobScheduler.

name=*"name"*

The name of the process class. Should this attribute be missing or empty ("") then the default process class will be changed.

See the process_class= attribute of the <job> (page 42) element.

See the process_class= attribute of the <job_chain> (page 47) element.

max_processes=*"number"* (Initial value:30)

Limits the number of processes.

Some operating systems limit the number of processes which the JobScheduler can start. The number of processes configured here should not exceed the number allowed by the operating system. A value below 64 is usually safe.

For Microsoft Windows systems, the maximum number of processes that are allowed to be executed in parallel is currently 30.

remote_scheduler=*"host:port"* Task execution on remote computers

Specifies the remote computer on which the tasks of this process class are to be executed. This computer is specified using its host name or IP number and TCP port (see <config tcp_port="…"> (page 21)).

The remote computer must allow access with <allowed_host level="all">.

Tasks executed communicate with the controlling JobScheduler via the API. However, the following points should be noted:

- `<include>` within `<script>` is executed by a task process. The file to be included is thereby read by the computer which carries out the task.
- The `Subprocess.timeout` and `Task.add_pid()` methods do not work. The JobScheduler cannot terminate remote subprocesses whose time limits have been exceeded.
- `Log.log_file()` is, as with almost all methods, carried out on the computer on which the JobScheduler is running and thereby accesses the files of its local file system.

Some settings are taken from the remote instead of from the controlling JobScheduler:

- `sos.ini` (section `[java]`, entry `javac= …`)
- `factory.ini` (section `[spooler]`, entry `tmp= …`)
- `<config java_options="…">`
- `<config java_class_path="…">`
- `<config include_path="…">`

Messages

`[warn]`	`SCHEDULER-849`	Timeout is not possible for a subprocess running on a remote host (it cannot be killed), pid=
`[warn]`	`SCHEDULER-850`	After lost connection to remote scheduler, process is going to be killed
`[info]`	`SCHEDULER-848`	Task pid= started for remote scheduler

`replace`=*"yes|no"* (Initial value:yes)

`replace="yes"` replaces the existing process class.

`replace="no"` only changes the attributes which are set by the process class.

XML Element `<process_classes>`

```
<process_classes
    ignore                          = "yes|no"
>
    process_class                 Process class
</process_classes>
```

Behavior with `<base>`

Supplements the `<process_classes>` element in the corresponding node of the basic XML configuration . Attributes specified here have precedence over those entered in the basic XML configuration.

Parent Elements

<config> - Configuration

Attributes

`ignore`=*"yes|no"* (Initial value:no)

`ignore="yes"` allows operation with process classes to be disabled. This means that tasks run as a part of the JobScheduler process, which enables debugging of tasks and orders to be carried out.

XML Element `<remote_scheduler>`

```
<remote_scheduler
   remote_scheduler            = "string"
   http_heartbeat_period       = "number"
   http_heartbeat_timeout      = "number"
> </remote_scheduler>
```

Defines a remote JobScheduler.

Example:

```
<process_class  max_processes="10">
    <remote_schedulers>
        <remote_scheduler remote_scheduler="http://127.0.0.2:5000"
               http_heartbeat_period="10"
               http_heartbeat_timeout="15"/>
    </remote_schedulers>
</process_class>
```

Behavior with `<base>`

Supplements the `<remote_scheduler>` element in the corresponding node of the basic XML configuration . Attributes specified here have precedence over those entered in the basic XML configuration.

Parent Elements

<remote_schedulers> - Remote Schedulers

Attributes

`remote_scheduler`=*"string"*

The URL of the remote scheduler - e.g. `http://127.0.0.2:5000`

`http_heartbeat_period`=*"number"* (Initial value:10)

`http_heartbeat_period` specifies a number of seconds after which the Agent will send a heartbeat signal to the Master if a HTTP operation with the Master is not otherwise performed.

The HTTP Heartbeat Period attribute should be a positive integer and less than the HTTP Heartbeat Timeout.

The HTTP Heartbeat Timeout attribute is only available with JobScheduler Universal Agents (available with JobScheduler 1.10 and later). JobScheduler Classic Agents (version 1.9) use keep-alive packets. See the `<param>` (page 63) element for more information.

`http_heartbeat_timeout`=*"number"* (Initial value:15)

`http_heartbeat_timeout` specifies the number of seconds within which the Agent expects to receive a heartbeat from the Master.

The HTTP Heartbeat Timeout attribute should be an integer and larger than the HTTP Heartbeat Period.

The HTTP Heartbeat Timeout attribute is only available with JobScheduler Universal Agents (available with JobScheduler 1.10 and later). JobScheduler Classic Agents (version 1.9) use keep-alive packets. See the `<param>` (page 63) element for more information.

XML Element `<remote_schedulers>`

```
<remote_schedulers
    ignore                        = "yes|no"
>
    remote_scheduler           Remote Scheduler
</remote_schedulers>
```

Behavior with `<base>`

Supplements the `<remote_schedulers>` element in the corresponding node of the basic XML configuration . Attributes specified here have precedence over those entered in the basic XML configuration.

Parent Elements

<process_class> - Process class

Attributes

`ignore`=*"yes|no"* (Initial value:no)

The remote JobSchedulers specified for a process class.

XML Element `<run_time>`

```
<run_time
    time_zone                     = "text"                    Time zone for a order / job /
                                                              schedule
    schedule                      = ""                        Operating Schedule
    once                          = "yes_no"
    begin                         = "hh:mm[:ss]"
    end                           = "hh:mm[:ss]"
    repeat                        = "hh:mm[:ss] or seconds"
    single_start                  = "hh:mm[:ss]"
    let_run                       = "yes_no"
    when_holiday                  = ""
>
    period                     Operating period
    at                         Start Time
    date                       Operating Times for Particular Days
    weekdays                   Operating Periods for Weekdays
    monthdays                  Operating periods on particular days of the month
    month                      The names of the months
    ultimos                    Ultimos - Operating Periods for Particular Days of the Month -
                               Counted from the End of the Month
    holidays                   Holidays
</run_time>
```

`<run_time>` defines the times at which the JobScheduler allows the tasks of this job to run. This is achieved through the use of operating periods (see `<period>`). At startup the JobScheduler selects the first period which is valid (that is, which has not yet ended). This operating period remains valid until its end. The JobScheduler then selects the next possible period.

An operating period with `single_start` is only valid at one point in time.

Daylight Saving Time

The JobScheduler recognizes the time changes at the beginning and end of daylight saving time. It always uses local time.

A job with a start time between 02:00 and 03:00 on the night at the end of summer time may be started twice.

Shortened Form Using <period> and the begin=, end=, repeat=, let_run= and single_start= Attributes

`<period>` may be used within the `<run_time>` the elements `<date>` , `<weekdays>` , `<monthdays>` and `<ultimos>` , should these possess no `<period>` element of their own.

```
<run_time>
    <period begin="07:00" end="09:00"/>
    <monthdays>
        <day day="1"/>
        <day day="2">
            <period begin="22:00" end="23:00"/>
        </day>
    </monthdays>
</run_time>
```

becomes

```
<run_time>
    <monthdays>
        <day day="1">
            <period begin="07:00" end="09:00"/>
        </day>
        <day day="2">
            <period begin="22:00" end="23:00"/>
        </day>
    </monthdays>
</run_time>
```

Should none of the `<date>` , `<weekdays>` , `<monthdays>` or `<ultimos>` elements be listed, then the `<period>` element is applied for every day of the week.

The `begin=`, `end=`, `repeat=`, `let_run=` and `single_start=` attributes apply when `<period>` is specified, and allow the JobScheduler to create a similar attribute of the same name.

```
<run_time begin="07:00" end="09:00"/>
```

becomes

```
<run_time>
    <period begin="07:00" end="09:00"/>
<run_time>
```

Example:

```
<run_time/>
```

is, because of the default settings for `begin=` and `end=`, the same as

```
<run_time begin="00:00" end="24:00"/>
```

is, because `<run_time>` is empty, the same as

```
<run_time>
    <period begin="00:00" end="24:00"/>
</run_time>
```

is an operating period valid 24 hours every day. The job can always run.

Parent Elements

<job> - Definition of jobs

<add_order> - Add an order

<order> -

Attributes

`time_zone`=*"text"* Time zone for a order / job / schedule

Overwrites the global time zone setting for a JobScheduler object (order, job or schedule).

See `<config time_zone="…">` (page 21).

`schedule`=*""* Operating Schedule

Specifies the `<schedule>` that is to be used.

All other attributes and child elements are ignored.

`once`=*"yes_no"* (Initial value:no)

When `once="yes"` the Scheduler starts a job once after starting itself, in so far as this is allowed by the `<run_time >`. In addition persistent jobs with `once="yes"` are startet instantly if they are added to a Live-Folder while the JobScheduler is running and this is allowed by the `<run_time>`.

`begin`=*"hh:mm[:ss]"* (Initial value:00:00)

Should the `<run_time>` element be empty (i.e. it does not contain a `<period>`), then the JobScheduler will generate an operating period using this setting. This setting is also the default setting for the `<run_time>` child elements (see the shortened form (page 73) above).

`end`=*"hh:mm[:ss]"* (Initial value:24:00)

Should the `<run_time>` element be empty (i.e. it does not contain a `<period>`), then the JobScheduler will generate an operating period using this setting. This setting is also the default setting for the `<run_time>` child elements (see the shortened form (page 73) above).

`repeat`*="hh:mm[:ss] or seconds"*

Should the `<run_time>` element be empty (i.e. it does not contain a `<period>`), then the JobScheduler will generate an operating period using this setting. This setting is also the default setting for the `<run_time>` child elements (see the shortened form (page 73) above).

`single_start`*="hh:mm[:ss]"*

Should the `<run_time>` element be empty (i.e. it does not contain a `<period>`), then the JobScheduler will generate an operating period using this setting. This setting is also the default setting for the `<run_time>` child elements (see the shortened form (page 73) above).

`let_run`*="yes_no"*

This attribute determines whether a running task should be stopped or allowed to carry out further process steps after a `<run_time>` period has ended. The default setting here is that a job does not carry out any further process steps after the period has ended. The job is then ended (let_run="no").

The following applies for order controlled jobs:
When an order controlled job defines a period such as 12:00 - 14:00 and an order is started during this time, the order will be completely carried out. After the order has been completed, the idle_timeout value is used to determine whether the task remains active and open for further orders. It is only after the task remains started and has accepted an order that the value of let_run will be considered.

let_run="yes": the order will be carried out.

let_run="no": the order will only be carried out when its starting time lies within the job `<run_time>` period.

Should the `<run_time>` element be empty (i.e. it does not contain a `<period>`), then the JobScheduler will generate an operating period using this setting. This setting is also the default setting for the `<run_time>` child elements (see the shortened form (page 73) above).

`when_holiday`*=""*

When this element is empty (i.e. does not contain any `<period>`), then the JobScheduler generates a period with this setting. Otherwise, this is the default setting for the child elements (see Shortened Form (page 73) above).

XML Element `<schedule>`

```
<schedule
    name                          = "name"
    substitute                    = "schedule_path"              A schedule for temporary
                                                                 substitution
    valid_from                    = "yyyy-mm-dd HH:MM[:ss]"
    valid_to                      = "yyyy-mm-dd HH:MM[:ss]"
> </schedule>
```

The other elements and attributes that can be specified for `<run_time>` can also be specified here, with the exception of `schedule=` and `time-zone`.

With distributed orders, a change in `<schedule>` only takes effect the next time the order proceeds along the job chain.

Parent Elements

<schedules> - Timetables

Attributes

`name`=*"name"*

The schedule name.

A named schedule can be called up using `<run_time schedule="…">`.

`substitute`=*"schedule_path"* A schedule for temporary substitution

The schedule specified using `substitute="`*`schedule_path`*`"` should be replaced with the temporary schedule defined here.

The substitution period is specified using `valid_from=` and `valid_to=`.

A temporary schedule cannot be directly called using `<run_time schedule="…">`.

Messages

[ERROR]	SCHEDULER-463	schedule: substituted 'schedule' is a substitute
[info]	SCHEDULER-705	Substitute 'schedule' is valid now
[info]	SCHEDULER-706	Standard 'schedule' is valid now

`valid_from`=*"yyyy-mm-dd HH:MM[:ss]"*

If `substitute=` has been specified and `valid_from=` not specified, then the schedule specified will be substituted immediately.

Messages

[ERROR]	SCHEDULER-465	'schedule' overlaps schedule
[ERROR]	SCHEDULER-466	'schedule' is a substitute for another schedule and cannot be used directly

`valid_to`=*"yyyy-mm-dd HH:MM[:ss]"*

If `substitute=` has been specified and `valid_from=` not, then substitution will continue indefinitely.

Messages

[ERROR]	SCHEDULER-464	schedule: valid_from=""> is not before valid_to=""
[ERROR]	SCHEDULER-465	'schedule' overlaps schedule
[ERROR]	SCHEDULER-466	'schedule' is a substitute for another schedule and cannot be used directly

XML Element `<schedules>`

```
<schedules >
    schedule                          Schedule
</schedules>
```

Behavior with <base>

Supplements the `<schedules>` element in the corresponding node of the basic XML configuration . Attributes specified here have precedence over those entered in the basic XML configuration.

Parent Elements

<config> - Configuration

XML Element `<script>`

```
<script
    language                          = "language"
    com_class                         = "com_class_name"
    filename                          = "file_name"
    java_class                        = "java_class_name"
    java_class_path                   = "java_class_path"
>
    include                           Includes text from a file
</script>
```

The program code to be executed is specified here, either direct as text, or indirect as a reference to binary code.

Source code can be included as text in `<script>`. It can be included in `<[ CDATA[` and `]]>`.

Behavior with <base>

Supplements the `<script>` element in the corresponding node of the basic XML configuration . Attributes specified here have precedence over those entered in the basic XML configuration.

The script can be added to a script defined in the a basic configuration.

Parent Elements

<config> - Configuration

<scheduler_script> -

<job> - Definition of jobs

<monitor> - Job Monitor

Attributes

`language`=*"language"*

The language of the program code. Is not used in conjunction with the `com_class`. Case is not important here.

`com_class`=*"com_class_name"*

The name of a COM-Class (Windows only). The COM class can implement the spooler_open(), spooler_process() etc. methods.

`filename`=*"file_name"*

Should the name of the dll which implements the COM class not be registered, then its name can be given here, in conjunction with the `com_class` attribute.

`java_class`=*"java_class_name"*

Should a job be implemented as a Java class, then the class name must be defined using this attribute.

A name specified in the basic configuration can be overwritten here. The next task (running in a separate process) uses a new class.

`java_class_path`=*"java_class_path"*

Allow a job-specific Class-Path.

This java classpath will be prepend to the java classpath customized in the factory.ini

XML Element `<security>`

```
<security
    ignore_unknown_hosts        = "yes_no"
>
    allowed_host               Allowed Host Computers
</security>
```

`<security>` defines the computers and networks which are allowed to send commands per TCP und UDP.

Behavior with `<base>`

Supplements the `<security>` element in the corresponding node of the basic XML configuration . Attributes specified here have precedence over those entered in the basic XML configuration.

Parent Elements

<config> - Configuration

Attributes

`ignore_unknown_hosts`=*"yes_no"* (Initial value:no)

The JobScheduler ignores unrecognized or unresolved host names when `ignore_unknown_hosts="yes"` in `<allowed_host>`.

This attribute only affects the `<allowed_host>` defined here and not the basic configuration.

XML Element `<service_request>`

```
<service_request
    url                             = "url"
>
    web_service                    Web Service
    content                        Content of a Web Service Request
</service_request>
```

Example:

```
<service_request url="http://host.company.com:80/web_service">
    <content>
        <my_request>
            …
        </my_request>
    </content>
</service_request>
```

`<service_request>` tritt an zwei Stellen auf:

- Als Eingabe einer XLST-Transformation mit `<web_service request_xslt_stylesheet="…">`.
- Als Ergebnis einer XLST-Transformation mit `<web_service forward_xslt_stylesheet="…">`.

Attributes

`url` =*"url"*

The Web Service URL

XML Element `<service_response>`

```
<service_response >
    content                        Content of a Web Service Request
</service_response>
```

XML Element `<sos.spooler.variable_set>`

```
<sos.spooler.variable_set >
    variable                       A Variable
</sos.spooler.variable_set>
```

Example:

```
<sos.spooler.variable_set>
    <variable name="param1" value="11111">
    <variable name="param2" value="2222">
</sos.spooler.variable_set>
```

`<sos.spooler.variable_set>` is used when saving the `Order.payload` in the database.

See also `<params>` (page 66).

XML Element `<spooler>`

```
<spooler >
   config                      Configuration
</spooler>
```

XML Element `<start_job>`

```
<start_job
   job                           = "job_name"
   name                          = "name"
   after                         = "number"
   at                            = "yyyy-mm-dd hh:mm:ss | now |
                                 period"
   force                         = "yes|no"
   web_service                   = "name"
>
   environment                 Environment Variables
   params                      Parameters
</start_job>
```

Example:

```
    <start_job job="my_job" at="now">

    <params>
        <param name="number" value="100"/>

    </params>
</start_job>
```

Parent Elements

<commands> - XML Commands

Attributes

`job`=*"job_name"*

The job name.

`name`=*"name"*

A task can be given a name here.

`after`=*"number"*

A delay - the number of seconds after which a task should be started.

`at`=*"yyyy-mm-dd hh:mm:ss* | *now* | *period"* (Initial value:now)

The time at which a task is to be started. `<run_time>` is deactivated.

Relative times - `"now"`, `"now + HH:MM[:SS]"` and `"now + SECONDS"` - are allowed.

`at="period"` allows a job to start when allowed by `<run_time>` (that is in the current or next period).

`force`=*"yes|no"* (Initial value:yes)

`force="no"`:

- A job that with the "stopped" state will remain in this state.
- If a start time has been specified with either `<run_time>` or `<schedule>` and this does not allow a job to start then the JobScheduler will postpone the start to the next period.
- This means that `at="now"` has the same effect as `at="period"`.

`force="yes"`:

- This means that `at="now"` has the same effect as `at="period"`. A job that with the "stopped" state will be immediately started.
- A job with a start time specified using `at=` will be started at this time, regardless of whether or not a run-time period has been specified using `<run_time>` or `<schedule>`.

`web_service`=*"name"*

After a task has been executed, it is transformed with a style sheet and handed over to a Web Service.

See `<web_service>` (page 83).

XML Element `<start_when_directory_changed>`

```
<start_when_directory_changed
    directory                   = "path"
    regex                       = "regex"
> </start_when_directory_changed>
```

`<start_when_directory_changed directory="directory" regex="regex"/>` functions the same as a programmed start
`start_when_directory_changed( "directory", "regex" ).`

A task is started should a monitored directory be removed after the JobScheduler has been started. At the same time the current monitoring job will be stopped. No further tasks will be started (because the job has been stopped).

The job should be regularly restarted using `<run_time repeat="…">` and `<delay_after_error>` used.

See Directory Monitoring with File Orders (page 165) and `Job.start_when_directory_changed()`.

Parent Elements

<job> - Definition of jobs

Attributes

`directory`=*"path"*

A change in the directory (the addition or deletion of a file in the directory) leads to the start of a task. This also occurs when the directory being monitored itself is deleted.

Environment variables (e.g. `$HOME`) are replaced by this attribute (see Settings which Allow Environment Variables to be Called (page 106)).

`regex`=*"regex"*

Only file names which correspond with this regular expression are noted.

Deleting a file whose name corresponds with the regular expression does not cause the job to be started.

XML Element `<to_state>`

```
<to_state
    state                          = "integer"
> </to_state>
```

`<to_state>` specifies the state within the current job chain that is to be moved to next. This element will be triggered if a job returns the return code specified in the `return_code` attribute of a parent `<on_return_code>` element.

The function of the `<to_state>` element - in particular when used together with `<add_order>` in an `<on_return_code>` element is described on the `<on_return_code>` element page.

Parent Elements

<on_return_code> - On Return Code

Attributes

`state`=*"integer"*

The state within the current job chain that is to be moved to next.

XML Element `<ultimos>`

```
<ultimos >
    day                          Periods for Particular Days
</ultimos>
```

Sets the operating period for a particular day of the month - counted from the end of the month.

Example:

```
<ultimos>
    <day day="0">
        <period begin="10:00" end="12:00"/>
    </day>
    <day day="1">
        <period begin="08:00" end="12:00"/>
        <period begin="15:00" end="18:00"/>
    </day>
</ultimos>
```

Defines the operating periods as being the last day of the month, from 10:00 to 12:00 and the second last day of the month from 08:00 to 12:00 and 15:00 to 18:00.

Behavior with <base>

Supplements the `<ultimos>` element in the corresponding node of the basic XML configuration . Attributes specified here have precedence over those entered in the basic XML configuration.

Parent Elements

<run_time> - The Job Run Time

XML Element `<variable>`

```
<variable
    name                            = ""
    value                           = ""
> </variable>
```

Behavior with <base>

Supplements the `<variable>` element in the corresponding node of the basic XML configuration with the attribute `name=` . Attributes specified here have precedence over those entered in the basic XML configuration.

Parent Elements

<environment> - Environment Variables

<sos.spooler.variable_set> - Variable Set

Attributes

`name=""`

The name of the environment variable. The use of upper / lower case is not significant on Windows machines.

The same name can be used repeatedly, should it be desirable to extend the value (see the example with `PATH`).

`value=""`

The value of an environment variable which has previously been set using `<variable>` can be returned using '$'.

XML Element `<web_service>`

```
<web_service
    name                        = "name"        The service name in the JobScheduler
    url_path                    = "url_path"    The URL path used to reach a service
    job_chain                   = "job_chain"   The job chain executing a service
    timeout                     = "seconds"     Waiting time
    request_xslt_stylesheet     = "path"        Transforms a request into a
                                                JobScheduler command
    response_xslt_stylesheet    = "path"        Transforms a reply to a command into a
                                                Web Service response
    forward_xslt_stylesheet     = "path"        Forwarding after an order or task has
                                                been completed
    debug                              = "yes|no"
>
    params                         Parameters
</web_service>
```

Example:

```
<web_service
    name      = "my_web_service"
    url_path  = "/webservice"
    job_chain = "my_service_job_chain"
/>
```

Example:

```
<web_service
    name                         = "my_web_service"
    url_path                     = "/webservice"
    request_xslt_stylesheet  = "$SCHEDULER_CONFIG/web_service_request.xsl"
    response_xslt_stylesheet = "$SCHEDULER_CONFIG/web_service_response.xsl"
    forward_xslt_stylesheet  = "$SCHEDULER_CONFIG/web_service_forward.xsl"
/>
```

Web services can be set up to create orders or tasks and immediately answer requests. The results of these orders or tasks can be forwarded to another Web Service.

The TCP port for the HTTP-Server is specified using `<config tcp_port="…">`.

The Procedure with job_chain

The JobScheduler creates an order in response to a HTTP-POST to the Web Service URL. It then adds this order to the job chain queue. Jobs can access the HTTP data using `Order.web_service_operation`. The HTTP query is answered using the `Web_service_response.send()` method.

The Procedure with request_xslt_stylesheet

In this case a HTTP-POST to the Web Service URL initiates the following steps:

Error Handling

When it is not possible to forward an XML document using POST, then a "404 Bad Request" HTTP error code will be generated.
An error in the transformation causes a HTTP "500 Internal Server Error" error code.

Forwarding Request Results

Orders and tasks can be allocated to a Web Service. In this case, the `<add_order>` and `<start_job>` commands are given the new `web_service="`*`service_name`*`"` attribute.

Such orders and tasks are transformed by `forward_xslt_stylesheet` and forwarded as `<order>` or `<task>`, once they have been completed and only if the Web Service has been allocated a `forward_xslt_stylesheet`:

```
<order service="service_name" last_job="job_name" …>
    <payload>
        <params>
            …
        </params>
    </payload>
    <log last_error="…" last_warning="…"/>
    …
</order>
```

and

```
<task job="…" …>
    <log last_error="…" last_warning="…"/>
    …
</task>
```

The result of the transformation is a `<service_request>` :

```
<service_request url="url">
    <content>…data…</content>
</service_request>
```

The scheduler_service_forwarding Job Chain

The JobScheduler packs the `<service_request>` in the payload of a new order which it then hands over to the predefined `scheduler_service_forwarding` job chain.

The scheduler_service_forwarder Job

The only job in the `scheduler_service_forwarder` job chain. Uses a URL to make a HTTP connection, transfers data using POST and waits for the answer, which it then logs.

The properties of this job can be defined in the configuration file in the same way as if the job were published using `<base>` . For example:

Example:

```
<job name="scheduler_service_forwarder">
    <delay_order_after_setback setback_count="1" delay="00:01"/>
    <delay_order_after_setback setback_count="2" delay="01:00"/>
    <delay_order_after_setback setback_count="3" delay="24:00"/>
    <run_time>
        <period begin="07:00" end="17:00"/>
    </run_time>
</job>
```

The job has been implemented in Java (see also `<web_service_forward_xslt_stylesheet="…">` (page 83)).

Behavior with `<base>`

Supplements the `<web_service>` element in the corresponding node of the basic XML configuration . Attributes specified here have precedence over those entered in the basic XML configuration.

Parent Elements

<http_server> - HTTP server

Attributes

`name`=*"name"* The service name in the JobScheduler `url_path`=*"url_path"* The URL path used to reach a service

This path should start with a forward slash (/).

Environment variables (e.g. `$HOME`) are replaced by this attribute (see Settings which Allow Environment Variables to be Called (page 106)).

`job_chain`=*"job_chain"* The job chain executing a service

A Web Service request creates a new order which is added to the job chain.

Cannot be used with the `request_xslt_stylesheet` and `response_xslt_stylesheet` attributes.

See `Order.web_service_operation`.

`timeout`=*"seconds"* Waiting time

When an order is not forwarded to the first job within the allocated waiting time, then the JobScheduler will reject HTTP call with "`504 Gateway Timeout`" and cancel the order with a `SCHEDULER-290` message.

`request_xslt_stylesheet`=*"path"* Transforms a request into a JobScheduler command

The path to the XSLT style sheet with which the XML document forwarded using HTTP-POST is transformed into a JobScheduler command.

Cannot be used with the `job_chain` attribute.

Environment variables (e.g. `$HOME`) are replaced by this attribute (see Settings which Allow Environment Variables to be Called (page 106)).

`response_xslt_stylesheet`=*"path"* Transforms a reply to a command into a Web Service response

The path to the XSLT style sheet used to transform the XML result of the JobScheduler command into an XML document. This style sheet is used to answer a Web Service query.

Cannot be used with the `job_chain` attribute.

Environment variables (e.g. `$HOME`) are replaced by this attribute (see Settings which Allow Environment Variables to be Called (page 106)).

`forward_xslt_stylesheet`=*"path"* Forwarding after an order or task has been completed

Valid for an order which has been started with `<add_order web_service="…">` or a task with `<start_job web_service="…">` and `forward_xslt_stylesheet` has been specified. Transforms the order or task with the style sheet to a `<service_request>`, which in turn calls another Web Service.

Cannot be used with the `job_chain` attribute.

Requires Java and the class `xercesImpl.jar` class archive in the `CLASS_PATH` (see `<config java_class_path="…/xercesImpl.jar">`).

Environment variables (e.g. `$HOME`) are replaced by this attribute (see Settings which Allow Environment Variables to be Called (page 106)).

`debug`=*"yes|no"* (Initial value:no)

Only when `request_xslt_stylesheet` is specified:

`debug="yes"` - the JobScheduler allows internally created XML documents to be saved in the directory specified with `-log-dir` :

The JobScheduler does not clean up the files.

XML Element `<weekday>`

```
<weekday
    day                           = "weekday"
    which                         = "integer"
>
    period                      Operating period
</weekday>
```

Sets the run time for a particular day of the month.

Example:

```
<monthdays>
    <day day="1">
        <period begin="10:00" end="12:00"/>
    </day>
    <day day="2">
        <period begin="08:00" end="12:00"/>
        <period begin="15:00" end="18:00"/>-->
    </day>
</monthdays>
```

Sets the run time for the first day of the month from 10:00 to 12:00 and for second day from 08:00 to 12:00 and from 15:00 to 18:00.

Example:

```
<monthdays>
    <day day="5">
        <period single_start="16:00"/>
    </day>
</monthdays>
```

Start a job on the 5th of a month at 16:00.

Behavior with `<base>`

Supplements the `<weekday>` element in the corresponding node of the basic XML configuration . Attributes specified here have precedence over those entered in the basic XML configuration.

Parent Elements

<monthdays> - Operating periods on particular days of the month

Attributes

`day`=*"weekday"*

The name of the week day: `"monday"`, `"tuesday"`, `"wednesday"`, `"thursday"`, `"friday"`, `"saturday"` and `"sunday"`.

More than one day can be specified by leaving an empty space between the names of the days.

`which`=*"integer"*

`which="1"` bis `which="4"`: From the first to the fourth week days in a month.

`which="-1"` bis `which="-4"`: From the fourth-last to the last week days in a month.

XML Element `<weekdays>`

```
<weekdays >
    day                              Periods for Particular Days
</weekdays>
```

Sets the operating period for particular weekdays.

Example:

```
<weekdays>
    <day day="1">
        <period begin="10:00" end="12:00"/>
    </day>
    <day day="2">
        <period begin="08:00" end="12:00"/>
        <period begin="15:00" end="18:00"/>
    </day>
</weekdays>
```

Defines Mondays 10:00 to 12:00 and Tuesdays 08:00 to 12:00 and 15:00 to 18:00 as being operating periods.

Behavior with `<base>`

Supplements the `<weekdays>` element in the corresponding node of the basic XML configuration . Attributes specified here have precedence over those entered in the basic XML configuration.

Parent Elements

<run_time> - The Job Run Time

1.2 Configuration Using Hot Folders

Jobs, job chains, permanent orders, process classes and locks (referred to as objects in the following section) can be stored in individual files, which the JobScheduler then automatically processes after any changes are made to these objects. *Hot Folders* are directories which are monitored by the JobScheduler. It creates, modifies and deletes jobs, job chains and other objects from files which are added to, modified in or deleted from these directories.

1.2.1 Configuration Directory

The JobScheduler reads the objects described above from the configuration directory and its sub-directories. The configuration directory can be set using:

- `<config configuration_directory="…">`, the default setting is the `./config/live` directory specified in the configuration file.
- `-config`. in which case the JobScheduler expects to find the configuration file in the configuration directory under the name `scheduler.xml`. (The default setting is `./config`)

The JobScheduler monitors the configuration directory and its sub-directories and automatically reads out files which have been added and changed. Deletion of a file leads to the corresponding object in the JobScheduler being deleted.

On Windows systems, the JobScheduler uses the operating system directory monitoring and therefore notices changes immediately. Furthermore, it checks the directories at minute intervals.

On Unix systems, the JobScheduler monitors the directories at intervals of between 5 and 60 seconds. Should no change occur in a directory, the interval is 60 seconds. After a change occurs, the JobScheduler reduces the monitoring interval to five seconds - should no further change occur, the interval is increased in steps back up to sixty seconds.

Caution!
Folders whose names start with a . (dot) will not be monitored with versions 1.5.xx of the JobScheduler engine and newer. Configuration files stored in such folders are ignored.

1.2.2 Files for Process Classes, Locks, Jobs, Job Chains and Permanent Orders

These files contain the XML elements defining objects as follows and are processed according to the following naming convention:

The `name=` attribute should not be specified here. Should it be necessary to define it here, then it must correspond with the file name.

The `replace=` and `spooler_id=` attributes are not valid here.

Example of an Order Controlled Jobs: File `hello_world.job.xml`:

```
<job order="yes">
    <script language="shell"><![CDATA[
      echo hello world
    ]]></script>
</job>
```

Example of a Job Chain : File `echo_hello.job_chain.xml`:

```
<job_chain>
    <job_chain_node  state="start" job="hello_world" next_state="success"
error_state="error"/>
    <job_chain_node  state="success"/>
    <job_chain_node  state="error"/>
</job_chain>
```

Example of an Order: File `echo_hello,echo_trigger.order.xml`:

```
<order>
    <run_time>
      <period repeat="3600"/>
    </run_time>
</order>
```

1.2.3 Directory Mirroring with the JobScheduler

The JobScheduler creates an object corresponding to each file possessing a recognised file name ending or extension (`.job` etc.) and links it with the file. The JobScheduler then monitors the time stamp of each file and in the event of a change occurring proceeds as follows:

- A file name added to the directory causes the JobScheduler to create a new, empty object, which is linked to the file. The object can be inspected using `<show_state>`. For example, a `xxx.job.xml` file is mirrored in the JobScheduler with the `<job name="xxx">` object, even when the file may not be readable or empty.
- A deleted file causes the mirrored object to be deleted from the JobScheduler. Note that deletion is generally delayed, because, for example, a job must wait for the end of a task.
- Should a file be re-added to the directory before the deletion process has been completed, then the JobScheduler will proceed as if a change had been made to the file.
- After a change has been made to a file - i.e. a change has occurred in its timestamp - then the JobScheduler will read the file. If the file can be loaded, then the JobScheduler changes process classes and locks immediately, whereas jobs, job classes and permanent orders are changed after a delay. This allows operations to be completed which are currently being carried out using the object which is to be changed. A more detailed description of this follows below.
- An error occurring whilst a file is being read does not affect a corresponding and already existing object. The JobScheduler notes the error in the object and sends an e-mail. `<show_state>` shows the error.
- Objects with the same (file) names can be present in different directories. The JobScheduler differentiates between these objects through their paths. When it is necessary to link to objects from other directories - e.g.

  ```
  <job  process_class="/my_project/multi.process_class.xml"/>

  <job_chain_node  job="/other_project/hello_world"/>
  ```
  then the paths must be specfiied - either relatively or absolutely.

1.2.4 Effects of the Change and Delete Commands

Commands causing changes to be made to objects do not cause the same changes to be made to the corresponding files.

Commands causing objects to be deleted cause the corresponding files to be deleted.

1.2.5 Behaviour of Individual Object Types

1.2.5.1 Process classes

Changes made to a process class file are immediately taken over by the JobScheduler.

To delete a process class, the JobScheduler first stops all the related tasks. The process class is only deleted when all the tasks have been ended. Until this takes place, the JobScheduler behaves as if the process class has been exhausted.

1.2.5.2 Locks

The JobScheduler immediately takes over changes made to a lock.

To delete a lock, the JobScheduler first stops all related tasks. The lock is only then deleted, when all the tasks have been ended. Until then, the JobScheduler does not start any further jobs and does not allow any other task access to the lock.

1.2.5.3 Jobs

The JobScheduler only takes on a change to a job after all associated tasks have been completed.

The JobScheduler behaves in the same way when deleting a job object. No new tasks will be started.

Should a process class or a lock for a job be missing, then the JobScheduler behaves as if the process class were exhausted or the lock unavailable.

1.2.5.4 Job chains

The JobScheduler only processes changes to a job chain after all the orders in the chain have completed their jobs. This stops further job steps being initiated.

Orders in job chain nodes with the same order state are carried over by the JobScheduler into the changed job chain.

The JobScheduler proceeds in the same manner when deleting job chains.

Should a job chain be missing a job, then orders will collect in the job chain nodes, in the same way as if the job was not ready for execution.

1.2.5.5 Nested Job Chains

Should a job chain be missing a nested job chain `<job_chain_node.job_chain>`, then the complete (parent) job chain will remain unavailable. All subordinate (child) job chains must be available before the JobScheduler can determine whether or not the order identifiers are unique.

In the same way, a subordinate job chain will only be deleted once the superordinate (parent) job chains have been deleted.

1.2.5.6 Permanent Orders

Permanent orders are handled differently to all the other objects. On the one hand, the names of permanent orders have two parts (`job_chain=` and `id=` instead of `name=`) and on the other, the JobScheduler does not always take account of changes made to permanent order files. The JobScheduler only takes account of deleted or changed permanent order files, when:

- the order is not recognised;
- the order has not been started or
- the order is to be repeated by way of `<run_time>` but has not yet been started.

1.2.6 The <show_state> Command

This command returns an `<file_based>` answer for every file-based object.

The `<replacement>` answer indicates when the JobScheduler is in the process of replacing objects from hot folders (jobs, job chains, orders, locks and process classes).

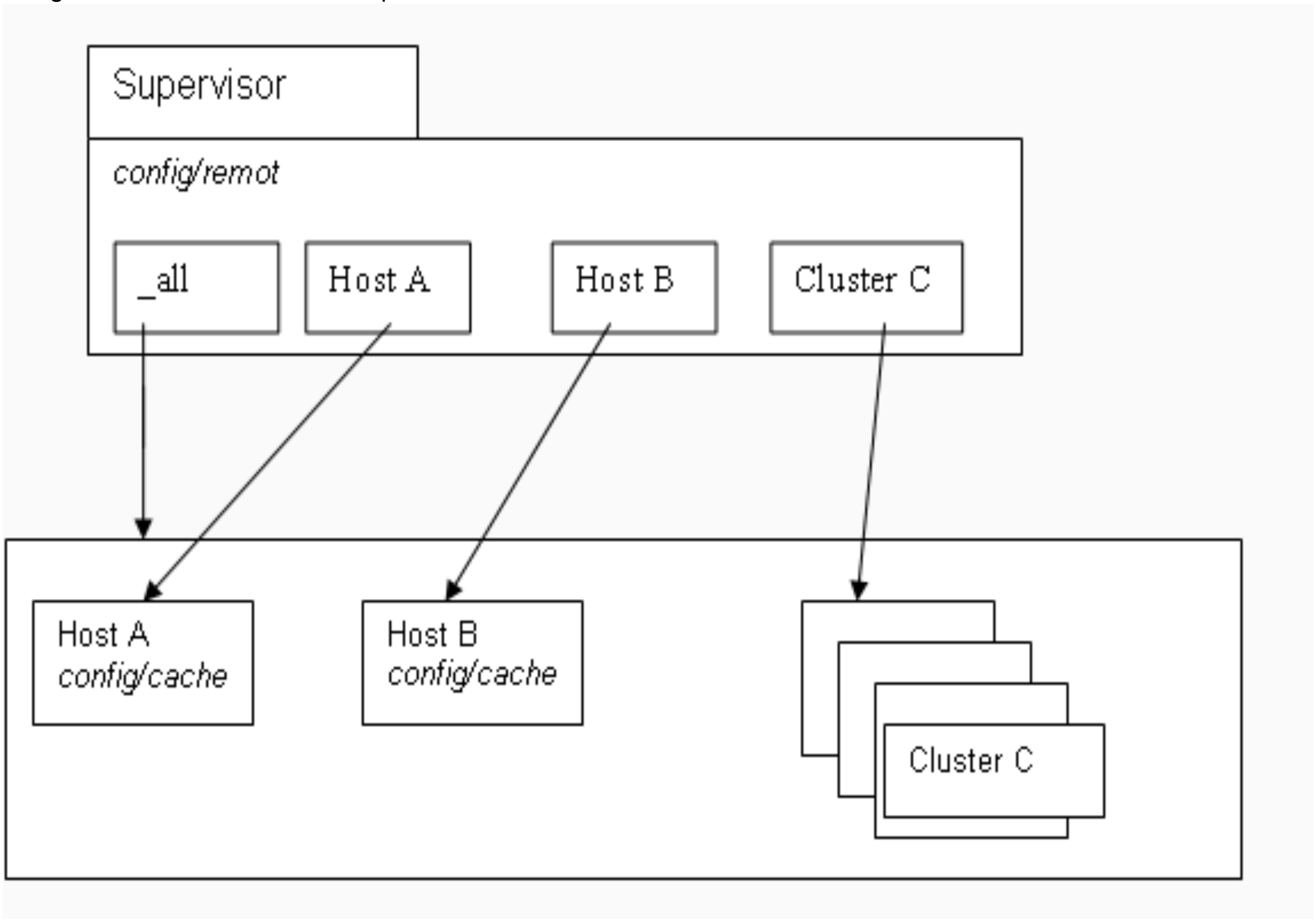
`<removed>` indicates that a file has been deleted but the object associated with the file has not yet been deleted.

1.3 Central Configuration Using a Supervisor JobScheduler

In the centralised administration of the configuration of objects such as jobs, job chains, orders and locks, so called "Workload JobSchedulers" register themselves with a central "Supervisor JobScheduler". The supervising JobScheduler then provides the Workload JobSchedulers with the configuration information for these objects. In addition to the configurations provided by the Supervisor, Workload JobSchedulers can read local configuration information from their own hot folders (page 88) (`./config/live`) and from their own `./config/scheduler.xml` configuration file.

1.3.1 A Typical Configuration

A Supervisor JobScheduler with a `./config/remote` directory. In this directory is an `_all` folder, containing all the general configuration objects for all the Workload JobSchedulers. Each Workload JobScheduler has a corresponding directory (`./config/cache`) containing the configuration information for objects being replicated using the information from the Supervisor.



1.3.2 The Supervisor JobScheduler

The Supervisor JobScheduler administers the configurations of the Workload JobSchedulers in its `./config/remote` directory. The Supervisor sends these configuration files to the relevant Workload JobSchedulers which are running. It sends the configuration information on starting, when a Workload JobScheduler registers itself with the Supervisor and after changes have been made to the configuration files. The Workload JobSchedulers replicate this configuration information in their `./config/cache` directories, which they monitor for changes. The Workload JobSchedulers use this configuration information to configure jobs, job chains, etc. (see also hot folders (page 88)).

In the Supervising JobScheduler's `./config/remote` configuration directory a sub-directory is created for each Workload JobScheduler under the name of `host#port`. The Supervisor's `./config/remote` directory also contains an additional sub-directory for every JobScheduler Cluster it administers (see Backup Clusters (page 145) and Load Balancing (page 170)). This sub-directory contains the `Scheduler ID` which has been allocated to the cluster. The directory structure relevant to each Workload JobScheduler or JobScheduler Cluster is replicated on the Workload JobScheduler or on the cluster.

In addition, the directory structure in the `_all` folder, is replicated on all the Workload JobSchedulers. This means that it is possible for definitions of all the objects which are valid for all the Workload JobSchedulers to be administered at a central place.

Should the supervising JobScheduler fail, then the Workload JobScheduler which was last successfully updated with the replicated configuration data can simply be started as usual.

1.3.3 Registering a JobScheduler with the Supervisor JobScheduler

A Workload JobScheduler registers itself with the Supervisor JobScheduler using the `<config supervisor="…">` attribute in the `config` element as defined in its configuration file `./config/scheduler.xml`. This attribute defines the host and port of the supervising JobScheduler by the syntax `host:port`.

Should the Supervisor JobScheduler not be available, then the Workload JobScheduler starts using its last successfully replicated configuration.

1.3.4 Effectiveness of Local Changes to the Configuration

Local configurations can be created in addition to those provided by the Supervisor JobScheduler.

The Workload JobScheduler's `./config/scheduler.xml` configuration file is used to define local object configurations. The hot folders (page 88) in the `./config/live` directory can also be used to define local object configurations.

The configuration centrally stored on the Supervisor JobScheduler and successfully replicated has priority over any local configurations. In the event of local and central configurations having the same name, then the central one will be given priority. When an element in the central configuration is added to a local configuration, then the Workload JobScheduler will become aware of this and reject the local element. An appropriate warning will then be added to the Workload JobScheduler's log file.

When an object that already exists in a local configuration, is added to the configuration of the Supervisor JobScheduler, the local configuration will be overwritten. The local configuration will be retained but no longer used and a warning added to the Workload JobScheduler's log file as described above.

When an object that exists in a both a local and a central configuration is deleted from the central configuration, then it will also be deleted from the local Workload JobScheduler configuration.

What exactly happens to the local configuration on the Workload JobScheduler configuration depends on how the duplicated job is configured on the Workload JobScheduler:

a) If a job is locally configured in hot folders (page 88) in `./config/live`: The local configuration will be used
b) If a job is configured in the `./config/scheduler.xml` file: The configuration will be read when the JobScheduler is next started.

1.3.5 Taking Over an Existing Configuration in the Central Administration

The following steps should be followed, when an existing, locally administered JobScheduler is to be included in the central administration of a Supervisor JobScheduler:

- A folder for the Workload JobScheduler should be created on the Supervisor (`./config/remote/host#port`).
- A `./config/cache` folder should be created on the Workload JobScheduler.
- The content of the Workload JobSchedulers' `./config/live` hot folder (page 88) should be copied to the `./config/remote/host#port` on the Supervisor JobScheduler.
- Configure the registration of the Workload JobScheduler on the Supervisor. To do this, the `<config supervisor="…">` entry should be added to the Workload JobScheduler's `./config/scheduler.xml` configuration file.

1.3.6 Behaviour of the JobSchedulers on Starting

1.3.6.1 Workload JobSchedulers

The Workload JobScheduler registers itself with the Supervisor and orders the configuration. Operation can begin after the configuration has been replicated. Should the supervising JobScheduler not be available, then the Workload JobScheduler uses its existing configuration - to be more exact, it uses the last successfully replicated profile. Should replication later become possible, then it will be carried out automatically. Any changes made in the central configuration will immediately become effective.

1.3.6.2 Supervisor JobScheduler

The supervising JobScheduler replicates the Workload JobScheduler's configurations when it starts.

1.4 factory.ini File

The factory.ini file contains settings for the JobScheduler. Note that empty settings are ignored and do not overwrite those made in the (page 6).

The location of this file can be specified when starting the JobScheduler using the `-ini` option. For example,

```
…scheduler installation path…\bin\scheduler.exe
-ini=C:/Programme/scheduler/config/factory.ini
```

During installation of the JobScheduler, the factory.ini file is saved in the

`…scheduler installation path…\config`
directory. Calls using the JobScheduler start script then automatically set the path to this file correctly.

If the path to the factory.ini file is not specified when calling the JobScheduler, then the JobScheduler will attempt to find it by looking according to the following criteria:

This file should be saved under Windows in the folder in which Windows expects to find the .ini files. Normally this is the c:\windows folder. Otherwise the file will be sought in the users' home directory.

Settings

Section: java

`class_path` = *class_path* Class path for the Java virtual machine

This file is used to extend the `class_path` of directories or jars which are only valid for the JobScheduler.

`options` = *text* Options for Java

Java VM parameters and Java properties can be specified using this setting. Java properties can be specified using the "-D" notation. For example:

```
options  = -Dlog4j.configuration="file:///${SCHEDULER_HOME}/lib/log4j.properties"
```
See the Java Documentation for a list of parameters.

This setting precedes changes in the `sos.ini` (section `[java]`, entry `options= …`) file.

Section: job

`history` = *yes*|*no* Write history?

Specifies whether a task history should be written. If yes, then the JobScheduler makes an entry in the database for every task carried out. Should the JobScheduler not be using a database, this entry will be made in a file specified in the `-log-dir` directory.

The `factory.ini` (section `[spooler]`, entry `history= yes`) setting is overwritten by this parameter.

`history_archive` = *yes*|*no*|*gzip* Archive history files?

This option causes the JobScheduler to archive and compress the history files with `gzip` after its last run, should the JobScheduler be writing the history in files and not in a database.

The `factory.ini` (section `[spooler]`, entry `history_archive= no`) setting is overwritten by this parameter.

`history_columns` = *name, name, ...* Additional columns in the history

A task using the `Task.set_history_field()` method can include fields specified here in the database table.

The `factory.ini` (section `[spooler]`, entry `history_columns= …`) setting is overwritten by this parameter.

`history_file` = *file_name* Name of the History File (for operation without a database)

(Since version 1.5, file based history are not longer written.)

The `factory.ini` (section `[spooler]`, entry `history_file= …`) setting is overwritten by this parameter.

`history_on_process` = *yes*|*no*|*number* Write history entry after calling spooler_process()?

Should the Job JobScheduler be writing a history, then the settings `history_on_process=no` or `=0` cause it to make an entry in the history at the start of a task.

When `history_on_process=yes` or `=1` then JobScheduler only makes an entry after `spooler_process()` has been called. This means that no entry will be made, when `spooler_open()` returns `false`.

When a number is specified in this setting (`history_on_process=x`), the JobScheduler makes an entry in the history only after the x-th execution of `spooler_process()`.

The `factory.ini` (section `[spooler]`, entry `history_on_process= 0`) setting is overwritten by this parameter.

Messages

`[ERROR]`	`SCHEDULER-335`	Only "yes", "no" and a number are allowed with ="": error

`history_with_log` = *yes|no|gzip* Write a protocol in the history?

The JobScheduler can add the task protocol to the history when the history is being recorded in a database, if required, compressed with gzip. (Here it is important to check that the protocol - which is saved in binary format - can be decompressed after being saved in the database.)

See the command: `<show_task what="log">` (page 211).

The `factory.ini` (section `[spooler]`, entry `history_with_log= no`) setting is overwritten by this parameter.

`log_level` = *log_level* Limit protocol level

Defines the level with which protocol entries should be written. Every protocol entry is given one of the following categories: `error`, `warn`, `info`, `debug1` to `debug9` (`debug1` is the same as `debug`).

The `-log-level` option has precedence over this parameter.

The `factory.ini` (section `[spooler]`, entry `log_level= info`) setting is overwritten by this parameter.

`log_mail_bcc` = *email_address* E-mail bcc recipient

The `factory.ini` (section `[spooler]`, entry `log_mail_bcc= …`) setting is overwritten by this parameter.

`log_mail_cc` = *email_address* E-mail cc recipient

The `factory.ini` (section `[spooler]`, entry `log_mail_cc= …`) setting is overwritten by this parameter.

`log_mail_from` = *email_address* E-mail sender

The `factory.ini` (section `[spooler]`, entry `log_mail_from= …`) setting is overwritten by this parameter.

`log_mail_subject` = *text* E-mail subject

The `factory.ini` (section `[spooler]`, entry `log_mail_subject= …`) setting is overwritten by this parameter.

`log_mail_to` = *email_address* E-mail recipient

The `factory.ini` (section `[spooler]`, entry `log_mail_to= …`) setting is overwritten by this parameter.

`mail_on_delay_after_error` = Suppress <delay_after_error>

Prerequisite: `mail_on_error=yes` or `mail_on_warning=yes`

This setting reduces the numerous e-mails after a job is restarted by way of `<delay_after_error>`.

This setting only works when `<delay_after_error>` has been set for as job. Should this not be the case, then `mail_on_delay_after_error=all` applies.

The `factory.ini` (section `[spooler]`, entry `mail_on_delay_after_error= first_and_last_only`) setting is overwritten by this parameter.

`mail_on_error` = *yes|no* Send an e-mail should a task end with an error

The `factory.ini` (section `[spooler]`, entry `mail_on_error= no`) setting is overwritten by this parameter.

`mail_on_process` = *yes|no|number* Send an e-mail at the start of a task using spooler_process()

Causes the task log to be sent when a task has completed at least the specified number of steps - i.e. calls of `spooler_process()`. Because non-API tasks do not have steps, the JobScheduler counts each task as a single step.

`yes` corresponds to 1, `no` corresponds to 0.

The `factory.ini` (section `[spooler]`, entry `mail_on_process= 0`) setting is overwritten by this parameter.

`mail_on_success` = *yes|no* Send an e-mail on successful completion of a task

The `factory.ini` (section `[spooler]`, entry `mail_on_success= no`) setting is overwritten by this parameter.

`mail_queue_dir` = *directory* Directory for e-mails which temporarily cannot be sent

E-mails which cannot be sent (because, for example, the SMTP server cannot be contacted) are stored in this directory.

In order to send these e-mails later it is necessary to write a job which calls up the `Mail.dequeue()` method.

This setting is generally made in `sos.ini` (section `[mail]`, entry `queue_dir= ...`).

Environment variables (e.g. `$HOME`) are replaced by this attribute (see Settings which Allow Environment Variables to be Called (page 106)).

The `factory.ini` (section `[spooler]`, entry `mail_queue_dir= ...`) setting is overwritten by this parameter.

The `sos.ini` (section `[mail]`, entry `queue_dir= ...`) setting is overwritten by this parameter.

`smtp` = *host_address* E-mail SMPT server hostname or IP number

These settings are generally made using `sos.ini` (section `[mail]`, entry `smtp= ...`).

`smtp=-queue` stops e-mails being sent. Instead mails are written into the file specified in `queue_dir`. See also `sos.ini` (section `[mail]`, entry `queue_only= ...`).

The `factory.ini` (section `[spooler]`, entry `smtp= ...`) setting is overwritten by this parameter.

The `sos.ini` (section `[mail]`, entry `smtp= ...`) setting is overwritten by this parameter.

Section: smtp

`mail.smtp.password` = *password* Password for Authentification on an SMTP Server

The password is sent together with the `mail.smtp.user` entry when registering on an SMTP server in order to be able to send e-mails.

`mail.smtp.user` = *name* Name for Authentification on an SMTP Server

Section: spooler

`config` = *file_name* Configuration file

Defines the Configuration File (page 6).

The `-config` option has precedence over this parameter.

`db` = *connection_string* Database connection string

The database connection string for the history. Should no value be specified here, then the files will be saved in .csv format. See `factory.ini` (section `[spooler]`, entry `history_file= …`) (page 99).

A simple file name ending in `.mdb` (e.g. `scheduler.mdb`) can also be specified here when the JobScheduler is running on Windows. The JobScheduler then uses a Microsoft MS Access database of this name, which is located in the protocol directory (see the option `-log-dir`). Should such a database not exist, then the JobScheduler will create this database.

The JobScheduler automatically creates the tables necessary for this database.

Example:

```
;    SQL Server 2000 via msbase.jar, msutil.jar, mssqlserver.jar
db = jdbc -class=com.microsoft.jdbc.sqlserver.SQLServerDriver
jdbc:microsoft:sqlserver://localhost:1433;selectMethod=Cursor;databaseName=scheduler
-user=scheduler -password=secret
;    SQL Server 2000, 2005 via sqljdbc.jar
db = jdbc -class=com.microsoft.sqlserver.jdbc.SQLServerDriver
jdbc:sqlserver://localhost:1433;sendStringParametersAsUnicode=false;selectMethod=cursor
;databaseName=scheduler -user=scheduler -password=secret

;    MySQL 4.1.7, 5.x
db = jdbc -class=com.mysql.jdbc.Driver jdbc:mysql://localhost/scheduler:3306
-user=scheduler -password=secret

;    Oracle 8.1.7, 9i, 10g
db = jdbc -class=oracle.jdbc.driver.OracleDriver jdbc:oracle:thin:@localhost:1521:orcl
-user=scheduler -password=secret

;    PostgreSQL 8.x
db   = jdbc -class=org.postgresql.Driver jdbc:postgresql://localhost:5432/scheduler
-user=scheduler -password=secret

;    IBM DB2 8
db   = jdbc -class=com.ibm.db2.jcc.DB2Driver
jdbc:db2://localhost:50000/scheduler:driverType=2;retrieveMessagesFromServerOnGetMessag
e=true; -user=scheduler -password=secret

;    Firebird 1.5
db   = jdbc -class=org.firebirdsql.jdbc.FBDriver
jdbc:firebirdsql://localhost:3050/scheduler -user=scheduler -password=secret

;    ODBC
db = odbc -db=scheduler_datasource -user=scheduler -password=secret

;    MS Access Datenbank
db = scheduler.mdb
```

`db_check_integrity` = *yes|no*

The JobScheduler carries out additional database integrity tests.

`db_history_table` = *name* Name of the history database table

See also `Spooler.db_history_table_name()`

`db_members_table` =

`db_order_history_table` = *name* Name of the order history database table

See also `Spooler.db_order_history_table_name()`

`db_order_step_history_table` = *name* Name of the database table containing the steps in the order history

`db_orders_table` = *name* Name of the order database table

See also `Spooler.db_orders_table_name()`

`db_tasks_table` = *name* Name of the task database table

See also `Spooler.db_tasks_table_name()`

`db_variables_table` = *name* Name of the JobScheduler internal variable database table

The JobScheduler records internal counters, for example, the ID of the next free task, in this database table.

See also `Spooler.db_variables_table_name()`

`history` = *yes*|*no* Write history?

Specifies whether a task history should be written. If yes, then the JobScheduler makes an entry in the database for every task carried out. Should the JobScheduler not be using a database, this entry will be made in a file specified in the `-log-dir` directory.

The `factory.ini` (section `[job]`, entry `history= yes`) setting has precedence over this parameter.

`history_archive` = *yes*|*no*|*gzip* Archive history files?

This option causes the JobScheduler to archive and compress the history files with `gzip` after its last run, should the JobScheduler be writing the history in files and not in a database.

The `factory.ini` (section `[job]`, entry `history_archive= no`) setting has precedence over this parameter.

`history_columns` = *name, name, ...* Additional columns in the history

A task using the `Task.set_history_field()` method can include fields specified here in the database table.

The `factory.ini` (section `[job]`, entry `history_columns= …`) setting has precedence over this parameter.

`history_file` = *file_name* Name of the History File (for operation without a database)

(Since version 1.5, file based history are not longer written.)

The `factory.ini` (section `[job]`, entry `history_file= …`) setting has precedence over this parameter.

`history_on_process` = *yes*|*no*|*number* Write history entry after calling spooler_process()?

Should the Job JobScheduler be writing a history, then the settings `history_on_process=no` or `=0` cause it to make an entry in the history at the start of a task.

When `history_on_process=yes` or `=1` then JobScheduler only makes an entry after `spooler_process()` has been called. This means that no entry will be made, when `spooler_open()` returns `false`.

When a number is specified in this setting (`history_on_process=x`), the JobScheduler makes an entry in the history only after the x-th execution of `spooler_process()`.

The `factory.ini` (section `[job]`, entry `history_on_process= 0`) setting has precedence over this parameter.

Messages

[ERROR] SCHEDULER-335 Only "yes", "no" and a number are allowed with ="": error

`history_with_log` = *yes|no|gzip* Write a protocol in the history?

The JobScheduler can add the task protocol to the history when the history is being recorded in a database, if required, compressed with gzip. (Here it is important to check that the protocol - which is saved in binary format - can be decompressed after being saved in the database.)

See the command: `<show_task what="log">` (page 211).

The `factory.ini` (section `[job]`, entry `history_with_log= no`) setting has precedence over this parameter.

`html_dir` = *directory* Directory for HTML files

The directory in which the HTML files for the JobScheduler HTTP server are to be found.

Should no entry be made here, then the JobScheduler uses the `html` directory in the configuration file directory.

`id` = *scheduler_id* JobScheduler identifier (id)

The JobScheduler only selects elements in the XML configuration whose `spooler_id` attributes are either empty or set to the value given here.

When the JobScheduler ID is not specified here, then the JobScheduler ignores the `spooler_id=` XML attribute and selects all the elements in the XML configuration.

See, for example, `<config>` (page 21).

The `-id` option has precedence over this parameter.

`include_path` = *directory* Directory path for <include>

The directory of the files which are to be included by the `<include>` element.

Environment variables (e.g. `$HOME`) are replaced by this attribute (see Settings which Allow Environment Variables to be Called (page 106)).

The `-include-path` option has precedence over this parameter.

`<config include_path="…">` is overwritten by this parameter.

`log` = *file_name* scheduler.log file name

This setting causes the JobScheduler to write a detailed protocol. This protocol is intended for use in problem diagnosis. The file name should be fully specified here (i.e. as a full path).

A plus character (+) written directly before the file name causes an already existing protocol to be continued. Otherwise such a protocol will be overwritten.

Categories can be used to extend or restrict the log file. Category names are added (separated by spaces) before the file name, which is then preceded by a larger than (>) sign.

The list of categories can be found here.

Example:

```
log = c:/tmp/scheduler.log
log = scheduler.wait >scheduler.log
log = scheduler.wait com_server.* >scheduler.log
```

The `-log` option has precedence over this parameter.

`log_dir` = *directory* Protocol directory

The directory in which the JobScheduler writes log files.

`log_dir=*stderr` allows the JobScheduler to write log files to the standard output (`stderr`, normally the screen) .

The `-log-dir` option has precedence over this parameter.

`log_level` = *log_level* Limit protocol level

Defines the level with which protocol entries should be written. Every protocol entry is given one of the following categories: `error`, `warn`, `info`, `debug1` to `debug9` (`debug1` is the same as `debug`).

The `-log-level` option has precedence over this parameter.

The `factory.ini` (section `[job]`, entry `log_level= info`) setting has precedence over this parameter.

`log_mail_bcc` = *email_address* E-mail bcc recipient

The `factory.ini` (section `[job]`, entry `log_mail_bcc= ...`) setting has precedence over this parameter.

`log_mail_cc` = *email_address* E-mail cc recipient

The `factory.ini` (section `[job]`, entry `log_mail_cc= ...`) setting has precedence over this parameter.

`log_mail_from` = *email_address* E-mail sender

The `factory.ini` (section `[job]`, entry `log_mail_from= ...`) setting has precedence over this parameter.

`log_mail_subject` = *text* E-mail subject

The `factory.ini` (section `[job]`, entry `log_mail_subject= ...`) setting has precedence over this parameter.

`log_mail_to` = *email_address* E-mail recipient

The `factory.ini` (section `[job]`, entry `log_mail_to= ...`) setting has precedence over this parameter.

`mail_encoding` =

`mail_on_delay_after_error` = Suppress <delay_after_error>

Prerequisite: `mail_on_error=yes` or `mail_on_warning=yes`

This setting reduces the numerous e-mails after a job is restarted by way of `<delay_after_error>`.

This setting only works when `<delay_after_error>` has been set for as job. Should this not be the case, then `mail_on_delay_after_error=all` applies.

The `factory.ini` (section `[job]`, entry `mail_on_delay_after_error= first_and_last_only`) setting has precedence over this parameter.

`mail_on_error` = *yes*|*no* Send an e-mail should a task end with an error

The `factory.ini` (section `[job]`, entry `mail_on_error= no`) setting has precedence over this parameter.

`mail_on_process` = *yes*|*no*|*number* Send an e-mail at the start of a task using spooler_process()

Causes the task log to be sent when a task has completed at least the specified number of steps - i.e. calls of `spooler_process()`. Because non-API tasks do not have steps, the JobScheduler counts each task as a single step.

`yes` corresponds to 1, `no` corresponds to 0.

The `factory.ini` (section `[job]`, entry `mail_on_process= 0`) setting has precedence over this parameter.

`mail_on_success` = *yes*|*no* Send an e-mail on successful completion of a task

The `factory.ini` (section `[job]`, entry `mail_on_success= no`) setting has precedence over this parameter.

`mail_on_warning` = *yes*|*no* An e-mail is sent after a task has ended, should a warning occur

`mail_queue_dir` = *directory* Directory for e-mails which temporarily cannot be sent

E-mails which cannot be sent (because, for example, the SMTP server cannot be contacted) are stored in this directory.

In order to send these e-mails later it is necessary to write a job which calls up the `Mail.dequeue()` method.

This setting is generally made in `sos.ini` (section `[mail]`, entry `queue_dir= …`).

Environment variables (e.g. `$HOME`) are replaced by this attribute (see Settings which Allow Environment Variables to be Called (page 106)).

The `factory.ini` (section `[job]`, entry `mail_queue_dir= …`) setting has precedence over this parameter.

The `sos.ini` (section `[mail]`, entry `queue_dir= …`) setting is overwritten by this parameter.

`mail_queue_only` = *yes*|*no* Do not send e-mail, add it to the e-mail queue

When this parameter is set to `yes`, then e-mails are sot sent but added to the e-mail queue - see `sos.ini` (section `[mail]`, entry `queue_dir= …`).

This setting is generally made in `sos.ini` (section `[mail]`, entry `queue_only= …`).

The `sos.ini` (section `[mail]`, entry `queue_only= …`) setting is overwritten by this parameter.

`max_db_errors` = *number* Number of database errors before the JobScheduler shuts itself down

The JobScheduler accepts this number of database errors. In the event of this number of errors being exceeded, and the option `need_db=yes`, then the JobScheduler shuts itself down immediately. Should `need_db=no` then the JobScheduler will continue without the database.

In the case of `need_db= yes`, then the errors occurring when attempting to open a database will not be not counted. This means that the JobScheduler can wait for a database which has not yet been opened.

`need_db` = *yes*|*no*|*strict* Is a database necessary?

`need_db=no`

Should the option `db=` not have been set, then the JobScheduler issues a warning and operates without a database.

The JobScheduler also issues a warning and operates without a database, should either the opening of a database or the automatic creation of the necessary tables not be successful.

In the event of a database error when the JobScheduler is in operation, the JobScheduler closes and reopens the database. Should the JobScheduler be unsuccessful in either of these operations, then it will operate without the database.

In Cluster operation (`-exclusive` or `-distributed-orders`) the JobScheduler rejects `need_db=no` with the following message:
`SCHEDUL`
`ER-358`

`need_db=yes`

In this case, in the event of the `db=` setting not being specified, then the JobScheduler will not start but return the following message, .

The JobScheduler will not start should either the opening of the database or the automatic creation of tables not function successfully.

The JobScheduler attempts to close and then reopen the database should a database error occur whilst it is in operation. Should the JobScheduler not be successful in these operations, then it will reattempt to open the database at one minute intervals. It will continue with these attempts to reopen the database indefinitely. The JobScheduler will not attempt to manage any tasks until it regains contact with the database.

The JobScheduler sends an E-mail (page 175) should repeated attempts to reopen the database be unsuccessful.

`need_db=strict`

As `need_db=yes` with the following exceptions:

Should a database error occur whilst the JobScheduler is in operation, then it attempts to close and then reopen the database. Should an error repeat itself then the JobScheduler will repeat this process for the number of times specified in the `max_db_errors=` setting. Should the JobScheduler not be successful reopening the database, then it will shut itself down.

`order_history` = *yes_no* Write orders in the history?

A separate history is recorded for orders.

`order_history_with_log` = *yes*|*no*|*gzip* Record the order log in the history?

The JobScheduler can record the order log in the database - compressed with gzip if required. (Here it is important to check that the log - which is saved in binary format - can be decompressed after being saved in the database.)

See the command: `<show_order what="log">` (page 209).

`param` = *text* For free use

Free text. This parameter can be read using `spooler. param`.

The `-param` option has precedence over this parameter.

`smtp` = *host_address* E-mail SMPT server hostname or IP number

These settings are generally made using `sos.ini` (section `[mail]`, entry `smtp= …`).

`smtp=-queue` stops e-mails being sent. Instead mails are written into the file specified in `queue_dir`. See also `sos.ini` (section `[mail]`, entry `queue_only= …`).

The `factory.ini` (section `[job]`, entry `smtp= …`) setting has precedence over this parameter.

The `sos.ini` (section `[mail]`, entry `smtp= …`) setting is overwritten by this parameter.

`subprocess.own_process_group` = *yes|no* Start a Sub-Process in its own Group

This is the default setting for `Subprocess.own_process_group`

1.5 sos.ini

The SOS licence keys are written in the `sos.ini` file.

Further, general settings for Java or e-mail are made here. Java settings can also be made in the `factory.ini` file, in the `[java]` (page 97) section.

When calling the JobScheduler from the command line, the path to the file can be specified using the `-sos.ini` option. For example,

```
…scheduler installation path…\bin\scheduler.exe
-sos.ini=C:/Programme/scheduler/config/sos.ini
```

During installation of the JobScheduler, the `sos.ini` file is written to the

`…scheduler installation path…\config`
directory. Calls using the JobScheduler start script automatically set this path corrrectly.

Should settings in the `sos.ini` file also be used in other products from the SOS GmbH., then this file can be saved centrally.

Should the `sos.ini` file not be specified when starting the JobScheduler, then the JobScheduler will attempt to find it by looking according to the following criteria:

This file should be saved under Windows in the folder in which Windows expects to find the .ini files. Normally this is the c:\windows folder. Otherwise the file will be sought in the users' home directory.

Settings

Section: java

`class_path` =

`debug` = *yes_no*

`debug=yes` causes the Java call to be entered in the `scheduler.log` file. This setting also sets Java in the debug mode, whereby, for example, an exception causes the call stack to be written to stdout (or stderr).

`javac` = *path* Path to the Java compiler

This setting is used to specify the path to the Java compiler. The JobScheduler uses this compiler when quell code is directly written in the `<script>` element, for example, when developing Java jobs

`options` = *text*

Java VM parameters and Java properties can be specified using this setting. Java properties can be specified using the "-D" notation.

For example:

```
options  = -Dlog4j.configuration="file:///${SCHEDULER_HOME}/lib/log4j.properties"
```

See the Java Documentation for a list of parameters.

This setting precedes changes in the `sos.ini` (section `[java]`, entry `options= …`) file.

`vm` = *file_name* Java Virtual Machine File Name

The default settings are `jvm.dll` for Windows, `jvm.sl` for HP-UX and `jvm.so` for all other UNIX variants.

The Windows preset value is read out of the registry. The `CurrentVersion` entry, which contains the current version number (such as "`1.5`"), is read from the

```
HKEY_LOCAL_MACHINE\software\JavaSoft\Java Runtime Environment
```

key. The version key - e.g.:

```
HKEY_LOCAL_MACHINE\software\JavaSoft\Java Runtime Environment\1.5
```

is then read and the path to the `jvm.dll` file taken from the `RuntimeLib` entry.

Section: licence

`key1` = *licence_key* Licence Key

Users with more than one licence key can enter subsequent keys using `key2=`, `key3=` etc.. Note that the numbering must be without spaces.

Section: mail

`queue_dir` = *directory* Directory for e-mails which temporarily cannot be sent

E-mails which cannot be sent (because, for example, the SMTP server cannot be contacted) are stored in this directory.

In order to send these e-mails later it is necessary to write a job which calls up the `Mail.dequeue()` method.

This setting is generally made in `sos.ini` (section `[mail]`, entry `queue_dir= …`).

Environment variables (e.g. `$HOME`) are replaced by this attribute (see Settings which Allow Environment Variables to be Called (page 106)).

The `factory.ini` (section `[job]`, entry `mail_queue_dir= …`) setting has precedence over this parameter.

The `factory.ini` (section `[spooler]`, entry `mail_queue_dir= …`) setting has precedence over this parameter.

`queue_only` = *yes*|*no* Do not send e-mail, add it to the e-mail queue

When this parameter is set to `yes`, then e-mails are sot sent but added to the e-mail queue - see `sos.ini` (section `[mail]`, entry `queue_dir= …`).

This setting is generally made in `sos.ini` (section `[mail]`, entry `queue_only= …`).

The `factory.ini` (section `[spooler]`, entry `mail_queue_only= …`) setting has precedence over this parameter.

`smtp` = *host_address* E-mail SMPT server hostname or IP number

These settings are generally made using `sos.ini` (section `[mail]`, entry `smtp= …`).

`smtp=-queue` stops e-mails being sent. Instead mails are written into the file specified in `queue_dir`. See also `sos.ini` (section `[mail]`, entry `queue_only= …`).

The `factory.ini` (section `[job]`, entry `smtp= …`) setting has precedence over this parameter.

The `factory.ini` (section `[spooler]`, entry `smtp= …`) setting has precedence over this parameter.

1.6 Settings which Allow Environment Variables with ${…} to be Called

Environment variables with the syntax `${` *name* `}` can be called in some XML attributes and `.ini` file settings. The `$` *name* is also possible when a special character (apart from the understroke _) follows the variable name.

The »`$`« character remains, when no »`{`« or letter follows the name. Similiarly, »`\$`« returns »`$`« when prefixed with »`\`«.

The values of the environment variables as the JobScheduler started remain valid.

Variables can also be replaced by programs using the `Variable_set.substitute()` method.

Example:

```
# File factory.ini
[java]
class_path = ${SCHEDULER_HOME}/lib/sos.spooler.jar
class_path = $SCHEDULER_HOME/lib/sos.spooler.jar
class_path = \\otherhost\C$\lib\classes.jar

<params>
    <param name="txt_file_regex" value="\.txt$"/>
</params>
```

1.6.1 XML Attributes

`<base file="…">`	Basic Configuration
`<config include_path="…">`	Configuration, Directory path for <include>
`<file_order_sink move_to="…">`	File Order Sink
`<file_order_source directory="…">`	File Order Source
`<http_directory path="…">`	HTTP File Directory, File system path
`<include file="…">`	Includes text from a file, Path to file to be included
`<include live_file="…">`	Includes text from a file, Path to the file to be added from the configuration directory
`<param value="…">`	Individual Parameters

`<start_when_directory_changed directory="…">`	Directory Monitoring
`<web_service url_path="…">`	Web Service, The URL path used to reach a service
`<web_service request_xslt_stylesheet="…">`	Web Service, Transforms a request into a JobScheduler command
`<web_service response_xslt_stylesheet="…">`	Web Service, Transforms a reply to a command into a Web Service response
`<web_service forward_xslt_stylesheet="…">`	Web Service, Forwarding after an order or task has been completed

1.6.2 Files factory.ini

`[spooler]` factory.ini (section `[spooler]`, entry `include_path= …`)	Directory path for <include>
`[spooler]` factory.ini (section `[spooler]`, entry `mail_queue_dir= …`)	Directory for e-mails which temporarily cannot be sent
`[job]` factory.ini (section `[job]`, entry `mail_queue_dir= …`)	Directory for e-mails which temporarily cannot be sent

1.6.3 Files sos.ini

`[mail]` sos.ini (section `[mail]`, entry `queue_dir= …`)	Directory for e-mails which temporarily cannot be sent

1.7 Command Line Operation

The JobScheduler is started using a straightforward command.

```
C:\>…scheduler installation path…\bin\scheduler.exe …\my_scheduler_configuration.xml
…
```

```
user@host:~>…installation path…/bin/scheduler …/my_scheduler_configuration.xml …
```

Parameters for the command line operations described here and which are written to the right of the "=" can be set in inverted commas ("") or in apostrophes / single quotes ('').

The JobScheduler installation program creates the start script - either `.\bin\jobscheduler.cmd` (Windows) or `./bin/jobscheduler.sh` (Unix), in which command line options and environment variables are already set. This script can be modified as required.

Note that the start script and environment variables are described in the »Installation and Configuration« handbook.

The command line can be used in the following ways:

- Starting the JobScheduler
- Installation of a Windows Service
- Forwarding a Job to a JobScheduler
- Forwarding an Order to a Running JobScheduler
- Sending an XML Command to a Running JobScheduler
- Stopping a JobScheduler with 'kill'

- Expand the Java Class Path
- Show Version Number

Starting the JobScheduler

`scheduler`

`-config`=*file_name*	Configuration file
`-log`=*file_name*	scheduler.log file name
`-log-dir`=*directory*	Protocol directory
`-id`=*scheduler_id*	JobScheduler identifier (id)
`-cd`=*directory*	Working Directory
`-pid-file`=*File Name*	File name for process ID
`-log-level`=*log_level*	(Initial value: info) Limit protocol level
`-param`=*text*	For free use
`-include-path`=*directory*	Directory path for <include>
`-port`=*number*	(Initial value: 0) HTTP, TCP and UDP ports for control commands for the JobScheduler
`-tcp-port`=*number*	(Initial value: 0) Port for HTTP and TCP commands for the JobScheduler
`-time-zone`=*text*	JobScheduler time zone
`-udp-port`=*number*	(Initial value: 0) Port for UDP commands for the JobScheduler
`-ip-address`=*ip_number*	(Initial value: 0.0.0.0) The interface IP address for TCP and UDP
`-reuse-port`	Reuse of the TCP and UDP ports
`-cmd`=*xmlcommand*	Immediately executed commands
`-ini`=*file name*	Alternative factory.ini file
`-sos.ini`=*file name*	Alternative sos.ini file
`-program-file`=*file name*	JobScheduler file name
`-service`	Start as a daemon
`-validate-xml`	Validate XML Ddocuments against an embedded schema
`-use-xml-schema`=*dateiname*	Schema for validating the JobScheduler configuration
`-env`=*name=value*	Set Environment Variables
`-exclusive`	Starts the JobScheduler for Exclusive Service
`-backup`	Starts a JobScheduler as a Backup Scheduler
`-backup-precedence`=*integer*	(Initial value: 1) Priority Amongst Backup JobSchedulers
`-distributed-orders`	Distributed Orders
`-configuration-directory`	Configuration directory
`-java-classpath`=*file_names*	Java class path for JobScheduler process
`-java_options`=*text*	Java options for JobScheduler process
`-job-java-classpath`=*file_names*	Java class path for Jobs
`-job_java_options`=*text*	Java options for Jobs

The `scheduler` file name must be completely and absolutely specified and be given an "`.exe`" ending on Windows systems. This is because the JobScheduler requires the complete file name in order to be able to start either a job or itself.

Example:

```
c:\bin\scheduler.exe c:\scheduler\config.xml -log-dir=c:\scheduler\logs
```

Installation of a Windows Service

`scheduler`

`-install-service=`*name*	Install as a Windows service
`-remove-service=`*name*	Remove a Windows service
`-service-name=`*name*	(Initial value: sos_scheduler) Windows service internal name
`-service-display=`*text*	Windows service name
`-service-descr=`*text*	Windows service description
`-need-service=`*name*	Service required by the JobScheduler (Windows only)

Forwarding a Job to a JobScheduler

`scheduler`

`-scheduler=`*host:port*	The JobScheduler TCP address
`-log=`*file_name*	scheduler.log file name
`-process-class=`*name*	
`-language=`*script_language*	(Initial value: shell) The Job Script language
`-at=`*yyyy-mm-dd HH:MM*	(Initial value: now) Start time

A temporary job is forwarded to the target JobScheduler, which is addressed using `-scheduler=host:port`.

This job only exists until the JobScheduler is restarted.

Example:

```
echo ls -l | scheduler -scheduler=localhost:4444 -at="2006-04-04 12:00"
```

Forwarding an Order to a Running JobScheduler

`scheduler`

`-scheduler`	
`-log`	
`-job-chain=`*name*	Job Chain
`-order-id=`*id*	The Order ID
`-title=`*text*	The Order Title

`-scheduler=host:port` is used to define the JobScheduler which an order is forwarded to.

The order parameters (see `Order.payload`) can be forwarded to the command line in the form:*name* = *value*.

Example:

```
scheduler -scheduler=localhost:4444 -job-chain=my_job_chain -order-id=123 city=Berlin
phone="+4930 864790-0"
```

Sending an XML Command to a Running JobScheduler

`scheduler`

`-tcp-port`	
`-send-cmd=`*xmlcommand*	Sending a command to another JobScheduler

Stopping a JobScheduler with 'kill'

`scheduler`

`-kill`	Stopping a Running JobScheduler using 'kill'
`-kill`=*pid*	Stopping a Running JobScheduler with 'kill'
`-pid-file`=*dateiname*	

Example:

```
scheduler -kill -pid-file=/home/scheduler/scheduler.pid
```

Expand the Java Class Path

`scheduler`

`-expand-classpath`	Expand the Java Class Path

Show Version Number

`scheduler`

`-V`	Show Version Number

Options

`-config`=*file_name* Configuration file

Defines the Configuration File (page 6).

The "`-config=`" prefix before file names can be omitted.

This option specifies the configuration file or configuration directory. Should a configuration directory be specified, then the JobScheduler will expect to find the configuration file in the configuration directory, under the name `scheduler.xml`.

The `factory.ini` (section `[spooler]`, entry `config= ...`) setting is overwritten by this parameter.

`-log`=*file_name* scheduler.log file name

This setting causes the JobScheduler to write a detailed protocol. This protocol is intended for use in problem diagnosis. The file name should be fully specified here (i.e. as a full path).

A plus character (+) written directly before the file name causes an already existing protocol to be continued. Otherwise such a protocol will be overwritten.

Categories can be used to extend or restrict the log file. Category names are added (separated by spaces) before the file name, which is then preceded by a larger than (>) sign.

The list of categories can be found here.

Example:

```
log = c:/tmp/scheduler.log
log = scheduler.wait >scheduler.log
log = scheduler.wait com_server.* >scheduler.log
```

The `factory.ini` (section `[spooler]`, entry `log= …`) setting is overwritten by this parameter.

`-log-dir`=*directory* Protocol directory

The directory in which the JobScheduler writes log files.

`log_dir= *stderr` allows the JobScheduler to write log files to the standard output (`stderr`, normally the screen) .

The `factory.ini` (section `[spooler]`, entry `log_dir= …`) setting is overwritten by this parameter.

`-id`=*scheduler_id* JobScheduler identifier (id)

The JobScheduler only selects elements in the XML configuration whose `spooler_id` attributes are either empty or set to the value given here.

When the JobScheduler ID is not specified here, then the JobScheduler ignores the `spooler_id=` XML attribute and selects all the elements in the XML configuration.

See, for example, `<config>` (page 21).

The `factory.ini` (section `[spooler]`, entry `id= …`) setting is overwritten by this parameter.

`-cd`=*directory* Working Directory

Changes the Working Directory.

`-pid-file`=*File Name* File name for process ID

On Unix systems: the JobScheduler writes its process ID (PID) in this file. This file is deleted by the JobScheduler on stopping.

`-log-level`=*log_level* (Initial value: info) Limit protocol level

Defines the level with which protocol entries should be written. Every protocol entry is given one of the following categories: `error`, `warn`, `info`, `debug1` to `debug9` (`debug1` is the same as `debug`).

The `factory.ini` (section `[job]`, entry `log_level= info`) setting is overwritten by this parameter.

The `factory.ini` (section `[spooler]`, entry `log_level= info`) setting is overwritten by this parameter.

`-param`=*text* For free use

Free text. This parameter can be read using `spooler.param`.

The `factory.ini` (section `[spooler]`, entry `param= …`) setting is overwritten by this parameter.

`-include-path`=*directory* Directory path for <include>

The directory of the files which are to be included by the `<include>` element.

Environment variables (e.g. `$HOME`) are replaced by this attribute (see Settings which Allow Environment Variables to be Called (page 106)).

The `factory.ini` (section `[spooler]`, entry `include_path= …`) setting is overwritten by this parameter.

`<config include_path="…">` is overwritten by this parameter.

`-port`=*number* (Initial value: 0) HTTP, TCP and UDP ports for control commands for the JobScheduler

Combines the `tcp_port` and `udp_port` settings.

See also `-tcp-port` and `-udp-port`.

Note that if a port is blocked, the JobScheduler will attempt to access it for two minutes before terminating itself.

`<config port="…">` is overwritten by this parameter.

`-tcp-port`=*number* (Initial value: 0) Port for HTTP and TCP commands for the JobScheduler

The JobScheduler can accept commands via a TCP port whilst it is running. The number of this port is set here - depending on the operating system - with a number between 2048 and 65535. The default value is 4444.

The JobScheduler operates a HTTP/HTML server on the same port, enabling it to be reached using a web browser - e.g. via http://localhost:4444.

The JobScheduler does not respond to the `tcp_port=0` default setting either with TCP or HTTP protocols. This setting can therefore be used to block a JobScheduler from being accessed - for example via TCP.

See also `-port` .

`<config tcp_port="…">` is overwritten by this parameter.

`-time-zone`=*text* JobScheduler time zone

Specifies the time zone in which a job or order is to start. Time zones are to be specified as defined in the tz database. A List of time zones is available in the Joda API, which is used in JobScheduler for the time functions.

The JobScheduler uses its local time if a time zone is not specified.

Example:

```
<config time_zone="Europe/Berlin">
```

`<config time_zone="…">` is overwritten by this parameter.

`-udp-port`=*number* (Initial value: 0) Port for UDP commands for the JobScheduler

The JobScheduler can also accept UDP commands addressed to the port specified in this setting. Note that a UDP command must fit in a message and that the JobScheduler does not answer UDP commands.

The default value of `udp_port=0` does not allow the JobScheduler to open a UDP port.

See also `-port` .

`<config udp_port="…">` is overwritten by this parameter.

`-ip-address`=*ip_number* (Initial value: 0.0.0.0) The interface IP address for TCP and UDP

The IP address to which the TCP and UDP ports are bound. The JobScheduler can then only be reached by way of this address.

A host name can also be specified.

The default setting is 0.0.0.0, which stands for all IP addresses.

When another IP address as 127.0.0.1 or localhost is given, then the JobScheduler cannot be reached by way of localhost.

`<config ip_address="…">` is overwritten by this parameter.

`-reuse-port` Reuse of the TCP and UDP ports

This option should only be used in exceptional situations, as it switches off the test whether or not a port has been set free by the operating system.

Calls `setsockopt( socket, SOL_SOCKET, SO_REUSEADDR, true)` .

This option should only be used in an emergency, should a port be permanently blocked by the operating system although it is no longer used by an application. This can happen with Windows XP, when an application is terminated by a debugger and UNIX can take up to a minute to release a port.

In some situations the JobScheduler cannot be reached with `-reuse-port` via TCP or UDP, when the port is being used by other applications. The use of `-reuse-port` is not recommended.

`-cmd`=*xmlcommand* Immediately executed commands

The JobScheduler executes xml commands such as `<start_job>` immediately on starting.

`-ini`=*file name* Alternative factory.ini file

Specifies the Path/ Name for an alternative `factory.ini` (page 94) file.

When references in this documentation are made to the `factory.ini` file, it should be clear that the value given in this option is meant.

See also `Spooler.ini_path`

`-sos.ini`=*file name* Alternative sos.ini file

Specifies the Path/ Name for an alternative `sos.ini` (page 104) global configuration file. Note that this file contains the JobScheduler license key.

`-program-file`=*file name* JobScheduler file name

This option is used when the JobScheduler is called from a Java class file (for example, when debugging) and jobs are to be started in their own processes.

`-service` Start as a daemon

The JobScheduler runs in the background under Unix, therefore no output can be written to the terminal; output for stdout und stderr should be redirected to a file in order to prevent loss.

`-validate-xml` Validate XML Ddocuments against an embedded schema

`-validate-xml-` switches the validation off (should the schema have errors).

`-use-xml-schema`=*dateiname* Schema for validating the JobScheduler configuration

JobScheduler use a build-in schema for validating the configuration files. With this option you can overwrite it.

`-env`=*name*=*value* Set Environment Variables

`-env=NAME=VALUE` sets the NAME environment variable.

Messages

`[ERROR]`	`SCHEDULER-318`	Option -env=NAME=VALUE: missing '=' between name and value: -env=""

`-exclusive` Starts the JobScheduler for Exclusive Service

Requires use of a database - see `factory.ini` (section `[spooler]`, entry `db= ...`) (page 98).

More than one JobScheduler with the same Scheduler-Id (`-id`) and the same database can be started with `-exclusive`. In this case, then only one JobScheduler will become active. It is only after this active

JobScheduler fails or is stopped (`<terminate shutdown="no">`), that another JobScheduler will become active and take over operation. The takeover takes approximately one minute.

Warning: All participating JobSchedulers must be started `-exclusive` or `-distributed-orders`. When one JobScheduler is started with and one without this setting, then two JobSchedulers will run simultaneously, which can lead to unexpected results.

See also `-backup`.

Messages

[ERROR]	SCHEDULER-357	This is a member of a cluster (option -exclusive or -distributed-orders), and therefore needs role 'scheduler'
[ERROR]	SCHEDULER-358	
[ERROR]	SCHEDULER-359	For an exclusive or distributed Scheduler, the database is not supported
[ERROR]	SCHEDULER-362	Scheduler is aborting because it has become inactive
[ERROR]	SCHEDULER-365	Illegal character in -id=
[ERROR]	SCHEDULER-367	Scheduler is aborting because it has lost its exclusivity
[ERROR]	SCHEDULER-371	DATABASE INTEGRITY IS BROKEN
[ERROR]	SCHEDULER-372	Exclusivity has been stolen by JobScheduler member ''
[ERROR]	SCHEDULER-377	After own late heart beat, JobScheduler member '' has taken exclusiveness
[ERROR]	SCHEDULER-381	Scheduler is not yet active and cannot execute the operation
[ERROR]	SCHEDULER-386	Last heart beat was time, seconds ago. Something is delaying JobScheduler execution, the JobScheduler is aborted immediately
[warn]	SCHEDULER-823	Dead JobScheduler member 'cluster_member_id' has resurrected
[warn]	SCHEDULER-827	Own heart beat is late: next_heart_beat has been announced for (this is seconds late)
[warn]	SCHEDULER-836	Deactivating that dead Scheduler
[warn]	SCHEDULER-837	Taking exclusiveness from that Scheduler
[warn]	SCHEDULER-994	No heart beat for seconds (time), expecting heart beat within seconds
[warn]	SCHEDULER-996	No heart beat for seconds (time) - JobScheduler seems to be dead
[warn]	SCHEDULER-997	Making up an extra heart beat
[info]	SCHEDULER-805	No JobScheduler is active
[info]	SCHEDULER-807	Using database product
[info]	SCHEDULER-811	Executing command read from database:
[info]	SCHEDULER-814	Inactive JobScheduler '' has the higher backup precedence (http_url)
[info]	SCHEDULER-819	Scheduler becomes active
[info]	SCHEDULER-820	Watching heart beat of that Scheduler
[info]	SCHEDULER-821	Scheduler member 'cluster_member_id' seems to be active, claiming its last heart beat was s ago (pid=, url)
[info]	SCHEDULER-822	Scheduler member 'cluster_member_id' seems to be exclusive, claiming its last heart beat was s ago (pid=, url)
[info]	SCHEDULER-825	No exclusive JobScheduler is running
[info]	SCHEDULER-826	That JobScheduler has terminated

[info]	SCHEDULER-831	Waiting s for start of non-backup Scheduler
[info]	SCHEDULER-832	This is a backup Scheduler, waiting for a non-backup Scheduler
[info]	SCHEDULER-833	Watching heart beat of that exclusive Scheduler, standing by to take over operation
[info]	SCHEDULER-834	Active JobScheduler has terminated properly
[info]	SCHEDULER-835	This JobScheduler is becoming exclusive now
[info]	SCHEDULER-838	. heart beat detected
[info]	SCHEDULER-995	No heart beat for seconds (time), ignored for seconds because of recent database reconnect

-backup Starts a JobScheduler as a Backup Scheduler

Only possible in combination with `-exclusive`.

A backup JobScheduler only takes over operation - it cannot start a new operation. This means that after `<terminate continue_exclusive_operation="no">` the backup JobScheduler does not start, but waits for another JobScheduler to start operation.

`-backup` is allocated its own service name when used together with `-install-service` and `-remove-service`. This means that the backup JobScheduler can be run on the same computer as the active JobScheduler as its own service.

The name of the main log file (page 174) contains the "`_backup`" suffix.

-backup-precedence=*integer* (Initial value: 1) Priority Amongst Backup JobSchedulers

Only possible in combination with `-exclusive`.

When more than one inactive backup JobSchedulers are available to replace a failed JobScheduler, (`-exclusive`), then operation is taken over by the JobScheduler allocated the lowest `-backup-precedence` value.

Note that it is possible for another JobScheduler as that dictated by the highest backup-precedence to take over operation, should the backup JobSchedulers run on different computers and should the clocks of these computers not be synchronized,

The default value is 1, when `-backup` is set, otherwise it is 0.

-distributed-orders Distributed Orders

Requires use of a database (). More than one JobSchedulers can share the execution of orders, when they are all started under the same Scheduler-Id (`-id`), use the same database and the same configuration.

`<job_chain distributed="yes">` allows a job chain to be used for distributed operation.

Messages

[ERROR]	SCHEDULER-357	This is a member of a cluster (option -exclusive or -distributed-orders), and therefore needs role 'scheduler'
[ERROR]	SCHEDULER-358	
[ERROR]	SCHEDULER-361	No database
[ERROR]	SCHEDULER-370	Operation can only be performed on a distributed orders Scheduler
[ERROR]	SCHEDULER-373	UNEXPECTED DEACTIVATION BY JOBSCHEDULER MEMBER

[ERROR]	SCHEDULER-375	Order is distributed and therefore does not support operation ''
[ERROR]	SCHEDULER-378	After own late heart beat, this JobScheduler has been deactivated and the occupied orders have been freed by JobScheduler member ''
[ERROR]	SCHEDULER-379	order is occupied by JobScheduler member ''
[ERROR]	SCHEDULER-380	job_chain orders_recoverable="no" cannot be combined with distributed="yes", in
[ERROR]	SCHEDULER-384	job_chain is distributed and therefore does not support operation ''
[ERROR]	SCHEDULER-385	Deletion of order in database has failed
[warn]	SCHEDULER-812	Just before processing, order record in database has been occupied or removed
[warn]	SCHEDULER-816	Unable to release occupation in database
[warn]	SCHEDULER-817	Missing order record in database
[info]	SCHEDULER-813	Order is occupied by JobScheduler ''
[info]	SCHEDULER-815	Task should end but it has just been started with an order attached, so one step will be done
[info]	SCHEDULER-829	Releasing occupied order job_chain:id
[info]	SCHEDULER-830	Because all JobScheduler tasks have been killed, the order in database has not been updated. Only the occupation has been released
[info]	SCHEDULER-879	Deactivating old cluster member with same ID

-configuration-directory Configuration directory

The default directory is the `live` directory, which is specified in the (`-config`).

The JobScheduler looks in this directory for jobs, job chains, permanent orders, process classes and locks.

-java-classpath=*file_names* Java class path for JobScheduler process

The Java `CLASS_PATH` setting is made here. This is a list of paths - on Windows Systems these paths are separated by semi-colons (;), on Unix systems by colons (:).

Note that jokers can be used in these paths. The JobScheduler then replaces these jokers with the respective file names. i.e. those existing in the file system.

- On Windows systems, the `*` and `?` characters may be used after the last directory separator.
- On Unix systems the `[` and `]` characters can also be used. Jokers can be used in every folder name of a path (as in the `csh` shell).
- Linux (GNU) also recognizes the `{,}` characters, when used in the following syntax:

  "*xxx*`{` *alternative1* , *alternative2* , ...`}` *xxx*", e.g. `/dir/sos.{hostware,mail.spooler}.jar`.

The following points apply to every path in the `class_path`:

- A path without a joker will be handed over to Java unchanged.
- A path containing a joker will be converted to the existing path before being handed over.
- Should no path corresponding to a joker be found then the path will be ignored.
- A path will also be ignored should it refer to a directory which cannot be read.
- Environment variables (e.g. `${HOME}`) will be replaced before a path is converted.

These settings are generally made in the `sos.ini` (section `[java]`, entry `class_path= ...`) file.

Example:

```
scheduler -java-classpath = s:\prod\bind\sos.*.jar;c:\jar\my.jar;c:\jar\*
```

Environment variables (e.g. `$HOME`) are replaced by this attribute (see Settings which Allow Environment Variables to be Called (page 106)).

`-java_options`=*text* Java options for JobScheduler process

This setting specifies the directory in which the HostJava is installed - e.g. `-Djava.library.path=`.

These options are passed to the Java Virtual Machine.

These settings are generally made using `sos.ini` (section `[java]`, entry `options= …`).

Environment variables (e.g. `$HOME`) are replaced by this attribute (see Settings which Allow Environment Variables to be Called (page 106)).

`-job-java-classpath`=*file_names* Java class path for Jobs

See `-java-classpath`

Environment variables (e.g. `$HOME`) are replaced by this attribute (see Settings which Allow Environment Variables to be Called (page 106)).

`-job_java_options`=*text* Java options for Jobs

See `-java-options`

See also `<job java-options="…">` (page 42).

Environment variables (e.g. `$HOME`) are replaced by this attribute (see Settings which Allow Environment Variables to be Called (page 106)).

`-install-service`=*name* Install as a Windows service

The JobScheduler is not started by this option but installed as a Windows service. In this case the following command line options may be specified:

`-cd`
`-config`

`-id`

`-ini`

`-log`

`-pid-file`

`-sos.ini`

`-env`

Note that `-install-service=` *name* is the same as `-install-service -service-name=` *name*.

`-remove-service`=*name* Remove a Windows service

Removes a service which was previously installed using `-install-service`.

`-remove-service=` *name* is the same as `-remove-service -service-name=` *name*.

`-service-name`=*name* (Initial value: sos_scheduler) Windows service internal name

Only used together with `-install-service` or `-remove-service`. When this option is missing but `-id=`is specified, then the JobScheduler uses the name `sos_scheduler_` *scheduler_id*. If `-backup` has been set, then this option adds `_backup`.

`-service-display`=*text* Windows service name

Only used in conjunction with the `-install-service`. This name is shown by the Windows service controller. It may contain spaces. Should this option not be specified, the JobScheduler creates a name from the service name (see `-service-name=`).

`-service-descr`=*text* Windows service description

Only used together with the `-install-service`.

`-need-service`=*name* Service required by the JobScheduler (Windows only)

This option specifies a service such as a database server which must be running before Windows starts the JobScheduler. This option must be used in conjunction with the `-install-service`.

This option can be repeated should the JobScheduler require a number of services.

`-scheduler`=*host:port* The JobScheduler TCP address

The name (or IP number) and port number of the JobScheduler being addressed.

`-log`=*file_name* scheduler.log file name

This setting causes the JobScheduler to write a detailed protocol. This protocol is intended for use in problem diagnosis. The file name should be fully specified here (i.e. as a full path).

A plus character (+) written directly before the file name causes an already existing protocol to be continued. Otherwise such a protocol will be overwritten.

Categories can be used to extend or restrict the log file. Category names are added (separated by spaces) before the file name, which is then preceded by a larger than (>) sign.

The list of categories can be found here.

Example:

```
log = c:/tmp/scheduler.log
log = scheduler.wait >scheduler.log
log = scheduler.wait com_server.* >scheduler.log
```

The `factory.ini` (section `[spooler]`, entry `log= …`) setting is overwritten by this parameter.

`-process-class`=*name*

The name of the temporary job process class -, see `<job process_class="…">` (page 42).

`-language`=*script_language* (Initial value: shell) The Job Script language

Script language, see `<script language="…">` (page 77).

`-at`=*yyyy-mm-dd HH:MM* (Initial value: now) Start time

Start time - in the form "yyyy-mm-dd HH:MM[:SS]". See `<at at="…">` (page 18).

A job will not be started after its start time has been passed.

The JobScheduler uses the current time from the computer on which it is running when, for example, starting jobs.

`-at=now` is the default setting and causes a job to be started immediately.

`-scheduler`

`-log`

`-job-chain`=*name* Job Chain

The name of the job chain on which an order is to be run - see `<add_order job_chain="…">` (page 13).

`-order-id`=*id* The Order ID

The order identifier - see `<add_order id="…">` (page 13).

`-title`=*text* The Order Title

An order can be given a title. See `<add_order title="…">` (page 13).

`-tcp-port`

`-send-cmd`=*xmlcommand* Sending a command to another JobScheduler

This option does not start a JobScheduler but sends an XML command to another JobScheduler on the same computer which has the same `-tcp-port=` option. This can be, for example, `<terminate>` : `-send-cmd="<terminate/>"`

This command is sent to the IP address specified in the `<config ip_address="…">` attribute or, should this not have been set, to 127.0.0.1.

`-kill` Stopping a Running JobScheduler using 'kill'

Stops a JobScheduler whose process is specified in the `-pid-file` file. The JobScheduler is stopped using `kill -SIGKILL`.

`-kill`=*pid* Stopping a Running JobScheduler with 'kill'

Stops the process with the specified PID using `kill -SIGKILL`.

`-pid-file`=*dateiname*

`-expand-classpath` Expand the Java Class Path

Expands the Java Class Path, which is specified as a parameter. This means that jokers are processed as `sos.ini` (section `[java]`, entry `class_path= …`).

The JobScheduler writes the results to `stdout`. Unless otherwise specified in an option, the JobScheduler will then terminate itself, without any other output.

Example:

```
export CLASSPATH="`scheduler -expand-classpath='/opt/java/lib/*.jar'`"
```

`-V` Show Version Number

Causes the JobScheduler to output its version number and date when starting.

1.8 JOE - JobScheduler Object Editor

1.8.1 Configuration and Documentation of JobScheduler objects

The JobScheduler reads its configuration from XML files, which can be created and edited using JOE. JOE provides a graphical user interface with forms for all elements of the JobScheduler configuration. Further user assistance in the form of tool-tips and wizards is provided.

Alternatively, the XML configuration can be maintained using any text editor.

JOE creates or modifies the XML elements in the JobScheduler configuration, in accordance with the entries made in its forms. When the changes made are saved, the JobScheduler configuration file - which is usually called *scheduler.xml*, is rewritten. JOE validates all entries made with the current JobScheduler XML schema - errors in entries will be detected immediately.

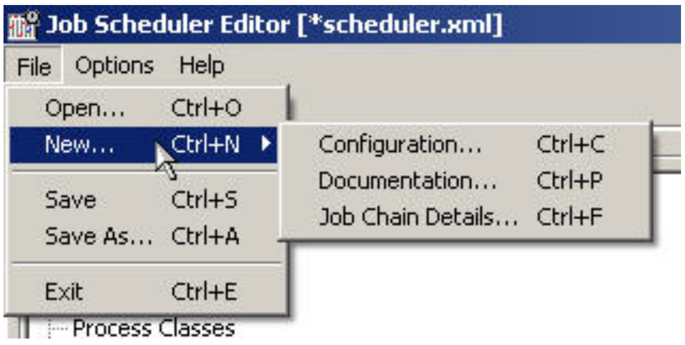
JOE can also be used to document jobs for the JobScheduler - it generates the documentation in an XML format, which can be shown with navigations elements in a web browser in HTML format, using an appropriate style sheet.

1.8.1.1 Starting JOE

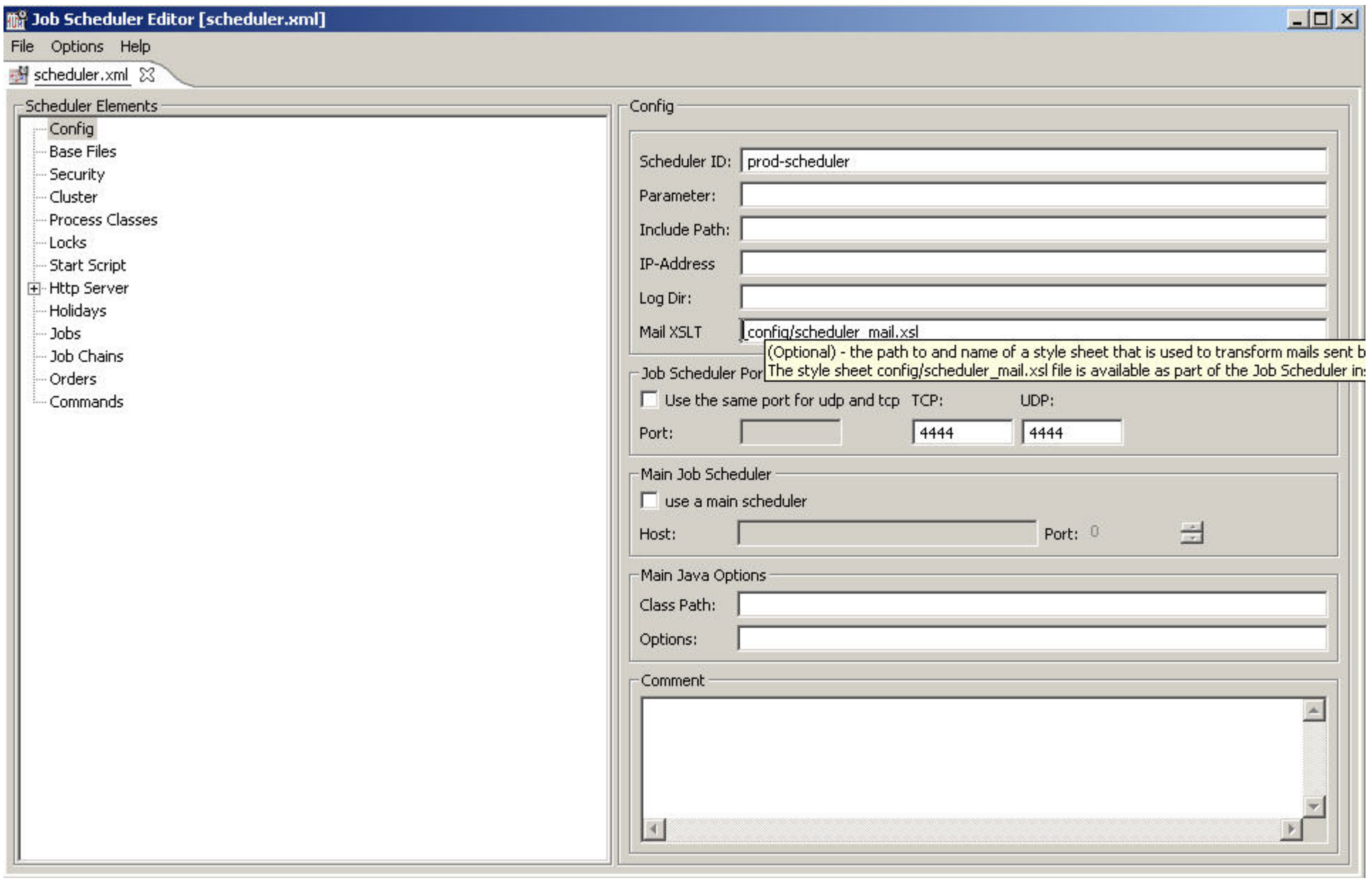
JOE belongs to the JobScheduler installation. The JobScheduler installation directory contains the *bin/jobeditor.cmd* file, which is used to start JOE. Here, the INSTALL_PATH parameter should be set to the JobScheduler installation directory path.

1.8.1.2 Creating the JobScheduler Configuration

In order to create a new configuration, the *File/New/Configuration* menu item should be selected:



JOE then opens its main window containing a tree view showing a series of elements such as *Process Classes*, *Jobs*, *Job Chains* etc.


Job Scheduler Editor [scheduler.xml]

Scheduler Elements: Config, Base Files, Security, Cluster, Process Classes, Locks, Start Script, Http Server, Holidays, Jobs, Job Chains, Orders, Commands

Config

Scheduler ID:	prod-scheduler
Parameter:	
Include Path:	
IP-Address	
Log Dir:	
Mail XSLT	config/scheduler_mail.xsl

(Optional) - the path to and name of a style sheet that is used to transform mails sent b
The style sheet config/scheduler_mail.xsl file is available as part of the Job Scheduler in:

Job Scheduler Por
Use the same port for udp and tcp TCP: UDP:
Port: TCP 4444 UDP 4444

Main Job Scheduler
use a main scheduler
Host: Port: 0

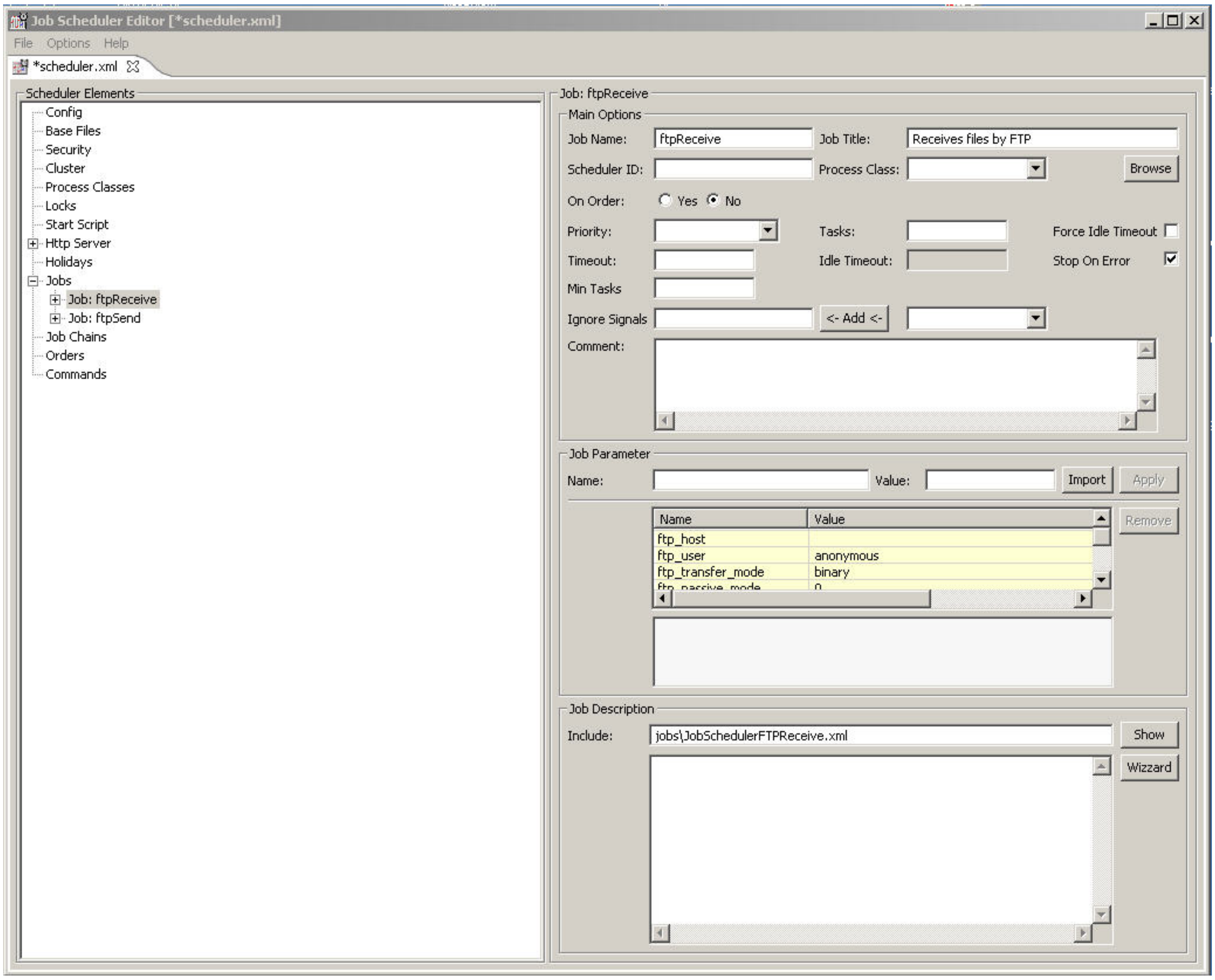
Main Java Options
Class Path:
Options:

Comment

Tool-tips are available for all form fields and control elements of JOE. In addition, the **F1** button can be used to start the JobScheduler documentation, which is contextual and will open at the relevant object that is currently being worked on.

For example:

When the job configuration form is opened:



Scheduler Elements

- Config
- Base Files
- Security
- Cluster
- Process Classes
- Locks
- Start Script
- Http Server
- Holidays
- Jobs
  - Job: ftpReceive
  - Job: ftpSend
- Job Chains
- Orders
- Commands

Job: ftpReceive

Main Options

Job Name: ftpReceive
Job Title: Receives files by FTP
Scheduler ID:
Process Class:
On Order: Yes [✓ No]
Priority:
Tasks:
Force Idle Timeout
Timeout:
Idle Timeout:
Stop On Error ✓
Min Tasks
Ignore Signals
Comment:

Job Parameter

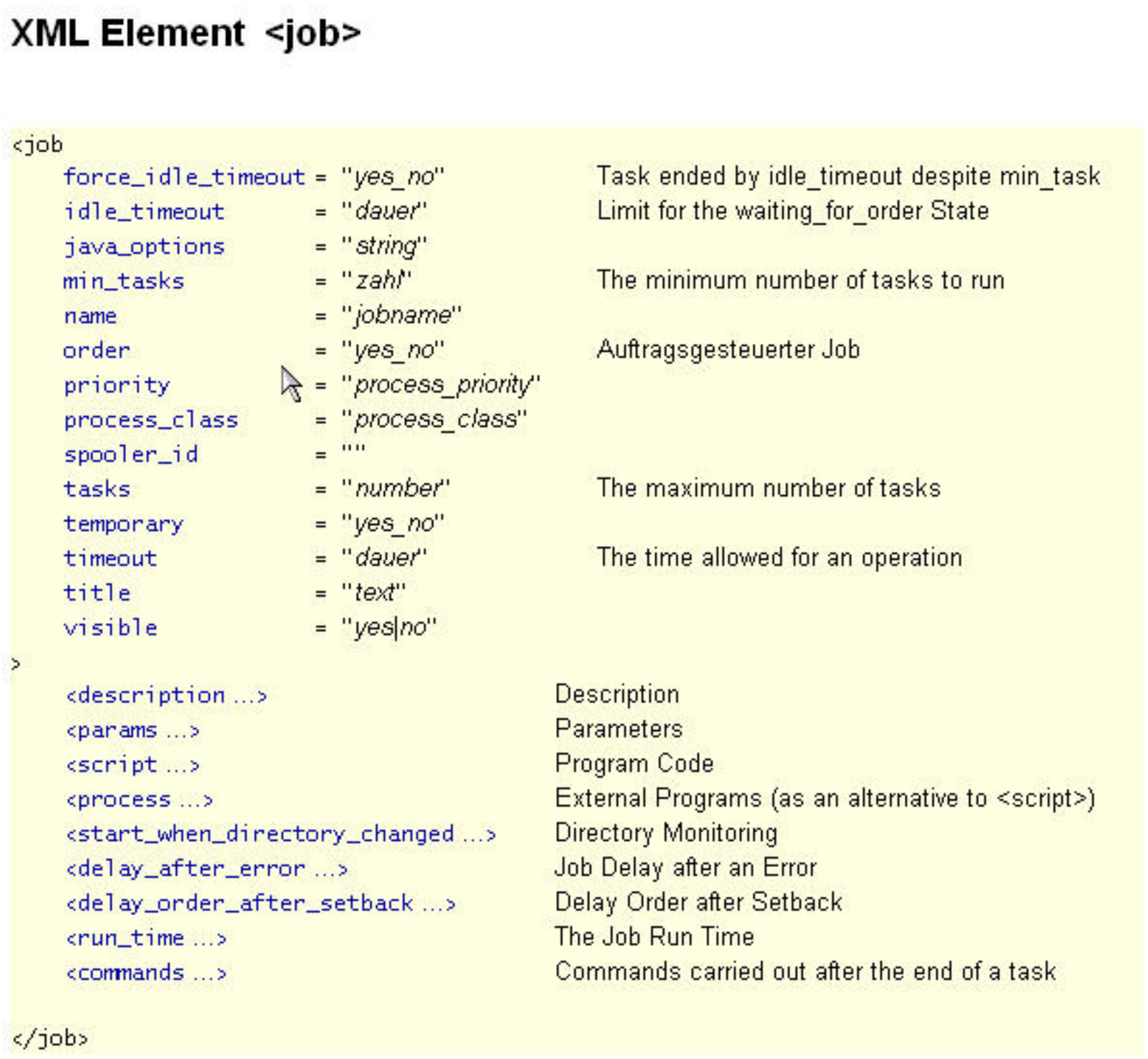
Name:
Value:

Name	Value
ftp_host	
ftp_user	anonymous
ftp_transfer_mode	binary
ftp_passive_mode	0

Job Description

Include: jobs\JobSchedulerFTPReceive.xml

F1 causes the documentation to open at the *<job>* XML element:

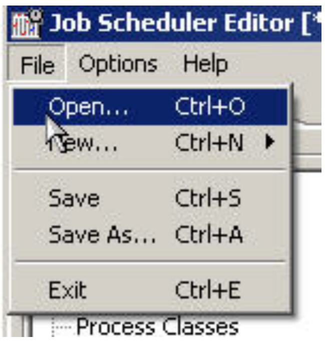


XML Element <job>

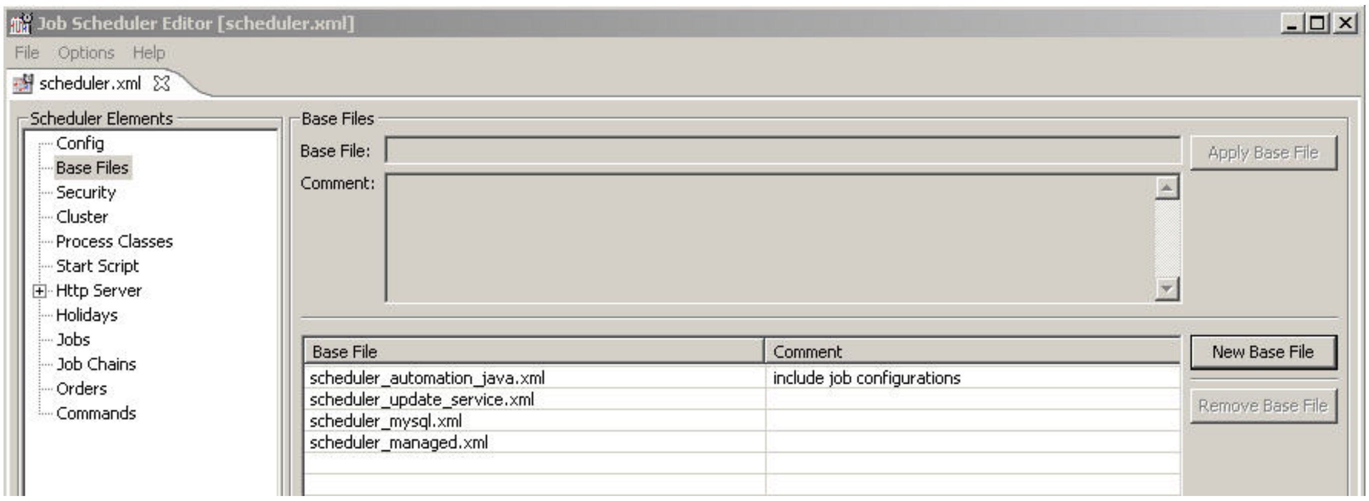
```
<job
    force_idle_timeout = "yes_no"          Task ended by idle_timeout despite min_task
    idle_timeout       = "dauer"           Limit for the waiting_for_order State
    java_options       = "string"
    min_tasks          = "zahl"            The minimum number of tasks to run
    name               = "jobname"
    order              = "yes_no"          Auftragsgesteuerter Job
    priority           = "process_priority"
    process_class      = "process_class"
    spooler_id         = ""
    tasks              = "number"          The maximum number of tasks
    temporary          = "yes_no"
    timeout            = "dauer"           The time allowed for an operation
    title              = "text"
    visible            = "yes|no"
>
    <description ...>                   Description
    <params ...>                        Parameters
    <script ...>                        Program Code
    <process ...>                       External Programs (as an alternative to <script>)
    <start_when_directory_changed ...>  Directory Monitoring
    <delay_after_error ...>             Job Delay after an Error
    <delay_order_after_setback ...>     Delay Order after Setback
    <run_time ...>                      The Job Run Time
    <commands ...>                      Commands carried out after the end of a task

</job>
```

1.8.1.3 Opening an Existing Configuration

Existing JobScheduler configuration files are opened using the *File/Open* menu items:



Example - the *scheduler.xml* file from the JobScheduler installation, showing the base file configuration:



Base File	Comment
scheduler_automation_java.xml	include job configurations
scheduler_update_service.xml	
scheduler_mysql.xml	
scheduler_managed.xml	

1.8.1.4 Recommended Procedure

- Open the *scheduler.xml* file in the JobScheduler installation.
- Enter the computer and/or network in the *Security* form, which is to execute commands via TCP and UDP.
- Enter the jobs the JobScheduler should execute, together with the relevant job parameters in the *Jobs* form. For each job entered, JOE creates a new branch in the job tree view. Each job has additional forms, in which further properties such as executable program code and start times are entered. A wizard is available for creating a job by copying a standard job.
  Note: when jobs are defined in XML files which themselves are included using <base> then the included XML file will be opened in JOE together with the job definition.
- When Jobs are to be executed within a job chain, then the job chain should be entered using the *Job Chains* form. Each node in a job chain must be specified individually and a job allocated to each node.

  Note: when job chains are to be defined in XML files, which are included using <base>, then the XML file with the job chain definitions is to be opened.
- Note that when order controlled jobs are to be used in job chains, then the way in which the orders are created must be specified:

  Orders can be created using directory monitoring - this is done by specifying file orders in the *Job Chains* form.

  Orders can also be directly created using the <add_order> command - this type of order is defined in the *Orders* form. Using the information entered in this form, JOE generates an <add_order> element, which can be shown using the *Commands* form.

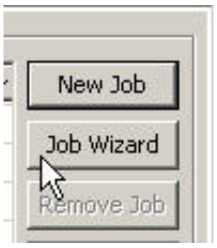
1.8.1.5 Job Wizard

The Job Wizard is used to create a new job on the basis of an already documented Template Job.

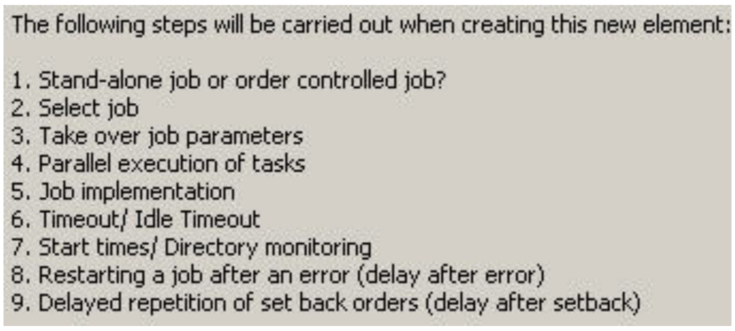
Existing job are available from a selection of documented template jobs that are delivered with the JobScheduler. These jobs can be found in the */jobs* directory in the JobScheduler installation directory.

The Job Wizard reads out this */jobs* directory and makes all the jobs found there available for import. An imported job has all the features of the original, in particular the job-language. Job parameters are also be taken over but can be modified in JOE. Further settings such as the number of tasks, the start times, time-out settings, can then be changed one after the other. Each step in the Wizard is supported by explanatory text

To start the Job Wizard, *Job Wizard* is simply selected in the *Jobs* form as shown:



The following steps are followed in the wizard in creating a new job:



The following steps will be carried out when creating this new element:

1. Stand-alone job or order controlled job?
2. Select job
3. Take over job parameters
4. Parallel execution of tasks
5. Job implementation
6. Timeout/ Idle Timeout
7. Start times/ Directory monitoring
8. Restarting a job after an error (delay after error)
9. Delayed repetition of set back orders (delay after setback)

1.8.1.6 Creating Job Documentation

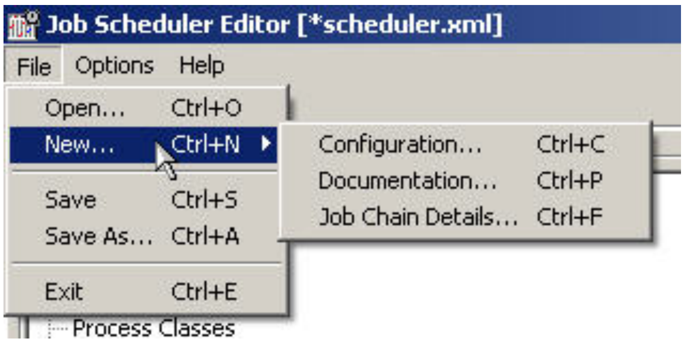
JOE can be used to create a documentation for JobScheduler jobs in a predefined format.

The documentation is generally written by the job developer and serves users/deployers or other developers configuring jobs for particular job implementations.

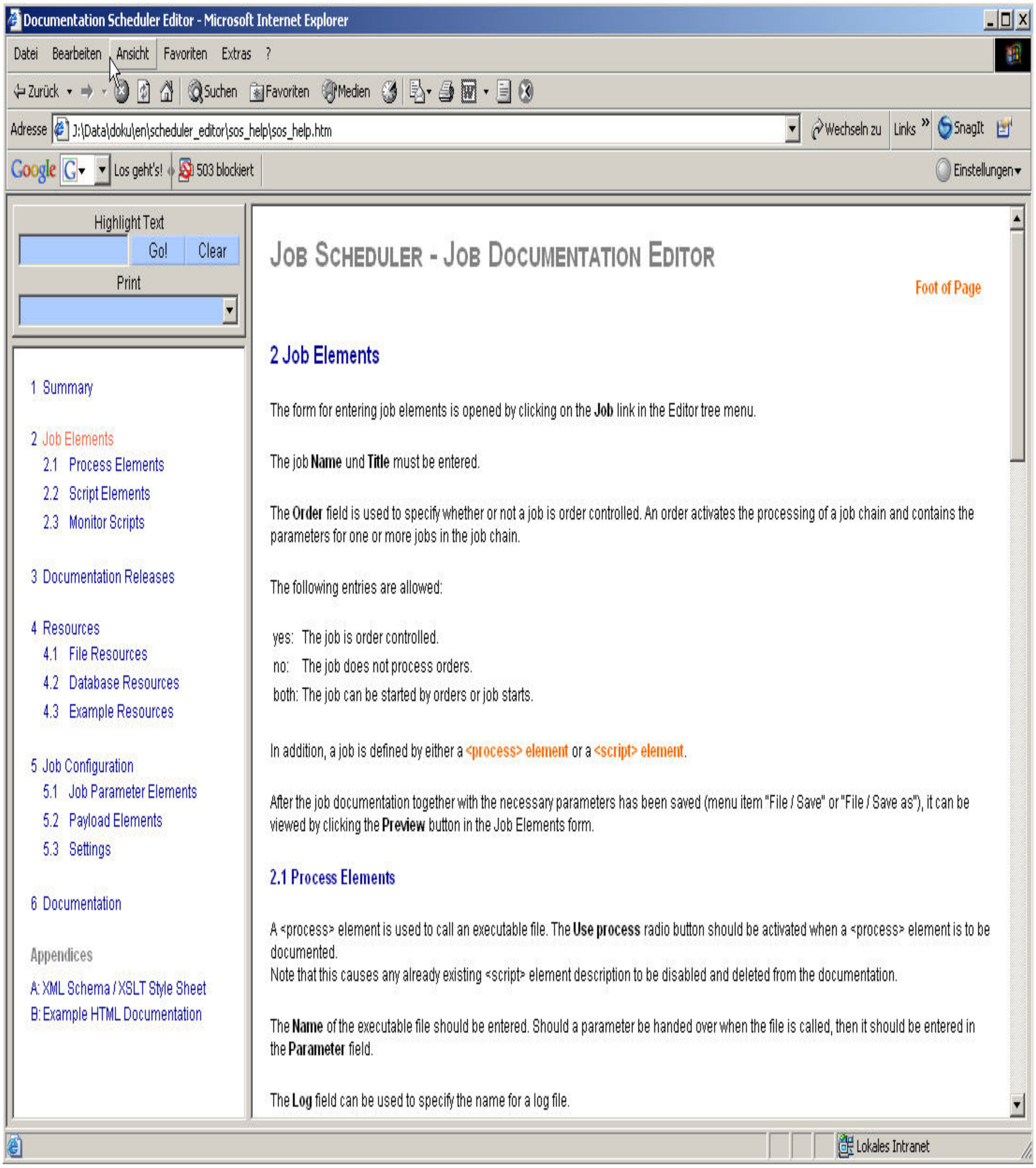
JOE allows the job documentation file to be created using forms, without knowledge of the XML format. The *scheduler_job_documentation.xsl* style sheet allows users to view the documentation in HTML format using a web browser.

A documented job is saved in the JobScheduler */jobs* directory, and it can thereby be used by the Job Wizard as a prototype for further jobs.

Select the *File/New/Documentation* item:



A contextual help is available for each form of JOE and can be opened using **F1**:


Documentation Scheduler Editor - Microsoft Internet Explorer
Datei Bearbeiten Ansicht Favoriten Extras ?
Zurück Suchen Favoriten Medien
Adresse J:\Data\doku\en\scheduler_editor\sos_help\sos_help.htm
Wechseln zu Links SnagIt
Google Los geht's! 503 blockiert Einstellungen
Highlight Text
Go! Clear
Print
Lokales Intranet

1 Summary
2 Job Elements
2.1 Process Elements
2.2 Script Elements
2.3 Monitor Scripts
3 Documentation Releases
4 Resources
4.1 File Resources
4.2 Database Resources
4.3 Example Resources
5 Job Configuration
5.1 Job Parameter Elements
5.2 Payload Elements
5.3 Settings
6 Documentation
Appendices
A: XML Schema / XSLT Style Sheet
B: Example HTML Documentation

JOB SCHEDULER - JOB DOCUMENTATION EDITOR

Foot of Page

2 Job Elements

The form for entering job elements is opened by clicking on the **Job** link in the Editor tree menu.

The job **Name** und **Title** must be entered.

The **Order** field is used to specify whether or not a job is order controlled. An order activates the processing of a job chain and contains the parameters for one or more jobs in the job chain.

The following entries are allowed:

yes:	The job is order controlled.
no:	The job does not process orders.
both:	The job can be started by orders or job starts.

In addition, a job is defined by either a <process> element or a <script> element.

After the job documentation together with the necessary parameters has been saved (menu item "File / Save" or "File / Save as"), it can be viewed by clicking the **Preview** button in the Job Elements form.

2.1 Process Elements

A <process> element is used to call an executable file. The **Use process** radio button should be activated when a <process> element is to be documented.
Note that this causes any already existing <script> element description to be disabled and deleted from the documentation.

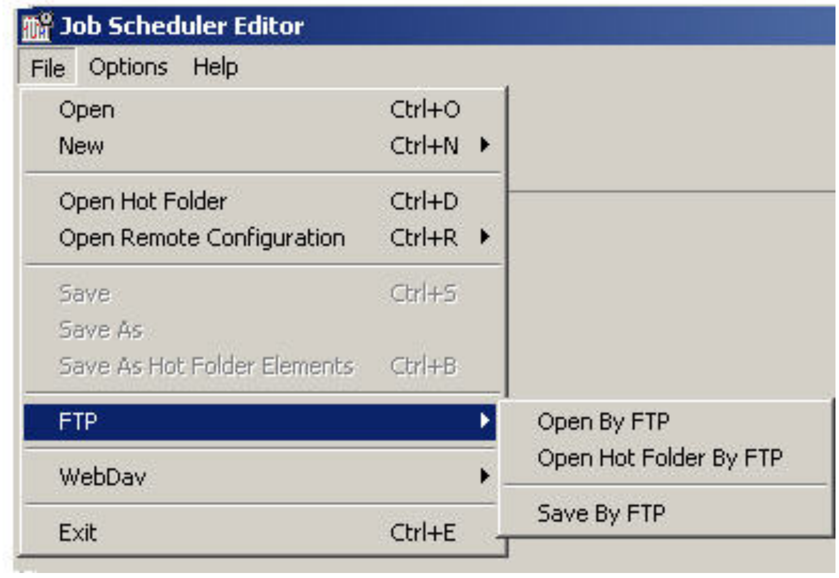
The **Name** of the executable file should be entered. Should a parameter be handed over when the file is called, then it should be entered in the **Parameter** field.

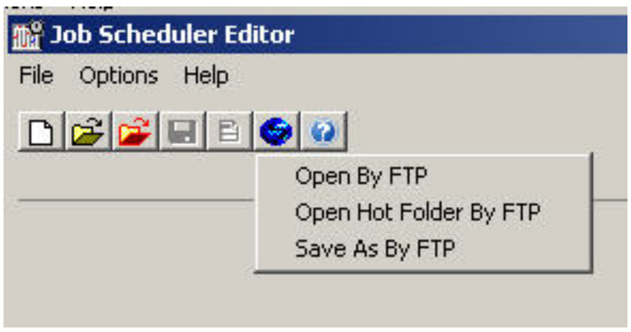
The **Log** field can be used to specify the name for a log file.

1.8.1.7 Remote Connections

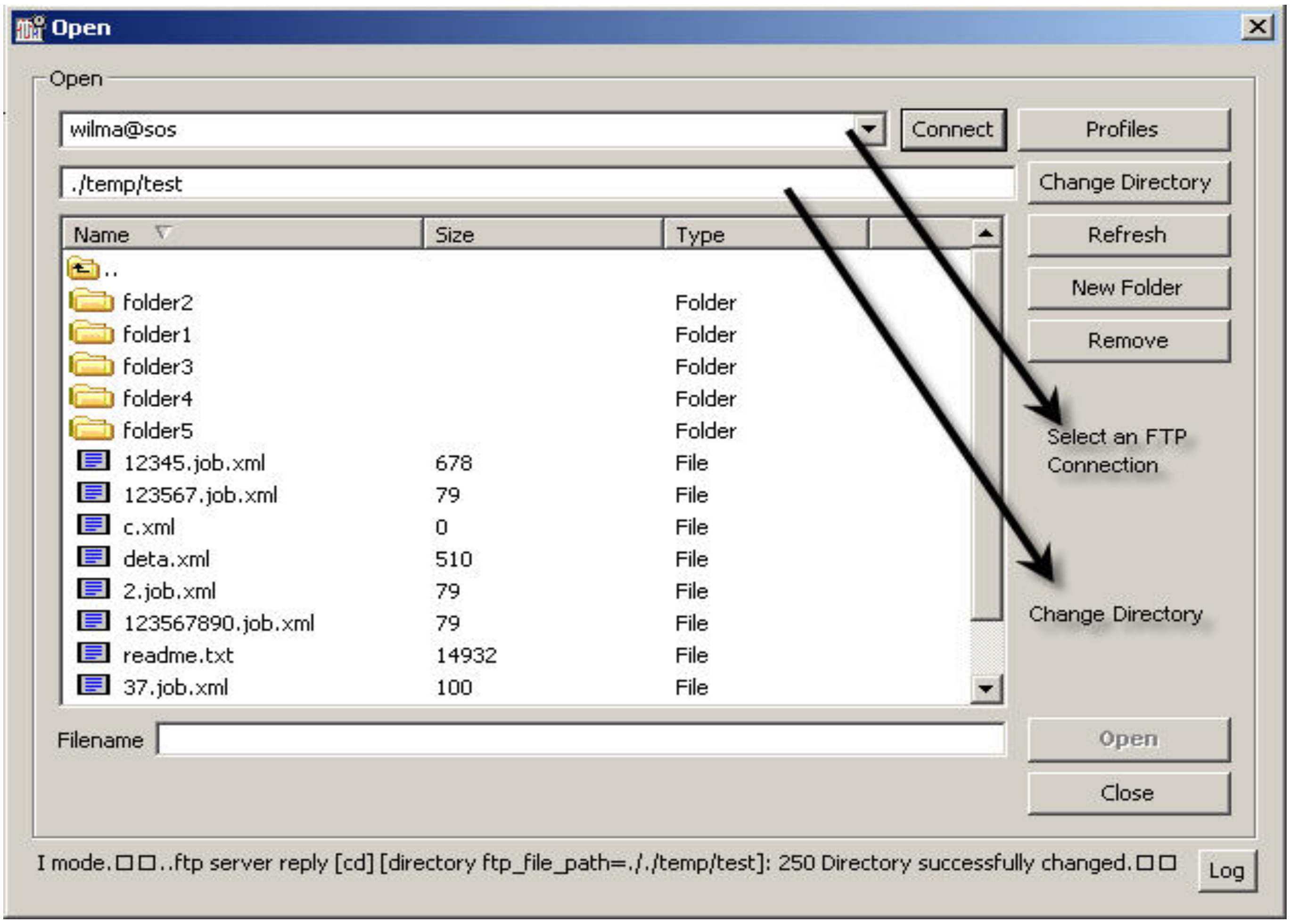
1.8.1.7.1 FTP/SFTP

The JobScheduler can open and save XML configuration files per FTP (File Transfer Protocol) and SFTP (FTP over SSH). The FTP dialog can either be opened by way of the File -> FTP menu entries or by clicking on the FTP toolbar icon:





This opens the dialog in which the profile (i.e. FTP connection) to be opened is selected, followed by the file to be opened.



wilma@sos

./temp/test

Name	Size	Type
..		
folder2		Folder
folder1		Folder
folder3		Folder
folder4		Folder
folder5		Folder
12345.job.xml	678	File
123567.job.xml	79	File
c.xml	0	File
deta.xml	510	File
2.job.xml	79	File
123567890.job.xml	79	File
readme.txt	14932	File
37.job.xml	100	File

```
I mode.☐☐..ftp server reply [cd] [directory ftp_file_path=./../temp/test]: 250 Directory successfully changed.☐☐
```

1.8.1.7.1.1 Control Elements

1.8.1.7.1.1.1 Select FTP Connection

FTP connections that have already been configured can be selected from the drop-down list.

1.8.1.7.1.2 Change Directory

This text box is used to show the pre-selected directory on the FTP server. This directory will be opened in the dialog as soon as the connection with the FTP server is made. When navigating on the server, this text box is used to show the current directory open. It can also be used for direct navigation. After entering the desired address in this box and clicking on the "Change Directory" button, the FTP server will directly go to the desired folder.

1.8.1.7.1.3 Refresh

Actualises all directory and file information from the FTP server.

1.8.1.7.1.4 New Folder

Creates a new folder on the FTP server.

1.8.1.7.1.5 Remove

Deletes the file or folder selected from the FTP server.

1.8.1.7.1.6 Open

If the FTP dialog has been opened using the "Open by FTP" menu entry, then the "Open" button will cause the XML configuration file selected to be downloaded from the FTP server and opened in JOE.

If the FTP dialog has been opened using the "Open Hot Folder by FTP" menu entry, then only directories will be shown in the dialog. Selection of a hot folder will then cause all the XML configuration files within that folder to be downloaded from the FTP server and opened in JOE.

1.8.1.7.1.7 Close

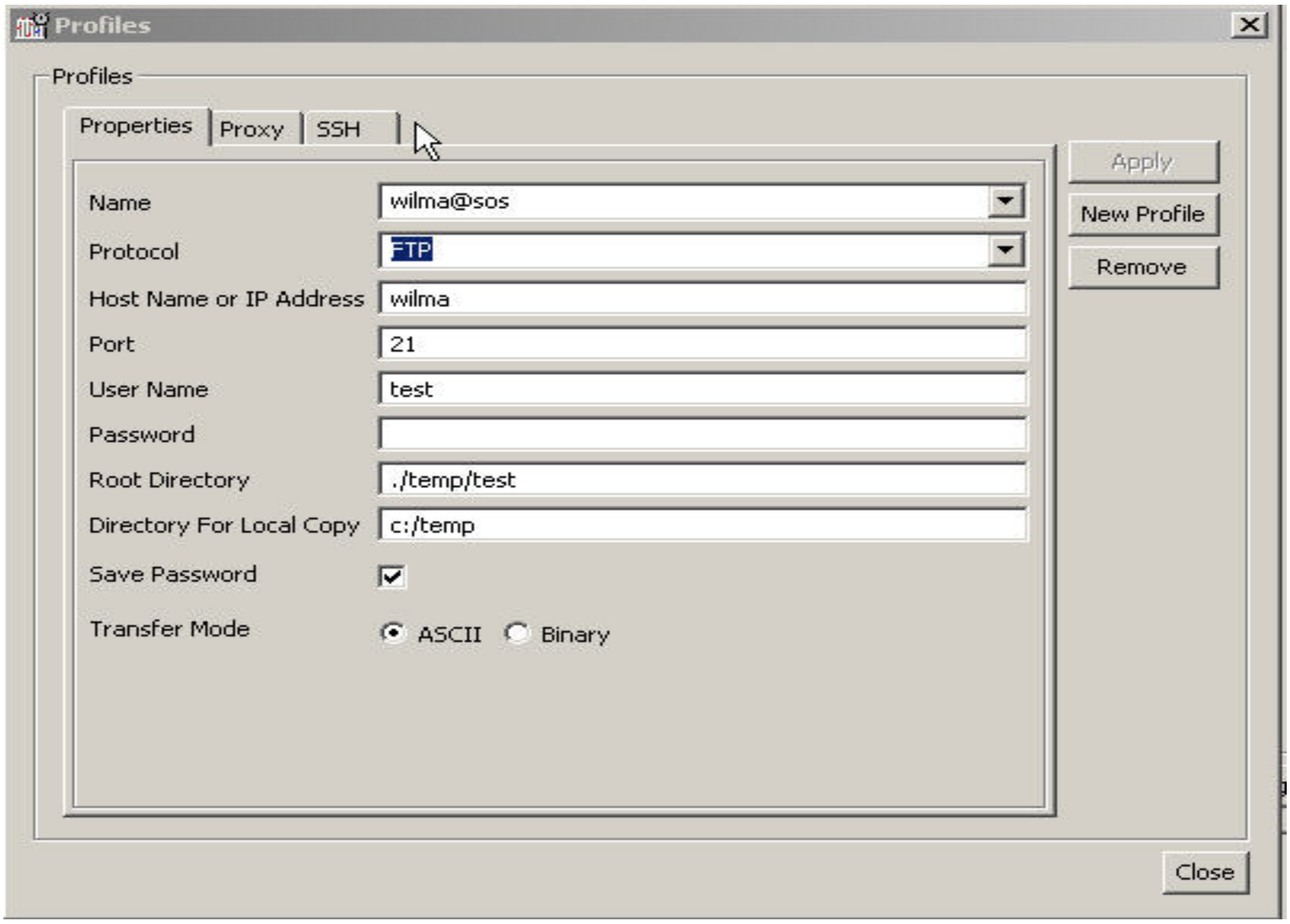
Closes the FTP dialog.

1.8.1.7.1.8 Log

Shows the protocol dialog. This contains all commands sent to the FTP server and its replies.

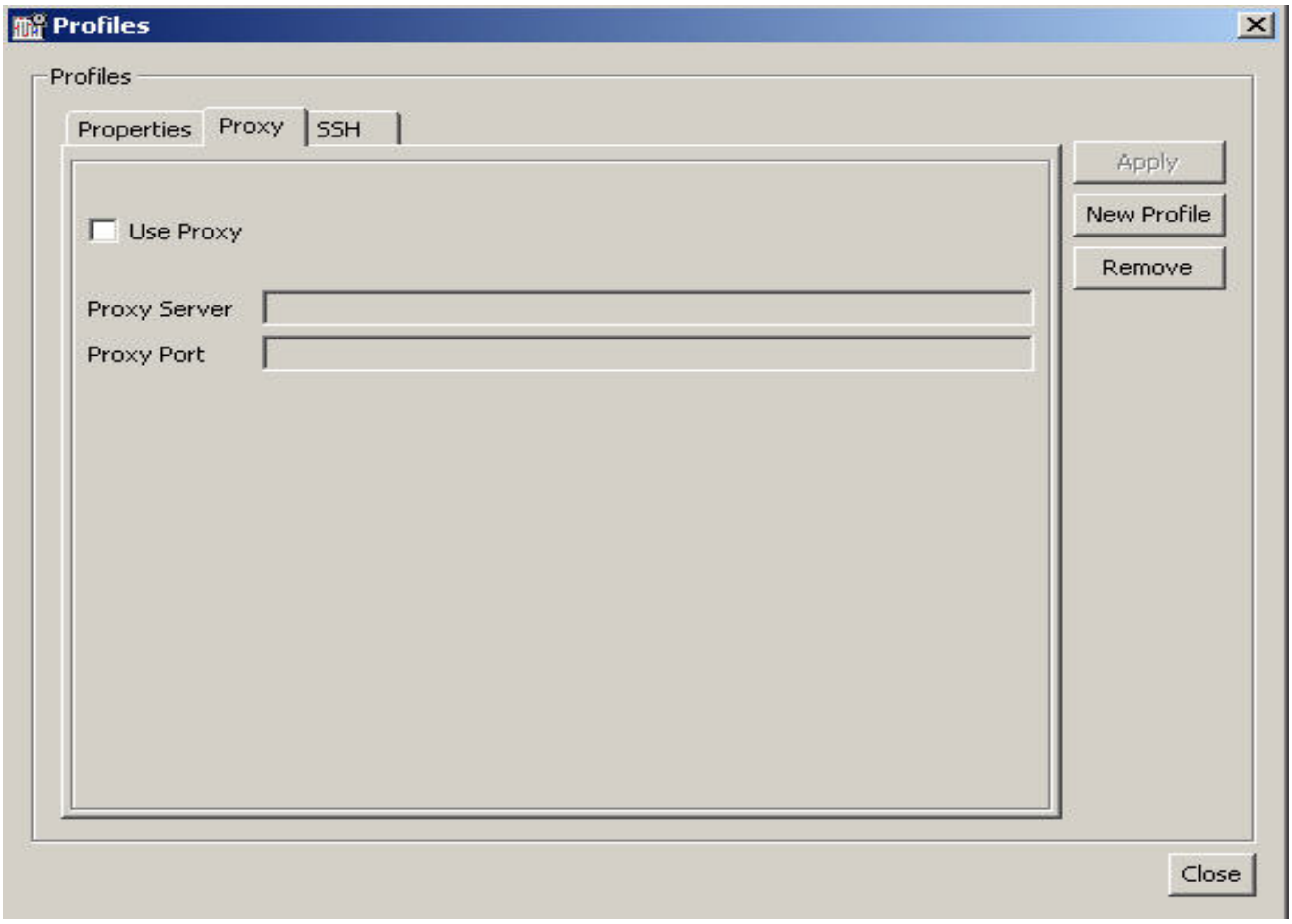
1.8.1.7.1.9 Profiles

Shows all configured FTP connections. Allows new connections to be added or existing ones to be edited or deleted.


Profiles

Profiles

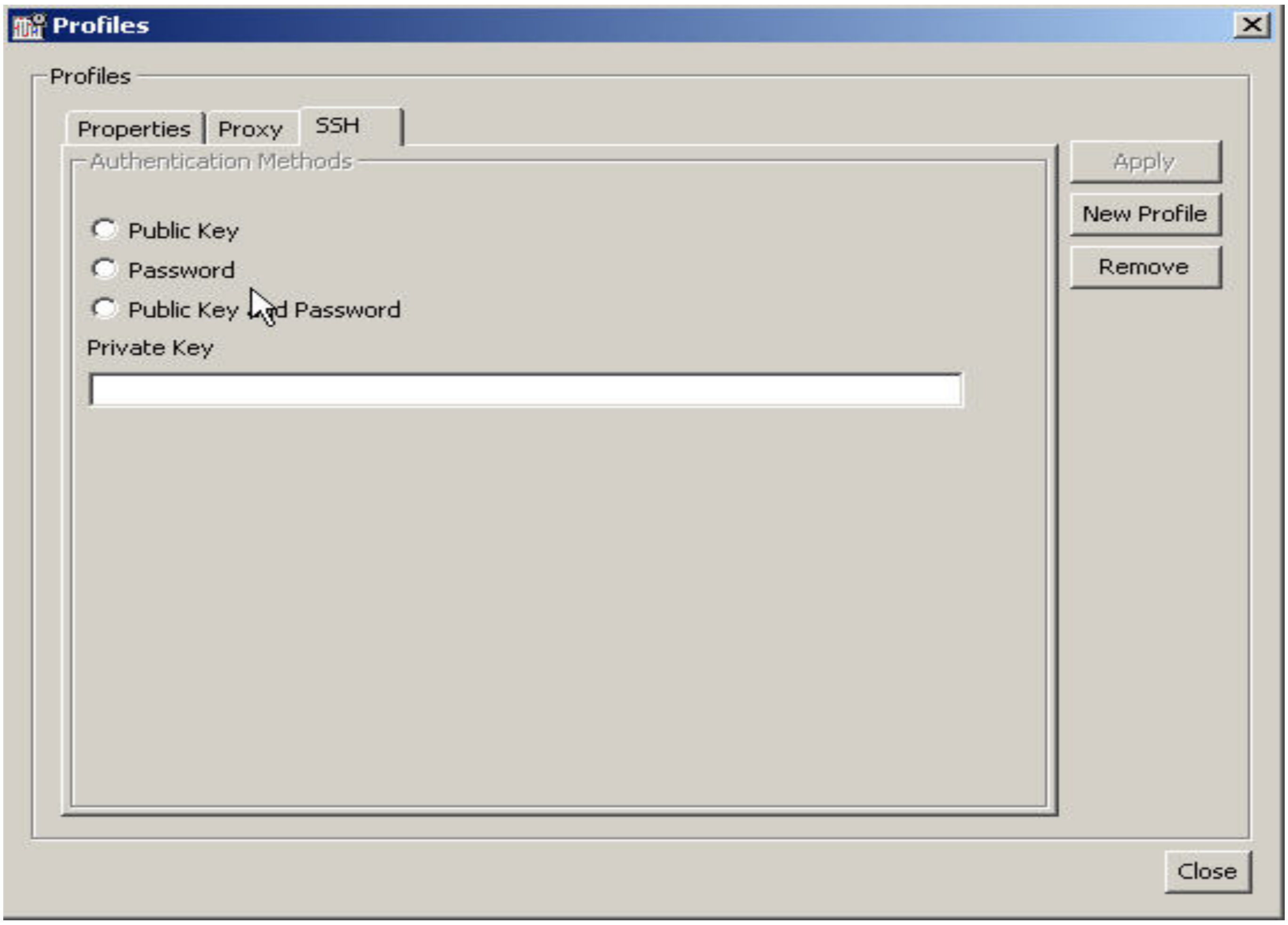
Name	wilma@sos
Protocol	FTP
Host Name or IP Address	wilma
Port	21
User Name	test
Password	
Root Directory	./temp/test
Directory For Local Copy	c:/temp
Save Password	✓
Transfer Mode	[✓ ASCII] Binary



Use Proxy [ ]

Proxy Server

Proxy Port



All FTP connections or profiles are saved in a configuration file that can be found under the address `scheduler_install_directory/config/factory.ini`. This file has the following section and entries for each profile:

[profile profilename]

host=

port=

user=

root=

localdirectory=

transfermode=

save_password=

protocol=SFTP

use_proxy=

proxy_server=

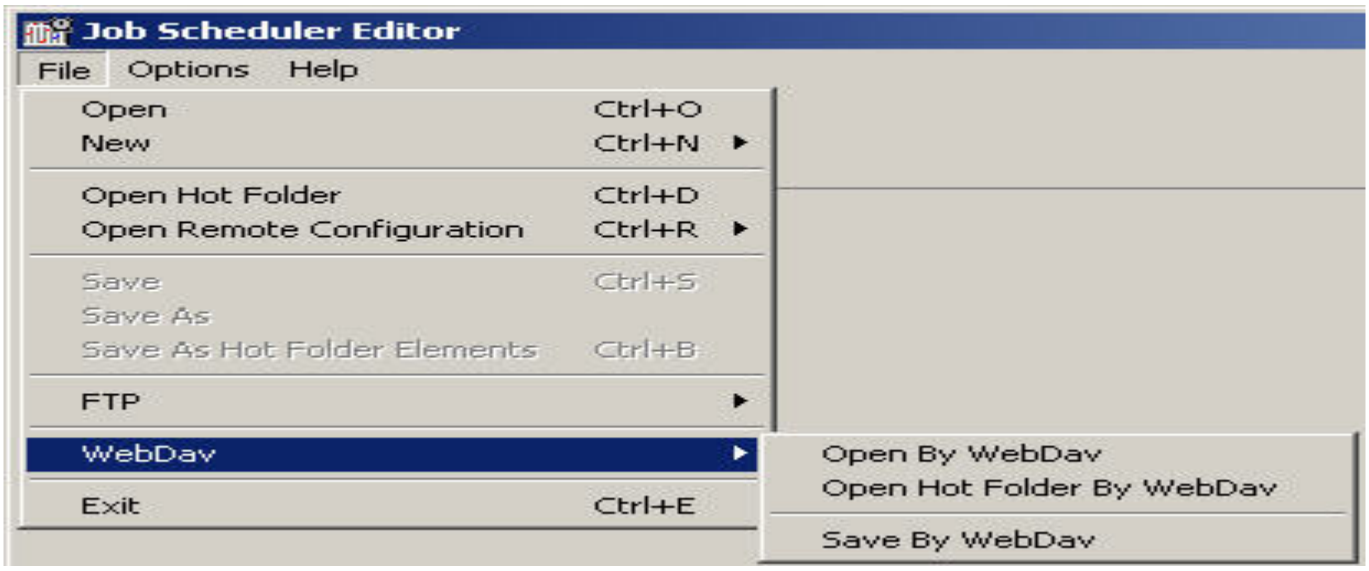
proxy_port=

auth_method=

auth_file=

1.8.1.7.2 WebDAV

JOE can open and save XML configuration files per WebDAV. The WebDAV dialog can be opened using the File -> WebDAV menu entries



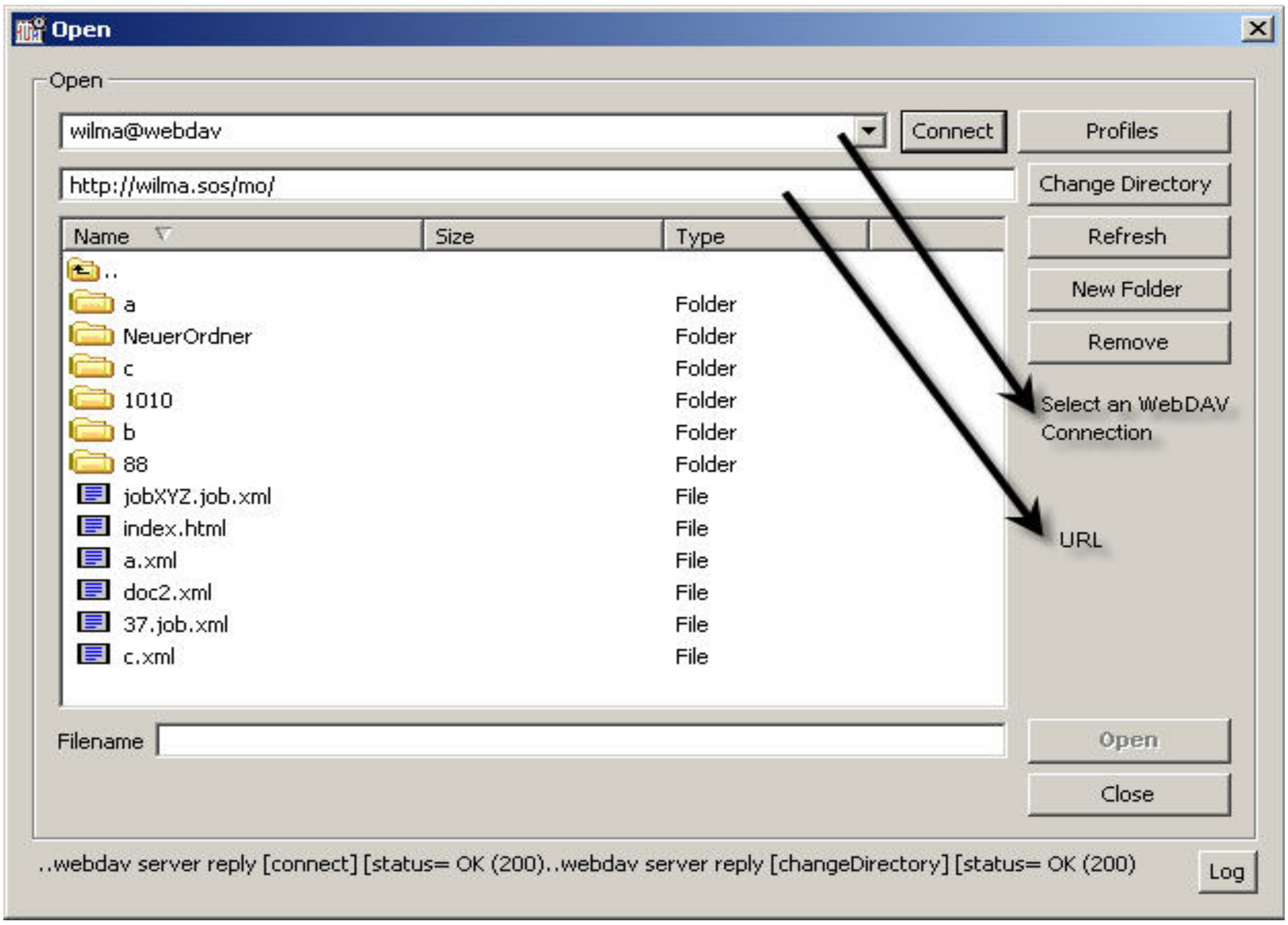
The WebDAV menu entry is only active, when the libraries necessary to make WebDAV connections are available. These libraries are not part of the JobScheduler distribution because of licensing reasons but can be downloaded from the Sourceforge `webdavclient4j` project under https://sourceforge.net/projects/webdavclient4j/.

The following libraries should be available in the *`scheduler_installation_directory`*`/lib` directory, although variation in the library name and number may occur.

- commons-logging.jar
- webdavclient4j-core-0.92.jar
- commons-codec-1.3.jar
- commons-httpclient-3.0.1.jar

JOE is started by a command file, which is found under *`scheduler_installation_directory`*`/bin`. The `CLASSPATH` in this file must be extended to include the above mentioned libraries.

JOE opens a dialog in which the profile (i.e. WebDAV connection) to be opened is selected, followed by the file to be opened.



Name	Size	Type
..		
a		Folder
NeuerOrdner		Folder
c		Folder
1010		Folder
b		Folder
88		Folder
jobXYZ.job.xml		File
index.html		File
a.xml		File
doc2.xml		File
37.job.xml		File
c.xml		File

```
..webdav server reply [connect] [status= OK (200)..webdav server reply [changeDirectory] [status= OK (200)
```

1.8.1.7.2.1 Control Elements

1.8.1.7.2.1.1 Select WebDAV Connection

WebDAV connections that have already been configured can be selected from the drop-down list

1.8.1.7.2.2 Change URL

This text box is used to show the pre-selected URL on the WebDAV server. This directory will be opened in the dialog as soon as the connection with the WebDAV server is made. When navigating on the server, this text box is used to show the current URL open. It can also be used for direct navigation. After entering the desired address in this box and clicking on the "Change Directory" button, the WebDAV server will directly go to the desired URL.

1.8.1.7.2.3 Refresh

Actualises all directory and file information from the WebDAV server

1.8.1.7.2.4 New Folder

Creates a new folder on the WebDAV server.

1.8.1.7.2.5 Remove

Deletes the file or folder selected from the WebDAV server.

1.8.1.7.2.6 Open

If the WebDAV dialog has been opened using the "Open by WebDAV" menu entry, then the "Open" button will cause the XML configuration file selected to be downloaded from the WebDAV server and opened in JOE.

If the WebDAV dialog has been opened using the "Open Hot Folder by WebDAV" menu entry, then only directories will be shown in the dialog. Selection of a hot folder will then cause all the XML configuration files within the folder to be downloaded from the WebDAV server and opened in JOE.

1.8.1.7.2.7 Close

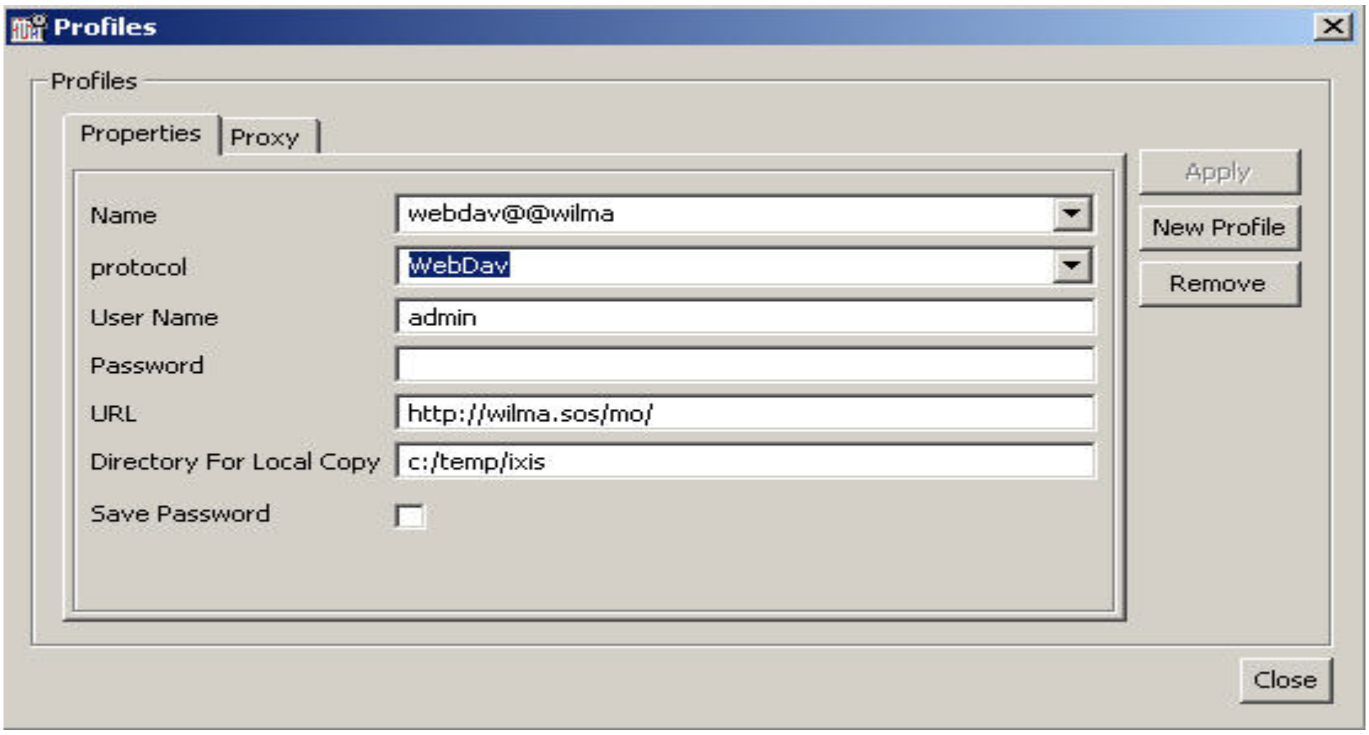
Closes the WebDAV dialog

1.8.1.7.2.8 Log

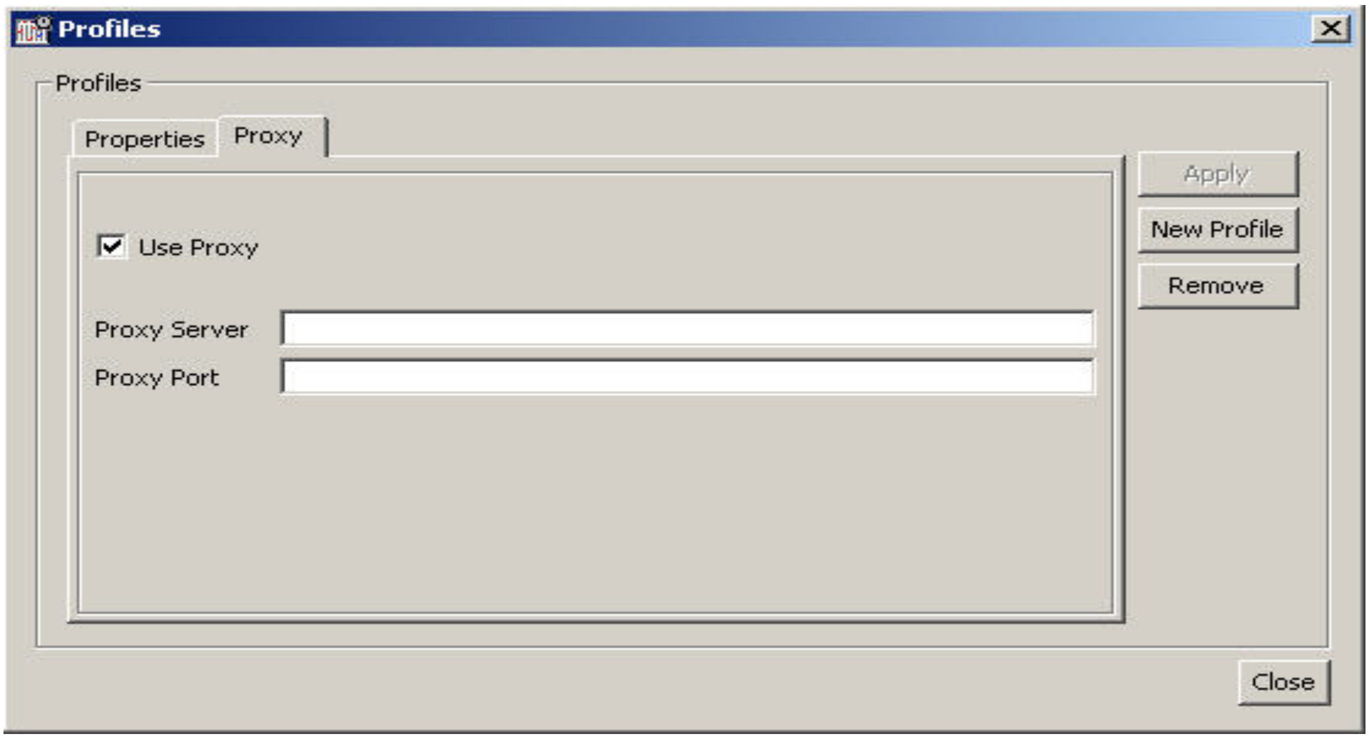
Shows the protocol dialog. This contains all commands sent to the WebDAV server and its replies.

1.8.1.7.2.9 Profiles

Shows all configured WebDAV connections. Allows new connections to be added or existing ones to be edited or deleted.



Name	webdav@@wilma
protocol	WebDav
User Name	admin
Password	
URL	http://wilma.sos/mo/
Directory For Local Copy	c:/temp/ixis
Save Password	☐



☑ Use Proxy

Proxy Server

Proxy Port

All WebDAV connections or profiles are saved in a configuration file that can be found under the address `scheduler_install_directory/config/factory.ini`. This file has the following section and entries for each profile:

[webdav_profile profilename]

url =

user =

password =

localdirectory=

save_password=

use_proxy=

proxy_server=

proxy_port=

debug=

1.9 Database

The JobScheduler be used either with or without a database. Operation with a database has the following advantages:

Task Queue

The JobScheduler stores all queued tasks in a table under its JobScheduler ID (Option `-id_`). When the JobScheduler restarts, then all queued tasks are reread from the database.

Orders

The JobScheduler saves every order in a database table with the JobScheduler ID (Option `-id`). When the JobScheduler restarts, it rereads the orders from the database.

Task History

The JobScheduler records information about every task performed, apart from the JobScheduler start and stop.

Instruction History

The JobScheduler records the course of every operation in the job stream in the instruction history.

Further information about the instruction history can be found here: Job History (page 138).

The creation of the database tables is described in »SQL Instructions Used by the JobScheduler« (page 341).

1.9.1 Settings

Database and Error Handling

`factory.ini` (section `[spooler]`, entry `db= …`)

`factory.ini` (section `[spooler]`, entry `need_db= …`)

`factory.ini` (section `[spooler]`, entry `max_db_errors= …`)

Table Names

`factory.ini` (section `[spooler]`, entry `db_variables_table= …`)

`factory.ini` (section `[spooler]`, entry `db_tasks_table= …`)

`factory.ini` (section `[spooler]`, entry `db_orders_table= …`)

`factory.ini` (section `[spooler]`, entry `db_history_table= …`)

`factory.ini` (section `[spooler]`, entry `db_order_history_table= …`)

Should a History be Written?

`factory.ini` (section `[spooler]`, entry `history= …`)

`factory.ini` (section `[spooler]`, entry `order_history= …`)

`factory.ini` (section `[job]`, entry `history= …`)

Should Protocols be included in the History?

`factory.ini` (section `[spooler]`, entry `history_with_log= …`)

`factory.ini` (section `[spooler]`, entry `order_history_with_log= …`)

`factory.ini` (section `[job]`, entry `history_with_log= …`)

The Addition of Custom Columns in the Task History

`factory.ini` (section `[spooler]`, entry `history_columns= …`)

`factory.ini` (section `[job]`, entry `history_columns= …`)

`Task.set_history_field()`

The Conditions for an Entry in the Task History

`factory.ini` (section `[spooler]`, entry `history_on_process= …`)

1.9.2 Application Programming Interface

The spooler object initializes table names

`Spooler.db_variables_table_name()`

`Spooler.db_tasks_table_name()`

`Spooler.db_orders_table_name()`

`Spooler.db_history_table_name()`

`Spooler.db_order_history_table_name()`

Describe history fields (`factory.ini` (section `[job]`, entry `history_columns= …`))

`Task.set_history_field()`

1.9.3 Commands

Show History

Command `<show_history>`

Command `<show_task>`

Command `<show_order>`

1.9.4 Error-Tolerance

The JobScheduler tolerates database errors and can carry out routine operations with a temporarily unavailable database. More detailed information can be found in `factory.ini` (section `[spooler]`, entry `need_db= …`).

The JobScheduler can be set to wait for a reconnection in the event of loss of contact with the database. In this situation, only commands other than TCP or HTTP are processed; The JobScheduler cannot be shut down, only terminated (see Termination (page 233)).

1.10 History

1.10.1 1. Purpose of the History

The progress of each job should be recorded and stored as follows:

- The record identifier is the ID of the task started for a job and is unique for each task.

- Scheduler Id
- Jobname
- Start time
- End time (if reached)
- The start event (should there be more than one start event, the JobScheduler will attempt to define the most important event)
- Parameter (in XML)
- The number of job steps (the number of the spooler_process calls)
- Error flag
- Error code and error text
- The job protocol (only when the protocol is recorded in a database)
- Extra Fields

Excerpts from the history can be read via the TCP interface.

The History Table Columns

id	The Unique Task Identifier (the primary table key)
spooler_id	Scheduler Id
job_name	Job name
start_time	Start time (yyyy-mm-dd HH:MM:SS)
end_time	End time (yyyy-mm-dd HH:MM:SS)
steps	Number of spooler_process() steps
cause	Cause of start
error	0: No error; 1: Error
error_code	Error code
error_text	Error text
parameters	Job parameters (if available) as XML document (Clob)
log	Job protocol (not in the tab-separated file when operating without a database)

1.10.2 2. History File

The history can either be written in a simple file or recorded in a database.

1.10.2.1 2.1 Simple File (tabular data)

A record file in the protocol file folder (Option `-log-dir=`) is created for every job. Fields in this file are separated by tabulator characters. An entry is made at the start of a job, which is then overwritten with the complete data after the job has ended.

The file name is: *log_dir* `/history`. *scheduler_id* . `job`. *jobname* . `txt`

The file is rewritten each time the JobScheduler is started. A previously existing file can be renamed and/or compressed before a JobScheduler start.

The JobScheduler writes column names in the first row of the file.

Tab characters in the file (resulting from error messages) are suppressed.

Every field has a maximum length of 1024 characters. Longer entries are cut off.

Archiving

Existing history files are overwritten when the JobScheduler is started. However, these files can be archived using the `history_archive` entry in the `factory.ini` (page 94):

```
[spooler]
history_archive = yes| no| gzip

[Job jobname]
history_archive = yes| no| gzip
```

`history_archive=yes` renames a history file. Note that this command causes the file name to be extended with an (exact to the second) time stamp.

`history_archive=gzip` compresses the file using `zlib` (`gzip`) from Jean-loup Gailly (http://www.gzip.org/zlib/). The file name ending is thereby extended with `.gz`. The resulting file can be decompressed using `gzip`. It is however also readable using Hostware with "`nl | gzip | history.gz`".

1.10.2.2 2.2 Database

All data is written in a database table. The task protocol is written in a blob.

The table name is `spooler_history`. If this file does not exist, then a database with the same name is created. If the database table `spooler_history` does not exist, then it will be created (only for MS Access databases).

The table names can be set in the `factory.ini` (page 94) configuration file:

```
[spooler]
  db_history_table   = tabellenname | SCHEDULER_HISTORY
  db_variables_table = tabellenname | SCHEDULER_VARIABLES
```

The next free job number is held in an entry in the `SCHEDULER_VARIABLES` table.

The `need_db=no` entry is used to allow the JobScheduler to start when the history database cannot be opened. The default setting here is `need_db=yes`.

```
[spooler]
db = odbc -db=scheduler
need_db = no
```

When only a database file name is specified with `db=` and `log_dir=*stderr` is set, then the database cannot be opened because of the missing directory. In this case the JobScheduler does not start if `need_db=yes` is set.

Automatic Creation of a MS Access Database

Should only a simple file name be specified with `db=`, then the JobScheduler can automatically create a Microsoft MS Access database when "`odbc -create`" is set (see ODBC file type).

The `SCHEDULER_HISTORY` and `SCHEDULER_VARIABLES` tables are used in the database. These tables are created as necessary using the Microsoft Access SQL syntax.

JobScheduler Start Entry

The JobScheduler makes an entry in the history with its own ID on starting. On ending, the JobScheduler writes the time in this entry, so that both start and end times are noted. The job name is "`(Spooler)`".

1.10.2.3 2.3 Configuration of the factory.ini File

The following parameters can be set in the `factory.ini` configuration file:

```
[spooler]
db                 = database

db_history_table   = SCHEDULER_HISTORY
db_variables_table = SCHEDULER_VARIABLES
history            = no|yes
history_columns    = field1, field2,...

history_on_process = yes|1|2
history_with_log   = no|yes|gzip
history_archive    = no|yes|gzip

[Job jobname]
history            = no|yes
history_columns    = field1, field2,...

history_on_process = yes|1|2
history_with_log   = no|yes|gzip
history_archive    = no|yes|gzip
```

Settings made in `[Job` *jobname*`]` have priority over those made in `[spooler]`.

`history=no` suppresses the history. Should a database be specified, however, the entry for the start of the JobScheduler will still be made.

`history_on_process` sets the number of requests of `spooler_process()` which can be made before an entry is made in the history. Should `spooler_open` return false, then an entry will not be made when `history_on_process` is set to 1.

`history_with_log` allows the task protocol to be recorded in the database as well. The protocol may be compressed if desired.

1.10.2.4 2.4 Start Events (Cause)

The `cause` of a job start is noted in a row in the history:

`none`	The task has not started (this is not noted in the history)
`period_once`	`<run_time once="yes">`
`period_single`	`<run_time single_start="..."> `
`period_repeat`	`<run_time repeat="...">`
`job_repeat`	`spooler_job.repeat=...`
`queue`	`spooler_job.start()` oder `<start_job>`
`queue_at`	as `queue`, with specified start time (Option `at`)

`directory`	Directory monitoring (`start_when_directory_changed`) has started the job
`signal`	`<signal_object>`
`delay_after_error`	`spooler_job.delay_after_error`

1.10.3 3. Task Recognition and Extra Fields

Every task has an identifier. When a database is used, the JobScheduler obtains this identifier from the `SCHEDULER_VARIABLES` table. This identifier is then unique to all JobSchedulers which use this database table. Otherwise consecutive numbers are given. The first task is given the number 1.

The identifier can then be recalled in scripts, which use the JobScheduler API, with the `id` property:

```
meine_id = spooler_task.id
```

The history can accept further fields which the job can fill:

```
spooler_task.history_field( "fieldname" ) = value
```

The *fieldname* must be declared as a row in the tabular file or in the history table. The case used here is irrelevant.

When the history is recorded in a database, then it may be significant which type a *value* is, in particular if it is a number or a character string.

1.10.3.1 3.1 Extra Fields in a Tabular File

Extra fields must be declared in the `factory.ini` configuration file

```
[spooler]
history_columns = columnlist


[Job jobname]
history_columns = columnlist
```

The *columnlist* is a list of column names, separated by columns and which is included in the tabular file.

1.10.3.2 3.2 Extra Fields in a Database

Extra rows in the history table are automatically recognized.

When the JobScheduler creates the `SCHEDULER_HISTORY` table, it also creates the necessary rows as defined in `history_columns`. These rows are defined as `char(250)` type.

1.10.4 4. Reading the History via the TCP Interface

```
<show_history job="jobname" prev="number| all "what="all"/>
```

Returns, for the current job *jobname*, the last *number* of entries in the history, sorted in reverse order. The default value for *number* is 10. All entries are read when `tail="all"` is set. Note that this function requires a lot of memory. This is because the history is compiled as an XML document with the DOM in the main memory. A maximum of 1000 entries will be returned.

`what="all"` returns the job protocol as well.

Extracts out of the history can be read using the following (note that the `what` attribute can always be used here):

to obtain the *number* of entries before the *ID*:

```
<show_history job="jobname" id="id" prev="number"/>
```

to obtain the *number* of entries after the *ID*:

```
<show_history job="jobname" id="id" next="number"/>
```

to obtain only the *id* entry: it is recommended that the job protocol for a particular task is read with `what="all"`.

```
<show_history job="jobname" id="id"/>
```

The result looks like:

```
<history>
    <history.entry id="identifier" job="jobname" start_time="starttime" end_time="end
 time" ...>

        <variableset>
            <variable name="name1" value="value1"/>
            <variable name="name2" value="value2"/>

        </variableset>
        <log>Jobprotocol</log>

    </history.entry>
    <history.entry ...>
        ...
    </history.entry>
    ...
</history>
```

1.10.5 5. Error Handling

Errors on opening or writing the history are noted in the JobScheduler protocol and otherwise ignored.

1.11 Running the JobScheduler as a Service or Daemon

The JobScheduler is operated on Window systems as a service. This is automatically set-up by the JobScheduler installation program. In the following section, the command line operations relevant to the set-up are described. These operations are written by the installation program in the `.\bin\jobscheduler.cmd` (Windows) and `./bin/jobscheduler.sh` (Unix) start scripts.

Calls

`scheduler -remove-service [-service-name=` *name*`] [-id=` *id*`]`

`scheduler [-remove-service] -install-service`

[`-service-name=` *name*] [`-service-display=` *shownname*] [`-service-descr=` *description*]

[`-need-service=` *name*]

[*options*]

Parameters

`-remove-service`

Removes the service. The service must exist.

`-install-service`

Installs the service. Additional command line options, which are not described here, are passed over when the JobScheduler service starts (see `-id=`, `-log-dir=`)

`-service-name=` *name*

Sets the (internal) service name for `-remove-service` or `-install-service`.

The default value is `sos_scheduler`: when the option `-id=` is set, the default value is `sos_scheduler_` *id*

`-service-display=` *shown name*

Sets the name under which the service will run.

The default value used should the service name not be specified is "SOS JobScheduler". When the option `-id=` is set: "SOS JobScheduler -id=*id*". and when `-service-name=` is specified, then this is used as the default name.

`-service-descr=` *description*

The service description. The default value here is "Job Automation Processor".

The JobScheduler ignores this option in Windows NT 4.

`-need-service=` *Service*

Specifies another service which this service depends upon. Windows then starts the current service when the other is running. This option can be repeatedly set. An unknown service name will not cause an error in Windows.

This option can be used, e.g. when the JobScheduler and its database are operated for the same server and it should be guaranteed that the database service is up and running before the JobScheduler starts.

The name of a service is shown in the System services panel.

Errors on Starting

The windows service manager does not allow a service to return an error message, should the service be unable to start. Therefore, services make simple entries in the events viewer.

In addition the error message is sent per E-mail. This is done using the `log_mail_from`, `log_mail_to`, `log_mail_cc`, `log_mail_bcc` und `smtp` parameters specified in the `factory.ini` file. A configuration file specified using `-ini=` is *not* used in this situation. However, settings specified in the `sos.ini` file, section `[mail]` will be used (see also `sos.ini`, section `[spooler]` (page 175)).

1.12 Backup JobSchedulers

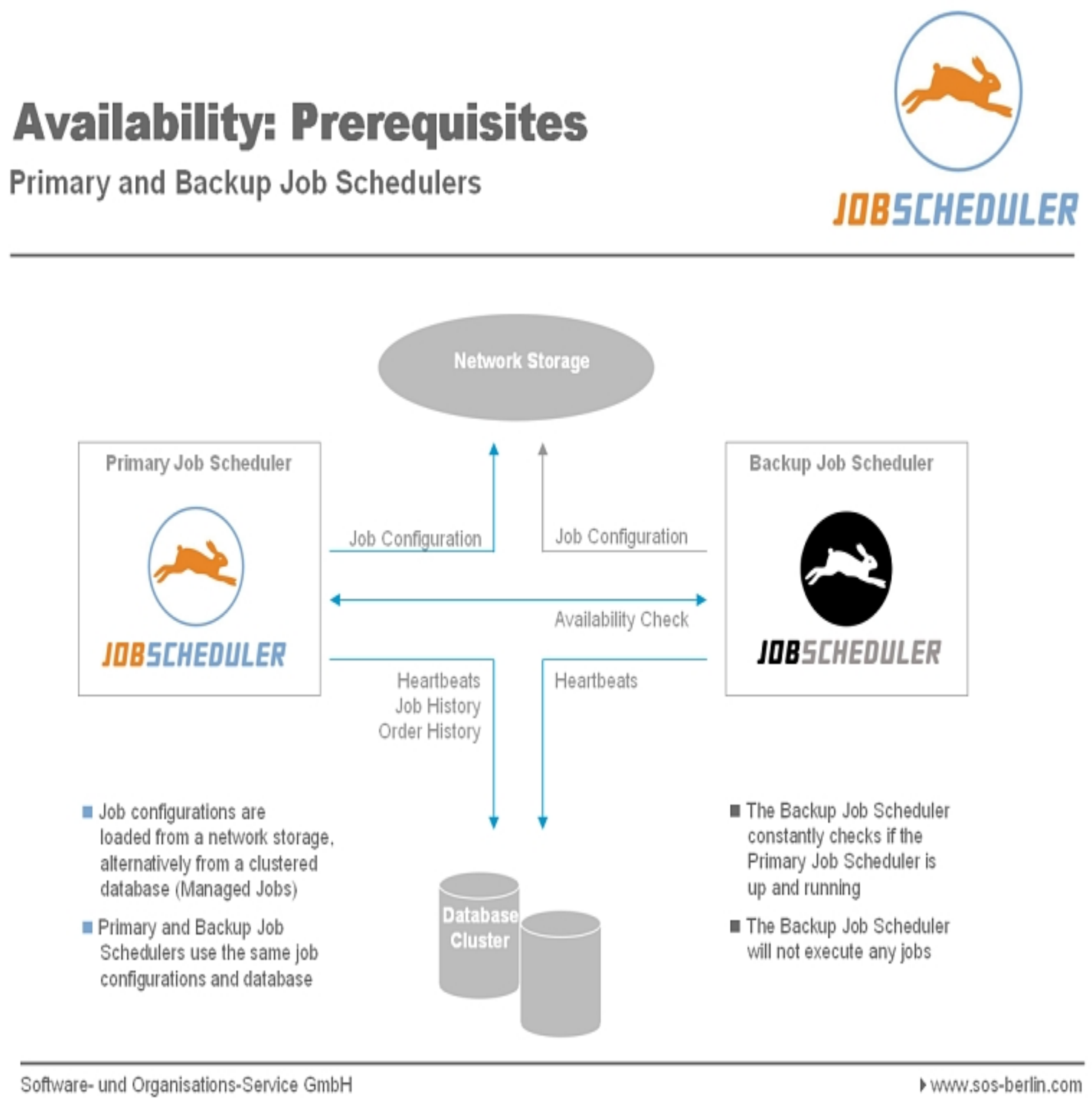
1.12.1 JobScheduler Backup Cluster

A JobScheduler backup cluster ensures fail-safe operation of a (primary) JobScheduler. The cluster comprises this primary JobScheduler and one or more reserve (backup) JobSchedulers. A fail-safe system consists of a primary JobScheduler and at least one backup, with both these JobSchedulers running on different computers.

All the JobSchedulers in a backup cluster show their own availability by sending out "heartbeats" and, at the same time, checking whether the other Schedulers in the cluster are available by monitoring their "heartbeats". Should one of the backup JobSchedulers determine the absence of the heartbeat from the primary JobScheduler over a longer period of time (ca. 1-2 minutes), then it will take over processing. This means that it will continue to process the open orders and jobs started by the primary JobScheduler and, if required, start new jobs.

At the most, only one JobScheduler in a cluster is active - the primary JobScheduler - and starts jobs and processes orders. The other backup JobSchedulers are inactive - that is they wait for the primary JobScheduler to fail before becoming active and taking over processing.

The requirements for the operation of a backup JobScheduler cluster are shown schematically in the following diagram and described in detail in the next section.



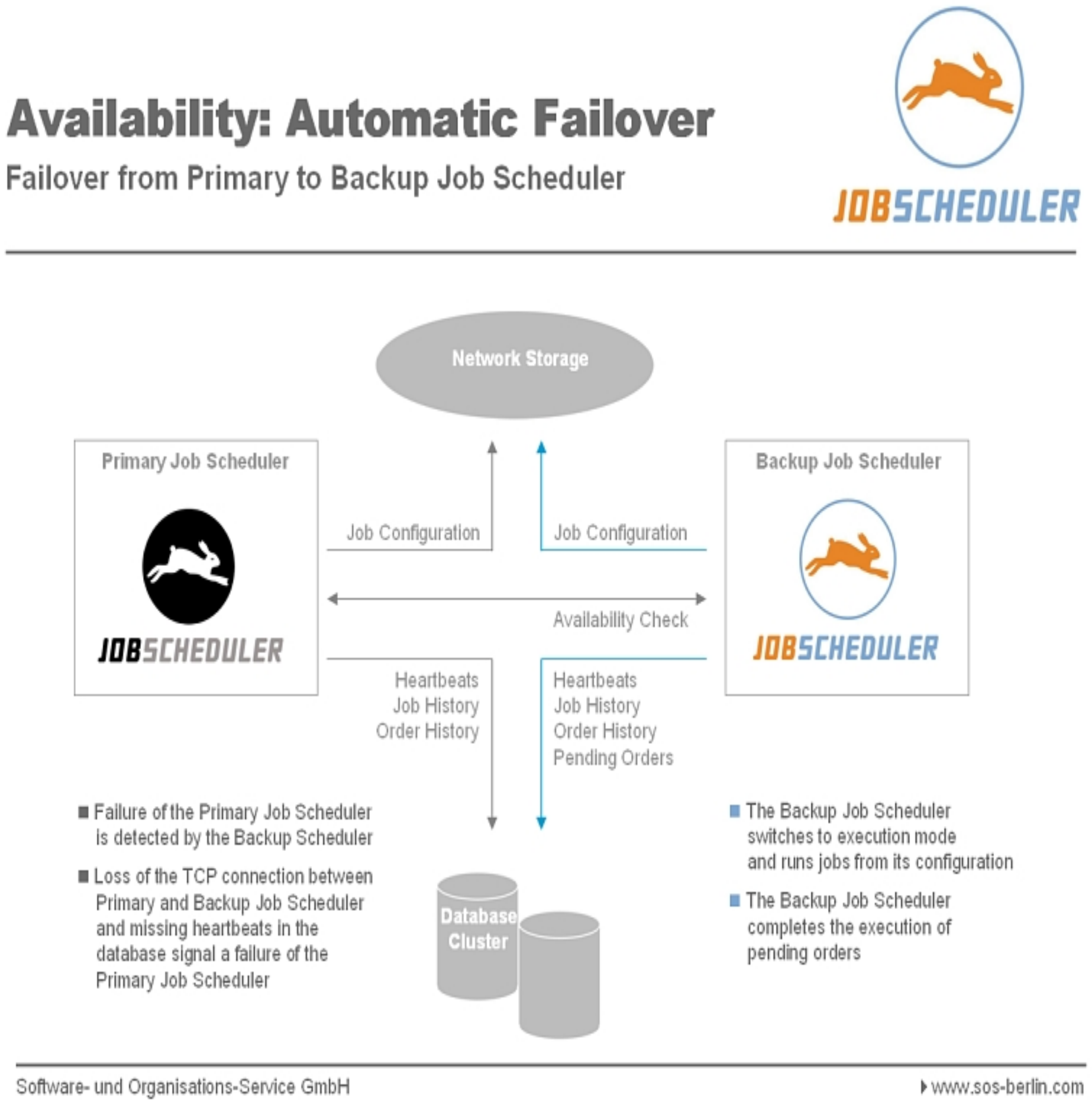
Availability: Prerequisites

Primary and Backup Job Schedulers

- Job configurations are loaded from a network storage, alternatively from a clustered database (Managed Jobs)
- Primary and Backup Job Schedulers use the same job configurations and database

- The Backup Job Scheduler constantly checks if the Primary Job Scheduler is up and running
- The Backup Job Scheduler will not execute any jobs

The diagram below shows schematically the situation where a backup JobScheduler has become active and taken over the processing of jobs and orders:


Availability: Automatic Failover
Failover from Primary to Backup Job Scheduler
JOBSCHEDULER
Network Storage
Primary Job Scheduler
JOBSCHEDULER
Job Configuration
Job Configuration
Backup Job Scheduler
JOBSCHEDULER
Availability Check
Heartbeats
Job History
Order History
Heartbeats
Job History
Order History
Pending Orders
■ Failure of the Primary Job Scheduler is detected by the Backup Scheduler
■ Loss of the TCP connection between Primary and Backup Job Scheduler and missing heartbeats in the database signal a failure of the Primary Job Scheduler
Database Cluster
■ The Backup Job Scheduler switches to execution mode and runs jobs from its configuration
■ The Backup Job Scheduler completes the execution of pending orders
Software- und Organisations-Service GmbH
www.sos-berlin.com

1.12.1.1 Conditions for Operating a JobScheduler Cluster

- All the JobSchedulers use the same database - Oracle, DB2, MySQL and Postgres databases are supported.
- The JobSchedulers must all use the same configuration file or an exact copy of the configuration file.
- The primary JobScheduler and the backups in the cluster are all started using the same JobScheduler ID.
- All the JobSchedulers - that is, the primary and the backups - must be started using `-exclusive`.

1.12.1.2 Starting a JobScheduler Cluster

The JobSchedulers which form the cluster are to be started in arbitrary series. The active (primary) JobScheduler is the first one to be started without the `-backup` option set.

1.12.1.3 Command Line Parameters

The following command line parameters configure a JobScheduler as a member of a backup cluster:

- `-exclusive` specifies that the JobScheduler is a member of the backup cluster.
- `-backup` specifies that a JobScheduler is to operate as a backup. Should this parameter not be set, then the JobScheduler is defined as being *primary*. Note that there can be more than one backup JobScheduler - should the active JobScheduler fail, then all the backup JobSchedulers have the same start priority.
- `-backup-precedence` is used to set the order in which backup JobSchedulers are made active. Should the active JobScheduler fail, then the JobScheduler with the smallest `backup-precedence` will become active.

1.12.1.4 Stopping a JobScheduler in a Backup Cluster Using Web Interface Functions

Job processes which are still running are allowed to finish when a JobScheduler is stopped. New processes are not started. The `-timeout` =<value> parameter can be used to specify a time after which running job processes are forced to stop immediately.

1.12.1.4.1 Stopping all the JobSchedulers in a Backup Cluster

A cluster is stopped in that the "terminate cluster" command is called from the JobScheduler Web Interface. This command stops all the JobSchedulers in the cluster.

The corresponding XML command is <terminate all_schedulers="yes">

1.12.1.4.2 Stopping all the JobSchedulers in a Backup Cluster Using Timeout

The JobScheduler Web Interface "terminate cluster within 60s" command is used to stop all the JobSchedulers in a cluster. This stops all the JobSchedulers in the cluster. All processes running are stopped within 60 seconds.

The corresponding XML command is <terminate all_schedulers="yes" timeout="60">

1.12.1.4.3 Restarting all the JobSchedulers in a Backup Cluster

All JobSchedulers in a cluster are stopped when the "terminate and restart cluster" command is called from the JobScheduler Web Interface. This causes all the JobSchedulers in the cluster to be stopped and then restarted.

After all the JobSchedulers have been restarted, then the primary JobScheduler is the active JobScheduler.

The corresponding XML command is <terminate all_schedulers="yes" restart="yes">

1.12.1.4.4 Restarting all the JobSchedulers in a Backup Cluster with Timeout

All JobSchedulers in a cluster are stopped when the "terminate and restart cluster" command is called from the JobScheduler Web Interface. This causes all the JobSchedulers in the cluster to be stopped and then restarted. The JobScheduler which was active before the restart will become active once more. All job processes still running will be stopped after 60 Seconds.

After all the JobSchedulers have been restarted, then the primary JobScheduler is the active JobScheduler.

The corresponding XML command is <terminate all_schedulers="yes" restart="yes" timeout="60">

1.12.1.4.5 Stopping the Active JobScheduler: Backup JobSchedulers Remain Started but Do Not Become Active

An active JobScheduler is stopped by calling the "terminate" command from the JobScheduler Web Interface. This command has no effect on backup JobSchedulers, they will not take over operation because no failure of the primary JobScheduler has occurred.

The corresponding XML command is <terminate>

1.12.1.4.6 Restarting a Primary JobScheduler: Backup JobSchedulers Remain Started but Do Not Become Active

A JobScheduler is stopped and then restarted by entering the "terminate and restart" command in the JobScheduler Web Interface.

The corresponding XML command is <terminate restart="yes">

1.12.1.4.7 Restarting a Backup JobScheduler

A JobScheduler is stopped and then restarted by entering the "terminate and restart" command in the JobScheduler Web Interface.

A backup JobScheduler restarted in this way will remain inactive after the restart. However, an inactive primary JobScheduler running in a cluster will become active after this command.

The corresponding XML command is <terminate restart="yes">

1.12.1.5 Reactivating a Primary JobScheduler

1. The primary JobScheduler is started. As a backup JobScheduler is already running, the primary JobScheduler does not become active and does not take over processing.
2. The Backup JobScheduler is then restarted (using "terminate and restart"). As the primary inactive JobScheduler becomes active, as soon as no other JobScheduler is active, it then takes over processing. Note that should there be more than one primary JobScheduler, the JobScheduler which will become active is not fixed.

1.12.1.6 Handing Over Processing to a Backup JobScheduler

The primary JobScheduler is stopped "fail-safe" from the Web Interface. A running backup JobScheduler then becomes active and takes over processing. When, however, the primary JobScheduler is stopped using restart, then it is not clear whether or not a backup JobScheduler will become active or whether the primary JobScheduler will remain the active processor.

1.12.1.7 Behavior As A Windows Service

- *Stopping* by way of the Windows Service Panel has the same effect as using the <terminate> command. That is, the backup JobScheduler(s) do not become active. Should, however, a backup JobScheduler be stopped and there be an inactive primary JobScheduler, then this primary JobScheduler will become active.
- *Restarts* of the Windows service are comparable with use of the <terminate restart="yes"> command. A primary JobScheduler and not the backup JobScheduler(s) becomes active.

1.12.1.8 Behavior When Restarting a Computer

- When a computer (on which the active JobScheduler is running) is shut down, then a backup JobScheduler running on a second computer (continue_exclusive_operation="yes") will become active.
- When both a primary and a backup JobScheduler are restarted, e.g. by server reboot, then it can be that the backup JobScheduler starts first. In this case, the backup JobScheduler does not become active immediately but first of all waits to see if it receives a heartbeat from the primary JobScheduler. Only when the backup JobScheduler has not received a heartbeat within 60 seconds does it start processing. This is comparable with the standard backup JobScheduler behavior in the event of a missing heartbeat.

1.12.1.9 Making an Inactive Backup JobScheduler the Active Primary JobScheduler

When an active backup JobScheduler has been stopped and is then restarted, then it will be inactive. Should in this situation the primary JobScheduler then be unavailable for a longer period of time, the backup JobScheduler must then be started as the primary JobScheduler. This can be done by using the `start_exclusive` parameter instead of `start` when calling the `jobscheduler.cmd` shell script.

1.12.1.10 Start Script Commands

The JobScheduler starts as specified in the Setup when the [start] parameter is given, without any further information.

The following additional commands are available for the operation of a JobScheduler in a backup cluster:

- **terminate_cluster** Shuts down all the JobSchedulers in a backup cluster
- **restart_cluster** Restarts all the JobSchedulers in a backup cluster. The primary JobScheduler active before the restart remains active.
- **terminate_fail-safe** Stops a JobScheduler. Another (inactive) JobScheduler in the cluster becomes active.
- **start -exclusive** Starts a primary JobScheduler in a backup cluster.
- **start -exclusive -backup** Starts a backup JobScheduler in a cluster.
- **start -exclusive -backup -backup-precedence=[n]** Starts a backup JobScheduler in a cluster with the `backup-precedence` [n].

1.12.1.11 Further reading

- Watchdog-Monitor for Cluster Failover.

2 Jobs

2.1 What is a Job?

A job determines the program to be executed, its run time and what is to be done in the event of an error occurring. Further, any parameters to be used, pre and post processing, locks preventing simultaneous access to a file and possible follow-on jobs may also belong to a job configuration.

A Job is defined in the XML configuration using **`<job>`** .

2.1.1 Job Configuration

The XML configuration of a job can be carried out in the central start configuration file, (which is usually `./config/scheduler.xml`) or in a separate configuration file in the configuration directory which is monitored by the JobScheduler (usually `./config/live`) - see also (page 88)

2.1.2 Implementation

The JobScheduler starts executable programs and can start individually implemented jobs, which use the JobScheduler's API.

Executable Programs

Executable programs may be implemented as *executable*, *shell scripts* or as *batch files*. This includes programs, such as JavaScript, VBScript, Perl, PHP, Ruby etc. for which an interpreter needs to be started with the executable program file, Java classes can be started using the Java Virtual Machine (JVM) configured for the JobScheduler.

Executable programs can be configured using the following elements:

`<script language="…">`

<script language="shell">

The JobScheduler creates a shell for the execution of the program.

Example:

```
<job name="simple_shell">
  <script language="shell"><![CDATA[
    echo hello world
    call my_script.cmd
    my_prog.exe
  ]]></script>
</job>

<job name="simple_command">
  <script language="shell">
    <include file="my_script.cmd"/>
  </script>
</job>
```

Commands for the command line can be directly included as content of the `<script>` elements. Alternatively, shell scripts can be specified using `<include>`.

<script language="java">

The JobScheduler starts a JVM for the execution of the Java class.

Example:

```
<job name="simple_java">
  <script language   = "java"
          java_class = "sos.scheduler.ftp.JobSchedulerFTPReceive"/>
</job>
```

Implementation with the JobScheduler API

Job implementations can use the JobScheduler API - for example for logging, informing per e-mail, calling job, task and order objects, etc.

The JobScheduler does not start these scrips per interpreter (see above) but instead makes use of a sub-programme interface that is provided by the respective the script language. The JobScheduler then makes the objects and methods of its API available to such languages.

Java

A Job implemented with the Java programming language inherits from the abstract class sos.spooler.Job_impl. (see Javadocs for details).

The Java Interface provides additional classes in the same manner as the COM, JavaScript and Perl interfaces. This allows the implementation of jobs in these languages.

The address of the implemented class is specified in `sos.ini` (section `[java]`, entry `class_path= …`).

JavaScript and JScript (page 354)

JavaScript is available in the SpiderMonkey implementation (http://developer.mozilla.org/en/docs/SpiderMonkey) in the JobScheduler installation package for all platforms.

Microsoft JScript is available on Windows platforms.

VBScript

Microsoft VBScript is available on Windows platforms.

Perl

Perl is generally already installed on Unix platforms. The JobScheduler installation programme attempts to configure the sub-programme interface to an existing Perl installation.

A Perl implementation such as that from http://www.activestate.com, can be installed if required.

COM (page 155)

A job can be implemented in any program language as a COM-Server (for Windows). Further information can be found in the section on Spooler Scripts (page 155).

2.1.3 The Status of a Job

Jobs have one of the following statuses:

`pending`	No task is running. This is the starting status.
`running`	At least one task is running.
`stopping`	The job is stopping. The JobScheduler will not start another task and all current tasks are being stopped. As soon as all tasks are stopped, the job status changes to `stopped`. See also the command: `<modify_job cmd="stop">`.
`stopped`	No tasks are running and no further tasks will be started by the JobScheduler. See also the command: `<modify_job cmd="stop">`.
`read_error`	The `reread` command has caused an error and the job is unusable as the program code cannot be read from the underlying file. See also the command: `<modify_job cmd="reread">`.
`error`	The JobScheduler does not start any new tasks after an error has occurred.

2.1.4 Changing the Status of a Job

The `<modify_job>` element is used to change the status of a job. The JobScheduler has a built-in HTML interface (page 180) with provision for the necessary operations.

2.1.5 Starting a Task

The `<run_time>` parameter is used in the configuration to specify whether a task should be started once or repeatedly.

Both the `<start_job>` and the `Job.start()` API method can be used to start a task.

A task will start automatically when no other task is running and when one of the following conditions are true:

- At the start of a `<period>`, when `repeat=` or `single_start=` is specified in the period.
- When a previous run set `Task.repeat` and the repeat time has been reached.
- When a previous run caused an error and `Job.delay_after_error` has ended.
- When the interval after the end of the previous task defined in `<period repeat="…">` has ended.
- When directory monitoring (page 168) is active and a change occurs in the monitored directory.

In addition, a task will start when:

- An order for the job is present and the number of tasks running is less than that specified in `<job tasks="…" >`.

A task will only start when it has a start time (`at`) or:

- the job has not been stopped,
- a period for the current time is given,
- and the (`Job.delay_after_error`) delay after an error is not active.

2.1.6 Locks

Jobs can be given locks, in order to stop the simultaneous processing by two tasks.

2.1.7 Directory Monitoring

The JobScheduler can start a job when a change takes place in a monitored directory. Further details can be found in the chapter Directory Monitoring (page 168).

2.1.8 Monitor Scripts

A monitor script can be configured for a job . The JobScheduler calls the functions of this script at the beginning and end of a task and before and after the `spooler_process()` methods.

The monitor can be defined using the `<monitor>` element.

Example:

```
<job name   = "simple_ftp"
  <params>
    <param name = "ftp_host"              value = "localhost"/>
    <param name = "ftp_user"              value = "anonymous"/>
    <param name = "ftp_password"          value = "anonymous"/>
  </params>

  <script language   = "java"
          java_class = "sos.scheduler.ftp.JobSchedulerFTPReceive"/>

  <monitor>
    <script language="javascript"><![CDATA[
      function spooler_task_before() {
        var today = yy = mm = dd = "";
        today = new Date();
        yy = today.getYear() + 1900;
        mm = today.getMonth() + 1;
        dd = today.getDate();
        if (parseInt(mm) < 10) mm = "0" + mm;
        if (parseInt(dd) < 10) dd = "0" + dd;
        spooler_task.params.set_var("ftp_file_path", "^test_" + yy + "-" + mm + "-" +
 dd + "\.csv$" );
        return true;
      }
    ]]></script>
  </monitor>
</job>
```

The monitor script uses a standard job included with the JobScheduler installation package to transfer a file per FTP. Thereby the job parameters for the name of the file to be transferred is dynamically created from the current date.

2.2 Implementation of JobScheduler scripts as COM Classes

In previous versions of the JobScheduler, jobs were created using a Scripting Engine and the code (text) of the scripts (VBScript, JScript, PerlScript) were directly entered in the `<script>` element.

A JobScheduler job can also be implemented as a com class. This class can be implemented in every language which supports com, such as C#, C++, Delphi and VisualBasic (VB6 or VB.net). The com class can, however, also be implemented with VBScript, JScript und PerlScript using the "Windows Script Component".

This class allows use of job, thread script or JobScheduler script methods as follows:

```
spooler_init()
spooler_exit()

spooler_open()

spooler_close()

spooler_process()

spooler_on_success()

spooler_on_error()
```

Each method is optional. Should a method not be implemented, then the existing JobScheduler script procedure will be followed.

In addition, this class should provide a method with which the JobScheduler context can be assigned:

```
spooler_set_context( context)
```

This context is a com object (`IDispatch`) with the following properties:

```
log
JobScheduler

job

task
```

These properties deliver the `spooler_log`, `spooler_task`, `spooler_job` and `JobScheduler` objects, known from the JobScheduler-Script.

Similarities between script in the Scripting Engine und the com class:

- the (`spooler_init()` , `spooler_open()` etc.) requests are the same in both
- these calls are optional in both

Differences between Script in the Scripting Engine and the com class:

Skript using the Scripting Engine	COM Class
Source code is included in the configuaration (or externally using `<include>`).	Implementation is independent of the JobScheduler. Any language which generates COM classes with late binding (`IDispatch`) can be used.

The context makes the following predefined variables available to the script: `spooler_log`, `JobScheduler`, `spooler_job` and `spooler_task`.	The JobScheduler makes the `spooler_set_context()` method of the com object available.
A script can be used reused by way of `<include>` at different stages when it is saved as a file	The class can be used at different stages. Seperate class objects are made.

Declaration in the JobScheduler Configuration

A job registered with a com class can be addressed by its class name:

```
<job name="delphijob">
    <script com_class="my_spooler_job_class"/>

</job>
```

The unique clSID can be specified instead of the class name:

```
<job name="delphijob">
    <script com_class="{F44FF458-D4DE-4cef-AA1A-CCC507346581}"/>

</job>
```

Direct DLL Specification

When a DLL requires a further DLL which cannot be loaded, then Windows stops the JobScheduler with a message box. The JobScheduler continues only after the ok button is clicked. Specification of the DLL is therefore not suitable for unsupervised use.

Should the com class not be registered, then the DLL can be directly specified. The class is then defined as a hexadecimal CLSID.

```
<job name="delphijob">
    <script com_class="{xxxx-xx-...}" filename="my_delphi.dll"/>

</job>
```

Example:

The com class is specified using the "Windows Script Component". The source is:

```
<?XML version="1.0"?>
<component>

    <registration

        progid      = "Joacim.Component"

        classid     = "{F44FF458-D4DE-4cef-AA1A-CCC507346581}"

        description = "Joacims Script Component"

        version     = "1"

    />


    <public>
```

```
        <method name="spooler_set_context"/>
        <method name="spooler_init"/>
        <method name="spooler_exit"/>
        <method name="spooler_open"/>
        <method name="spooler_process"/>
    </public>

    <script language="VBScript">
    <![CDATA[

dim spooler_log
dim spooler_task
dim i

function spooler_set_context(c)
    set spooler_log  = c.log
    set spooler_task = c.task
    spooler_log "Script component spooler_set_context"
end function

function spooler_init
    spooler_log "Script component spooler_init"
end function

function spooler_exit
    spooler_log "Script component spooler_exit"
end function

function spooler_open
    spooler_log "Script component spooler_open"
    i = 3
end function

function spooler_process
    spooler_log "Script component spooler_process i=" & i
    i = i - 1
    spooler_process = i > 0

end function


    ]]>

    </script>

</component>
```

This source code is written in the `job.wsc` file and registered as a COM-Server with the following command:

```
regsvr32 job.wsc
```

The server can be specified in the JobScheduler configuration with its class name:

```
<job name="component">
    <script com_class="Joacim.Component"/>

    <run_time once="yes"/>

</job>
```

or with its unique CLSID:

```
<job name="component">
    <script com_class="{F44FF458-D4DE-4cef-AA1A-CCC507346581}"/>

    <run_time once="yes"/>

</job>
```

Unix

Building a Unix port is possible in principle, despite com. The early binding would be used instead of the late binding through the IDispatch interface. This means that:

- the binding is made during compilation via header files instead of using `IDispatch`.
- the com class inherits fron the JobScheduler.Job class
- all requests are implemented. Requests are no longer optional. (The error code `E_NOTIMPL` can be returned)
- Depending on the available resources, the Delphi com support may or may not be usable. Should Delphi support not be available, then direct com requests will be used instead. This means that the return of an error message will be coded with `SetErrorInfo()`.
- the (shared object) module will not be registered, but will be directly specified using `<script filename="..."/>`.
- a job implementation modified for Unix will also run under windows. A `#ifdef` will most likely not be required. The job will run more quickly because of the early binding.

2.3 Locks

Locks in the JobScheduler stop the execution of a job, as long as a particular lock has been acquired by one or more jobs. Locks are available for individual jobs and for jobs in job chains. Should a job be waiting for a lock to be released (lock contention), then it will be automatically started as soon as the lock has been freed.

An example problem: jobs use a database, and a further job alters the database. However, this can only be done when the other jobs are not running.

2.3.1 Lock Configuration

Locks must first of all be declared before they can be acquired by jobs:

```
<config>

                                         <locks>

        <lock name="lock_name"/>
        <lock name="lock_name2"/>
        …
    </locks>
```

The locks to be used must also be declared for each job:

```
                                         <job>

    <lock.use lock="lock_name"/>
    <lock.use lock="lock_name2"/>
```

Locks can be used exclusively and non-exclusively. When a lock is used non-exclusively, it is possible to limit the number of non-exclusive acquisitions. The default setting for lock use is exclusive acquisition.

The JobScheduler does not start a job when a lock is used which has not been declared.

Careful: Lock names are case-sensitive!

A lock is acquired when a task starts and is freed when the task ends. An exclusive lock allows only one task.

Exclusive Locks

Exclusive locking is where a task which acquires a lock does not allow any other task access to the lock.

Declaration:

```
<config>

                                         <locks>

        <lock name="lock_name"/>
        …
    </locks>
```

Use:

```
                                         <job>

    <lock.use lock="lock_name"/>
```

2.3.2 Non-Exclusive Locking

A lock can be non-exclusively set and removed by more than one job.

The number of non-exclusive uses of a lock can be limited:

```
<lock name="lock_name" max_non_exclusive="2"/>
```

```
<job name="my_database_job">
    <lock.use lock="lock_name" exclusive="no"/> …</job>

<job name="my_other_database_job">
    <lock.use lock="lock_name" exclusive="no"/> …</job>

<job name="switch_database">
    <lock.use lock="lock_name"/> …</job>
```

The first two jobs can be run simultaneously, but cannot be run at the same time as the third job, which has exclusive use of the lock.

2.3.3 Locks in Job Chains

As locks are acquired by tasks, they can also be used in job chains, when the individual job node tasks end after execution. Ideally, in this situation, the `idle_timeout` attribute would be set to 0.

2.4 External Job Processing with Agents

Jobs can be processed by a JobScheduler running on a remote computer if it is configured as an Agent. Only Agents are able to receive and execute commands from a Main JobScheduler.

Characteristics of JobScheduler Agents:
- Agents do not require a live folder.
- Existing local jobs (if existing) are not executed. This is also true for JobScheduler default jobs such as *scheduler_event_service*.
- If the execution of a local job is suppressed a message will be logged.
- Internal jobs (*scheduler_file_order_sink* and *scheduler_web_service*) are not executed.
- Only Agents can receive and execute remote commands from a main JobScheduler. Trying to execute a remote command at a JobScheduler that is not configured as Agent, an error message will be logged.

Remotely processed jobs behave, with respect to the JobScheduler which is processing them, just the same as locally processed jobs. The only difference is that the processing load is transferred from the initiating to the processing JobScheduler.

This also means, for example, that all API calls refer to the local JobScheduler object.

The job log information, its end state and any possible error information will be forwarded to the initiating JobScheduler.

2.4.1 Application

In most cases an Agent will be installed on a different computer to the one on which the initiating JobScheduler is installed.

Example uses:
1. Using remote processing to balance load.
2. Software installations can be used which are not available on the computer on which the initiating JobScheduler is installed.
3. Hardware components such as printers can be used which are only available on a different computer to the one on which the initiating JobScheduler is installed.

2.4.2 Requirements for External Job Processing

1. Firewall Settings

   The initiating JobScheduler must be able to communicate with the Agent. All the firewall security settings must be set accordingly. So that communication to the Agent can take place, the port of the Agent must be opened in the firewall.

   In the opposite direction, task(s) must be able to communicate with the initiating JobScheduler using the port(s) <= 59999 counting downwards pro task being processed. The number of firewall ports which are opened must correspond with the maximum number of tasks expected.

2. Security Settings in the JobScheduler Configuration

   The Agent must obtain authorization from the initiating JobScheduler's security element.

   Example:

```
<security ignore_unknown_hosts="no">
    <allowed_host host="123.456.89.1" level="all"/>
</security>
```

3. Both JobSchedulers must be started.

2.4.3 Configuration

The job definition is saved in the configuration file of the initiating JobScheduler.

In order to be able to have a job processed on an Agent, the following configuration is required:

- A process class must be created and the `<process_class remote_scheduler="">` attribute set in this class. This causes all the jobs which are allocated to this process class, to be carried out remotely.

  Example:

```
<process_classes>
    <process_class name="remote"
                   max_processes="3"
                   remote_scheduler="remoteHost:4445"/>
```

```
</process_classes>
```

- The job must be allocated to the process class,

  see <job process_class="…">

2.4.4 Monitor Scripts

Any monitor scripts belonging to a job will be processed on the Agent. For example, this test script returns the name of the remote computer:

```
function spooler_process_before(){
    spooler_log.info("host:tcp_port:" + spooler.tcp_port);
    var  localhost = new java.net.InetAddress.getLocalHost();
    hostname = localhost.getHostName();
    ip = localhost.getHostAddress();
    spooler_log.info("==>" + hostname + ":::"+ ip);
    return true;
 }
```

2.4.5 The Context of API Calls

All API calls relate to the initiating JobScheduler. However some methods return values relating to the Agent and not to the initiating JobScheduler.

- Spooler.directory
- Spooler.log_dir
- Spooler.ini_path
- Spooler.include_path

2.4.6 Configuration Files

All settings are read by the initiating JobScheduler.

However, some settings are taken over from the Agent and not from the initiating JobScheduler:

- sos.ini (Section [java], entry javac=…)
- factory.ini (Section [spooler], entry tmp=…)
- <config java_options="…">
- <config java_class_path="…">
- <config include_path="…">

2.4.7 Log Files

Log files are saved in the files of the initiating JobScheduler. This applies to the order log as well as the task log. Output to `stdout` and `stderr` is written into the log files of the initiating JobScheduler. Other log output, which the JobScheduler writes, is written to the log output of the Agent.

3 Order processing and File Monitoring

3.1 Orders and Job Chains

An order is the instruction set describing how a job chain should execute a series of jobs. An order can be carried out immediately or when a preconfigured event takes place. Example events are times of the day, days of the week or files being added to a directory.

A job chain can be seen as an assembly line on which orders are run. The orders are processed by the individual machines of the assembly line one after the other. A job comprises exactly one step in the processing of an order. After a job has been completed, the state of the order is changed, which in turn affects the next step in the job chain. (For example, after completing one job, an order could be added to the queue for the next job.) This process will be repeated until the order reaches the end of the job chain.

An order has an identifier, which is valid within that job chain, and a readable title. The order also has a status, which changes after the processing of each job. An order can contain parameters, which are handed on to all the jobs in the job chain. It can also carry a load - i.e. an individual XML document, which it makes available to the jobs.

Orders allow jobs and job chains to be reused: an order can be allocated to a single job chain and define the time at which the job chain starts execution. This configuration is often used, when each job is only used in a single job chain and the parameters are set for each job.

It is also possible to configure more than one order for the same job chain. In this case, the orders can be given different parameters, which are then handed over to the jobs, so that the same jobs can be used for different purposes.

See the `Order` class.

Orders can also be transferred using TCP and the `<add_order>` command.

Further, the JobScheduler HTML interface (page 180) provides operations with which orders can be manually started.

3.1.1 Job Chains

Job chains define a series of jobs, whose execution is activated by an order. Job chains specify job dependencies for the successful and incorrect execution of the jobs. They can be restarted - that is, the JobScheduler saves the state of an order within a job chain in a database. Should the processing of an order be broken off at a particular job in a job chain, on restarting, the JobScheduler is able to restore the order at the job within the job chain, at which processing was broken off.

Job chains are described in the `Job_chain` and `Job_chain_node` classes.

A job is made order controlled using the `<job order="yes">` setting.

In order that a job can be used in a job chain, it should be defined using `<job order="yes">`. If this is not done, the jobs can be separately called up as standalone jobs.

3.1.2 Order Queue and Tasks

Every job which is order controlled has an order queue in which orders which are to be processed are collected. The JobScheduler starts processing an order as soon as it is added to the queue and the `<run_time>` element allows it to start. Should there be a number of orders in the job queue and the job allows multiple tasks, (`<job tasks="…">`), then the JobScheduler will process the tasks in parallel.

The JobScheduler hands an order over to a task by storing the order in the `Task.order` method and calling up the `spooler_process()` method. The `spooler_process()` method ends with either a `true` or `false` result, which, in turn, determines whether or not an order is processed by the next job in the job chain (see `Job_chain`).

3.1.3 Recognition of Double Orders

When an order is added to the job queue, the JobScheduler checks whether an order with the same `id` has already been entered (only, however, when the order `id` has been set). Should the order queue or job chain already have an order with the same `id`, then the already existing order will be overwritten. Should the new order have a different priority to the original, then the JobScheduler will take this into account.

3.1.4 Directory Monitoring with File Orders

See Directory Monitoring with File Orders (page 165) for information about orders which are created from the files in a directory.

3.1.5 Priority

Order priority is set using `Order.priority`. Orders with a higher priority are put by the JobScheduler at the front of the queue.

3.1.6 Ending Tasks

The JobScheduler lets tasks exist as long as the `<run_time>` parameter is valid. They then have the status `running_waiting_for_order`.

The maximum time which the JobScheduler will wait before a task is automatically ended is set using `<job idle_timeout="duration">`. This time should be set to a value such as 30 sec., so that resources can be set free.

Should the JobScheduler require to start a task for which no more process class resources are available, then it automatically looks for another task of the same process class with the `running_waiting_for_order` status, which it then terminates.

Tasks can be terminated using `Task.end()`, `<kill_task>` and `<modify_job cmd="end">`.

A task ends when `spooler_process()` is implemented. The order is then given the successor state.

3.1.7 Accelerated Order Processing

Jobs which are order controlled are prioritized. The `<job priority="…">` element contains an arbitrary value which defines the position of orders of the same status in the order queue. This value must not exceed the limit defined by the `<config priority_max="…">` attribute.

The JobScheduler gives orders a higher priority the further advanced they are in the job chain. This helps reduce congestion and means that jobs which are started first will be more quickly completed. It is as though the machines at the end of an assembly line were running more quickly than those at the beginning.

Jobs which are order controlled have, in general, priority over jobs which are not order controlled.

3.1.8 Database

More information about the longer term storage of order queues and the order history is to be found in the section on databases (page 136).

3.2 Directory Monitoring with File Orders

File orders can be used when a job chain is to process files from a directory. A file order is an order containing a reference to a file. The JobScheduler monitors a directory and creates a new file order for each new file in the directory.

```
<job_chain name="my_job_chain">
    <file_order_source directory="path"/>
    <file_order_source directory="other_path" regex="regex"/>
    <job_chain_node  state="100" job="job_1 error_state="error"/>
    <job_chain_node  state="200" job="job_2 error_state="error"/>
    <file_order_sink state="ok" remove="yes"/>
    <file_order_sink state="error" move_to="/errorpath.../ "/>
</job_chain>
```

See `<job_chain>` (page 47), `<file_order_source>` (page 35) and `<file_order_sink>` (page 34).

3.2.1 File Order Sources

A file order source `<file_order_source>` is used to monitor a directory. A file order is created when a file with a name corresponding to the optional regular expression is added to the directory.

```
<job_chain …>
    <file_order_source directory="…" regex="…"/>
    …
```

A file order is an `Order` with the following properties:

`Order.state`
: The status of the order is determined either by the status of the first job in the job chain or the job with the `<file_order_source next_state="…">` state.

`Order.id`
: The order identifier is a path, consisting of the directory name, as specified in the order source and the file name.

`Order.params`. `Variable_set.value()`
: contains the path, the same value as `Order.id`. This variable, which is reserved for the JobScheduler, defines an order as a file order.

3.2.1.1 Execution Sequence for File Orders

The oldest file (defined as that with the oldest last change date) is handled first.

3.2.1.2 Multiple Order Sources

A job chain can have a number of order sources. The JobScheduler handles all orders in chronological order as described above.

3.2.2 File Monitoring With a File Order

The JobScheduler removes a file order when the corresponding file is no longer in the monitored directory and:

- the file order has not been handed over to a job

  `[warn]` `SCHEDULER-982` File has been removed, so the file order is being removed too

- the file order is blacklisted

  `[info]` `SCHEDULER-981` File on blacklist has been removed

File orders being executed are not affected.

This check is carried out when:,

- the JobScheduler reads the directory being monitored,
- it hands the file order over to a job for the first time.

3.2.3 File Order Sink: Removes or Moves a File

After an order has been completed, the corresponding file can either be moved or removed. States defined `<file_order_sink>` are end states.

Should, however, the file already have been removed, the JobScheduler issues a warning and the file order is completed.

`[warn]` `SCHEDULER-339` File does not exist and can therefore neither be moved nor removed:

Should it not be possible to either move or remove the file, then it is added to the blacklist.

3.2.3.1 Moving a File

```
<file_order_sink state="…" move_to="directory_path">
```

A file in the target directory having the same name will be overwritten without warning.

3.2.3.2 Removing a File

```
<file_order_sink state="…" remove="yes">
```

3.2.4 Blacklist

Should the file still exist after a file order has been completed (and reached an end state), then the JobScheduler sets the order on the blacklist.

```
[ warn]     SCHEDULER-340     File still exists. Order has been set on the blacklist
```

It remains there until:

- the JobScheduler determines that the file has been removed from the directory

  ```
  [ info]     SCHEDULER-981     File on blacklist has been removed
  ```

- the `<remove_order>` command has been carried out.

This stops the file immediately creating a new file order.

3.2.5 Directory Errors

Should an error occur when a directory is being monitored - for example, because the directory has been uncoupled - then the JobScheduler gives out a warning and sends an e-mail according to the settings in `factory.ini` (section `[spooler]`).

The JobScheduler regularly attempts to restart the monitoring. In doing so it ignores error messages. The intervals in which it does this is set using `<file_order_source delay_after_error="…">`. Should a directory become readable once more, then the JobScheduler sends an appropriate e-mail and the following message:

```
[ info]     SCHEDULER-984     Recovered from previous error in directory
```

3.2.5.1 Errors While Creating a File Order

Should it not be possible to create the corresponding file name for an order - (because the path is too long for the database column) then the JobScheduler notes the path in order to avoid repeating the error message each time the directory is read. It then continues after issuing the following message:

```
[ warn]     SCHEDULER-346     Due to previous error the path will be ignored:
```

3.2.6 When is a Directory Read?

As soon as the first job to create an order is ready to be carried out for the first time, the JobScheduler will read the relevant directory (within any constraints set using regular expressions) and creates the file orders

Should the directory contain more files than allowed by `<file_order_source max="…">`, then a list of the rest files is held by the JobScheduler in memory. This list is then used to create the remaining orders at a later point in time. The relevant messages here are:

The JobScheduler re-reads a directory when:

- all the files from the first job have been handed over,
- the JobScheduler is ready for a further order,
- and the period set by `<file_order_source repeat="…">` has expired.

3.2.6.1 Directory Monitoring on Microsoft® Windows®

The JobScheduler is able use the operating system to monitor directories on Windows systems. A signal causes the JobScheduler to read the directory before the end of the period. This means that the JobScheduler can immediately react to a new file.

This does not mean that monitoring of a directory on another computer can be switched off using the setting `<file_order_source repeat="no">`. The Windows directory monitoring stops when a directory is removed and recreated. See also Microsoft's Article 188321: FindFirstChangeNotification May Not Notify All Processes on File Changes. The inquiry interval should be used to regularly restart the monitoring of a directory.

3.2.7 Order Controlled Non-API Jobs (<script language="shell">)

The JobScheduler API (Application Programme Interface) is a way of allowing program code to control the JobScheduler. XML jobs, or jobs which cause an executable file to be started are examples of non-API jobs - i.e. jobs which do not use the API.

The file path is contained in the environment variable.

3.3 Directory Monitoring

A job which processes files laid up in a directory can be used for directory monitoring. The request `Job.start_when_directory_changed()` causes the JobScheduler to start a task as soon as a change occurs in a directory.

Requests can be repeated for a number of directories.

A request can be included in the JobScheduler script (Element `<script>` in `<config>`) or in a Job itself (in `spooler_init()`). In the latter case, the job itself must be run before the request can be implemented (see `<run_time once="yes">`).

3.3.1 Definition of a Directory Change:

When a file or a subdirectory is added, deleted or renamed.

3.3.2 The Regular Expressions Filter

The method `Job.start_when_directory_changed()` is used to define a regular expression (according to POSIX 1003.2). In this case the JobScheduler only sees a directory as having been changed when, after the change, it contains a file or subdirectory whose name fits the regular expression.

3.3.3 The Directory in Which a Change has Taken Place

When a number of directories are being monitored, the `Task.changed_directories()` request returns the names of the changed directories, separated by semicolons.

3.3.4 The Files in the Directory

`Task.trigger_files` returns the paths corresponding to the regular expression (if present). These paths are put together from the directory and file name.

For non-API jobs (`<script language="shell">`) the paths are provided in the environment variable `SCHEDULER_TASK_TRIGGER_FILES`.

3.3.5 Errors when Directory Monitoring

The JobScheduler interprets the situation of a directory no longer being accessible (i.e. it had been deleted or is no longer available) as a change and starts a task.

A repeated `Job.start_when_directory_changed()` request leads to an error.

3.3.6 Example

The following example shows an example configuration for a job which processes all the files in a directory and then deletes them. (Careful when testing - the job really deletes files!)

```
<?xml version="1.0"?>
<spooler>
    <config>
        <process_classes/>

        <jobs>
            <job name="import">

                <script java_class="spooler_job.Import"><![CDATA[

                    package spooler_job;

                    import java.io.File;
                    import sos.spooler.*;

                    public class Import  extends sos.spooler.Job_impl
                    {
                        String[]  paths;
                        int       index;


                        public boolean spooler_open()
                        {
                            spooler_log.info( "changed_directories=" +
spooler_task.changed_directories() );
                            spooler_log.info( "trigger_files=" +
spooler_task.trigger_files() );

                            paths = spooler_task.trigger_files().split( ";" );
                            index = 0;
                            return index < paths.length;
                        }
                        public boolean spooler_process()
                        {
                            File file = new File( paths[ index++ ] );
                            spooler_log.info( "Processing file " + file );
                            file.delete();
                            return index < paths.length;
                        }
                    }
                ]]></script>

            </job>
        </jobs>
    </config>
</spooler>
```

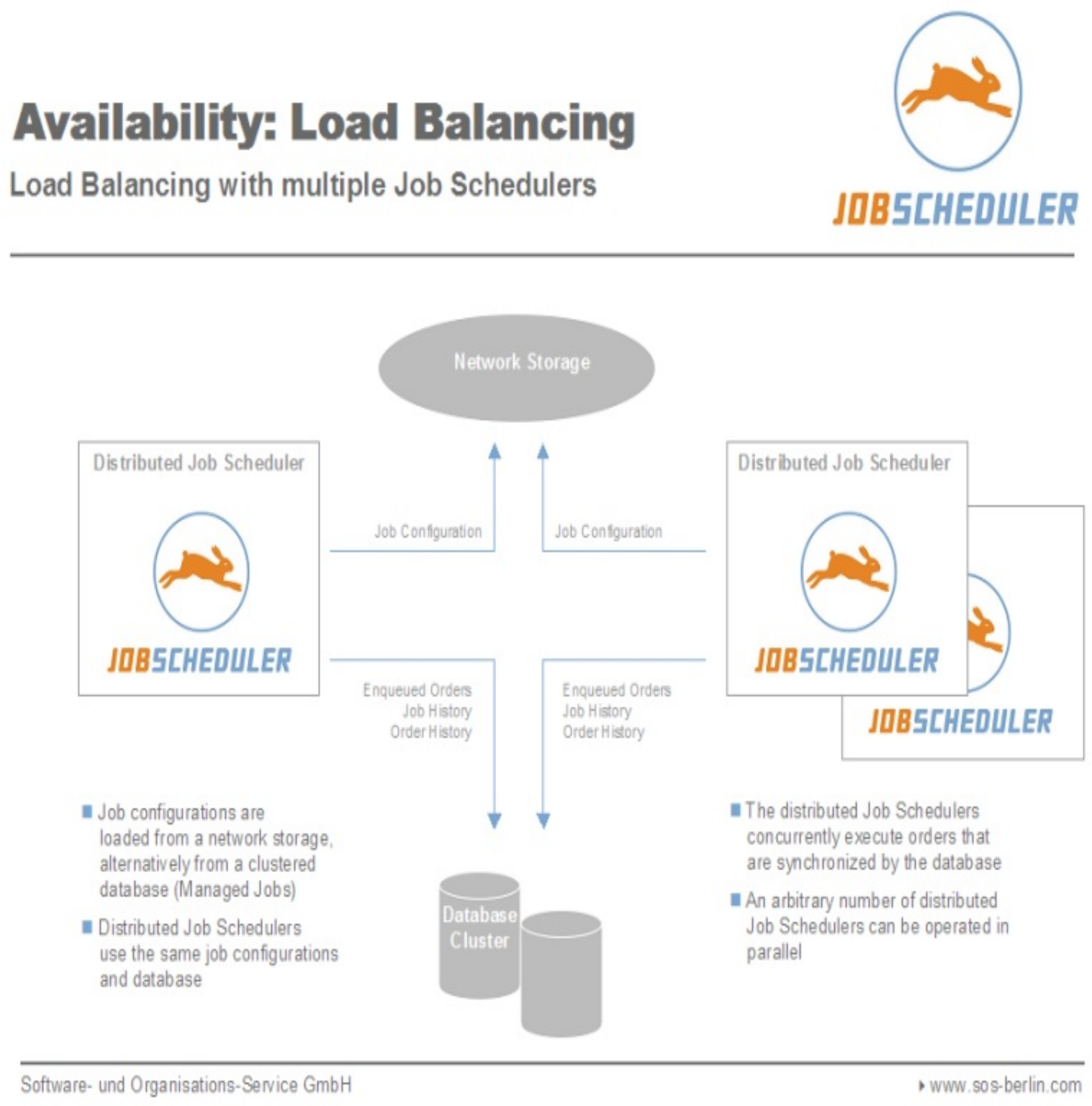
3.4 Distributed Orders

3.4.1 Distributed Order Processing

A JobScheduler cluster can be used to distribute orders for processing on more than one node. This feature can be used for load balancing - reducing processing times by using more hardware to process an order.

All the JobSchedulers in a cluster indicate their availability by sending out heartbeats (to the database). At the same time they check the availability of all the other JobSchedulers in the cluster. Should a JobScheduler determine that the heartbeat of another JobScheduler has been missing for a longer period of time (approx. 1-2 minutes), then it takes over the processing of the orders started by the missing JobScheduler.

The conditions under which a JobScheduler cluster can be used for distributed orders are schematically represented on the following diagram and described in the next section.



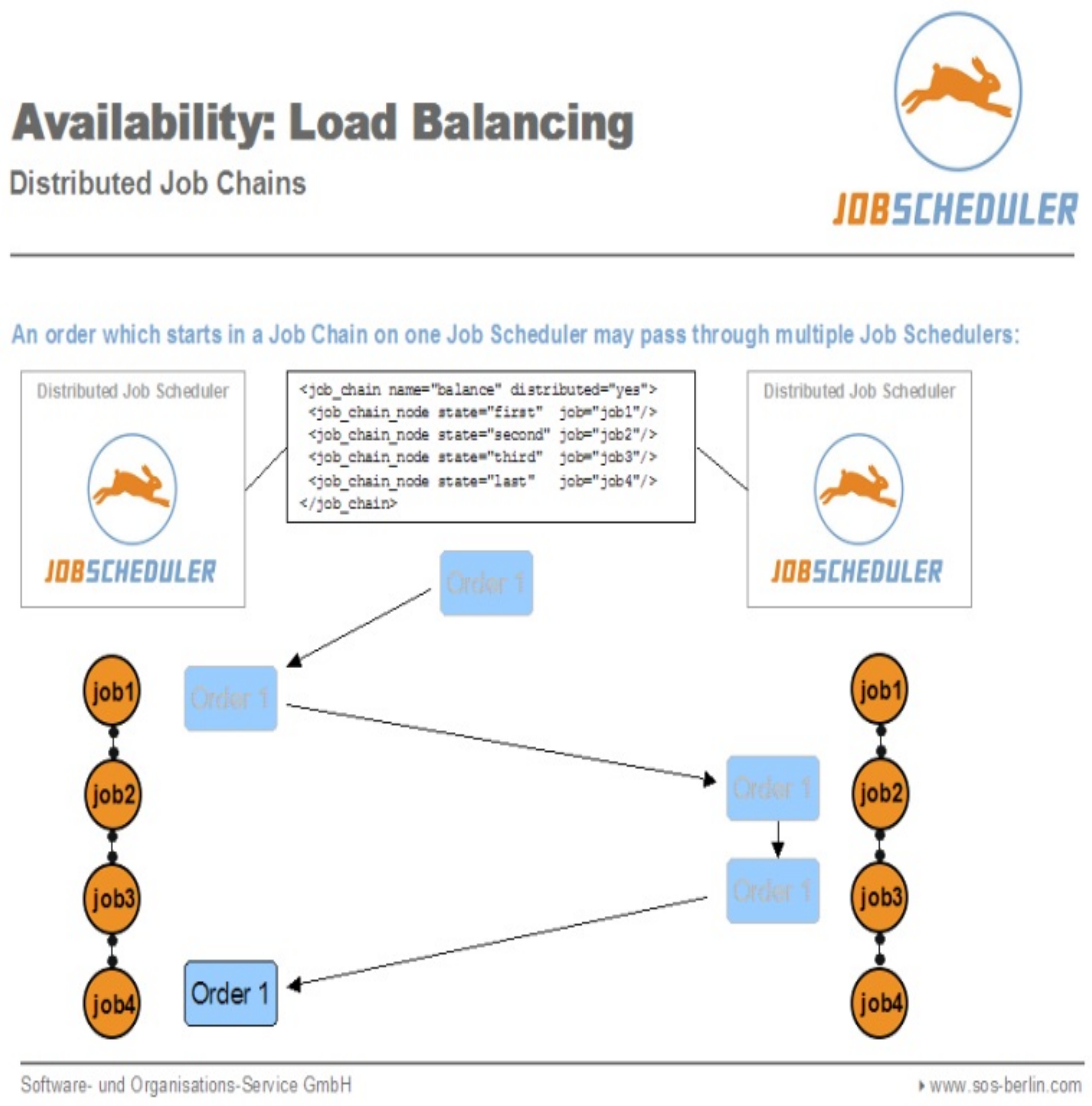
Availability: Load Balancing

Load Balancing with multiple Job Schedulers

- Job configurations are loaded from a network storage, alternatively from a clustered database (Managed Jobs)
- Distributed Job Schedulers use the same job configurations and database

- The distributed Job Schedulers concurrently execute orders that are synchronized by the database
- An arbitrary number of distributed Job Schedulers can be operated in parallel

Distributed orders can be processed from different JobSchedulers on their way along a job chain. The handing over of the processing of an order to another job node on another JobScheduler is schematically illustrated in the following diagram:


Availability: Load Balancing
Distributed Job Chains
JOBSCHEDULER
An order which starts in a Job Chain on one Job Scheduler may pass through multiple Job Schedulers:
Distributed Job Scheduler
Distributed Job Scheduler
Order 1
job1
job2
job3
job4
Software- und Organisations-Service GmbH
www.sos-berlin.com

```
<job_chain name="balance" distributed="yes">
 <job_chain_node state="first"  job="job1"/>
 <job_chain_node state="second" job="job2"/>
 <job_chain_node state="third"  job="job3"/>
 <job_chain_node state="last"   job="job4"/>
</job_chain>
```

3.4.1.1 The Requirement for using Distributed Orders

- All the JobSchedulers in the cluster must use the same database.
- The JobSchedulers must all use the same configuration file or an exact copy.
- All the JobSchedulers must be started with the same JobScheduler ID.
- The JobSchedulers must all be started with the `-distributed-orders` option.
- All the JobSchedulers must have access to any resources required by jobs run on a distributed job chain - for example, monitored directories. Directories which are used by all the JobSchedulers must be mounted (or linked) on the same path on all systems.

3.4.1.2 Starting a JobScheduler Cluster for Distributed Orders

The JobSchedulers making up the cluster can be started in any order. Any JobScheduler belonging to the cluster can be removed or added as required whilst the cluster is in operation. When a JobScheduler is removed from a cluster it should, where possible, be stopped (terminated) normally, to allow any orders which are being processed at the time to be completed.

3.4.1.3 Generation of Distributed Orders

Distributed Orders can either be generated by the `<add_order>` command or by directory monitoring. A persistent order can not be used as a distributed order.

3.4.1.4 Distributed Orders Using add_order

Either the `<add_order>` eAPI command or the `Job_chain.add_order()` iAPI function can be used. Neither the JobScheduler to which the command is sent to nor the JobScheduler on which the command is carried out is important - the order will be made available to the whole cluster for processing.

3.4.1.5 Distributed File Orders

Distributed File Orders are configured in a distributed job chain using `<file_order_source>`. Each JobScheduler in the cluster monitors the relevant directory or directories and can create file orders. A file order can be processed by a different JobScheduler to the one that created the order.

3.4.1.6 Stand-Alone Jobs with Distributed JobSchedulers

Independent jobs which do not process orders are processed in a JobScheduler cluster by the JobSchedulers with which they were configured.

3.4.1.7 Load balancing methods

The method used for load balancing can be set using the scheduler.order.distributed.balanced global parameter:

- **Overload balancing**:

  With this method, an additional JobScheduler of the cluster takes over the execution of a distributed order when the JobScheduler initially processing the order is working at maximum capacity - i.e. the allowed number of processes reaches a maximum value. This is the standard balancing method used by the JobScheduler.

  see also `<job tasks="…">` (page 42) and `<process_class attribute max_processes="…">` (page 68).

- **Equal distribution**:

  With this method, distributed order tasks are allocated equally to the JobSchedulers in a cluster.

4 Protocols and Forwarding of E-mails

4.1 Logs

The directory in which the JobScheduler writes its log files is specified using the `-log-dir` option.

Entries are numbered - see »Messages« (page 234)

Main Log Files

All log files are summarized in the main log file. The name of the main log file is made up from the scheduler id, the date and the time thus: `scheduler-2004-08-24-104111.log`.

Job Protocols

The JobScheduler records a job log file containing the task start information and the orders processed - for example: `job.my_job.log`

The JobScheduler changes characters in job names, which cannot be used in file names, into "_".

Task Protocols

A task log file is written for every task the JobScheduler starts. The name of the task log file is made up of the job name and, should more than one task be allowed at once, the task id. E.g. `task.my_job.log` and `task.a_job.1234.log`. Note that task log files with a task-id are deleted after completion of the task.

Order Protocols

The JobScheduler compiles an order log file containing the results of all tasks processed in the course of an order. The name of this log file is made up from the name of the job chain and the order id - e.g.: `order.my_job_chain.9876.log`. Order log files are deleted after individual orders are completed.

scheduler.log

The JobScheduler can write the `scheduler.log` as a debugging aid. The name of this log file is set using the `factory.ini` (section `[spooler]`, entry `log= …`) parameter. "`factory.ini`" is used in this documentation for the file name entered here.

For further information see the List of Log Categories.

4.1.1 Log File Size

The detail with which log files are noted (error, warn, info, debug to debug9) can be set using `-log-level` and `Log.level`

4.1.2 Message Codes

Messages are allocated codes, such as `SCHEDULER-900`. All codes can be found in the List of JobScheduler Message Codes (page 234) .

4.1.3 Full Disc Drive

An error occurs when insufficient disc space is available when a log file is opened. Should it not be possible to open the main log file then the JobScheduler aborts.

The JobScheduler stops should insufficient disc space be available to continue writing a log file. However, it will still respond to TCP, UDP or HTTP commands

In this situation, the `<modify_spooler>` command will only be carried out to a limited extent.

The JobScheduler can be made to continue using the `<modify_scheduler cmd="continue"/>`, once disc space has been made available.

The above procedure does not apply to the `scheduler.log` file, for which errors are ignored.

4.1.4 Database Storage of Protocols

The JobScheduler can store task and order log files in the history under the following conditions:

- operation with a database (page 136),
- activation of the history function using the `factory.ini` (section `[spooler]`, entry `history= …`)), setting,
- use of the `factory.ini` (section `[spooler]`, entry `history_with_log= …`) setting for the Task Protocol,
- use of the setting `factory.ini` (section `[job]`, entry `history_with_log= …`) for distinct jobs,
- use of the `factory.ini` (section `[spooler]`, entry `order_history_with_log= …`) setting for the Order Protocol.

4.1.5 Program Interface

The log files and settings available are set using the `Log` class.

4.1.6 E-mail Forwarding

Task log files can be sent per e-mail. Further details can be found in e-mail (page 175).

4.1.7 Protocol Display

See the `<show_state>` (page 210), `<show_task>` (page 211) and `<show_history>` (page 206) commands.

Protocols can be viewed real-time when the JobScheduler HTTP-Server (page 180) is called up using a web browser.

4.2 Sending E-mails

4.2.1 E-mail Settings

General E-mail settings are made as follows: `factory.ini` (section `[spooler]`, entry `smtp= …`)

factory.ini (section [spooler], entry log_mail_subject= …)

factory.ini (section [spooler], entry log_mail_from= …)

factory.ini (section [spooler], entry log_mail_to= …)

factory.ini (section [spooler], entry log_mail_cc= …)

factory.ini (section [spooler], entry log_mail_bcc= …)

and settings for particular jobs are made thus: factory.ini (section [job], entry log_mail_subject= …)
factory.ini (section [job], entry log_mail_from= …)

factory.ini (section [job], entry log_mail_to= …)

factory.ini (section [job], entry log_mail_cc= …)

factory.ini (section [job], entry log_mail_bcc= …)

Or in the application programming interface:
The spooler_log class provides the Log.mail object, which in turn makes the Mail object available. This allows sender, recipient, re., etc. to be set using the methods of the Mail object.

Example (in Java)

```
spooler_log.mail().set_to( "admin@xxx.com" );
```

4.2.2 E-mails Sent after Task Completion

On completion of a task, the JobScheduler can send an E-mail containing the task protocol.

The following settings are used to determine the general conditions under which a protocol should be sent on completion of a task: factory.ini (section [spooler], entry mail_on_success= …)
factory.ini (section [spooler], entry mail_on_process= …)

factory.ini (section [spooler], entry mail_on_error= …)

The following settings are used to determine the conditions for a particular job: factory.ini (section [job], entry mail_on_success= …)
factory.ini (section [job], entry mail_on_process= …)

factory.ini (section [job], entry mail_on_error= …)

and in the application programming interface:
The Log.mail_it request is used to determine whether the JobScheduler sends a protocol on completion of a task.

4.2.3 Settings Priorities

1. The job script uses the Mail class to make settings.
2. Should an error occur, the JobScheduler overwrites the mail subject - i.e. the job Mail.log_mail_subject setting.
3. The e-mail XSLT stylesheet (<config mail_xslt_stylesheet="…">), can make used to make the settings.

4. Empty settings are filled with values from

   factory.ini (section [job]) and

   factory.ini (section [spooler]).

4.2.4 E-mails Sent when the JobScheduler Terminates because of an Error

The settings for e-mails sent when the JobScheduler terminates because of an error are to be found in section [spooler] of the factory.ini file.

Should a error occur before the -ini option can be carried out, the JobScheduler uses the (default) settings from the factory.ini file under its original name.

When running as a service or daemon the JobScheduler will send an e-mail containing the relevant error message, should an error occur which is so serious that the JobScheduler must abort.

The JobScheduler sends an e-mail should a database error occur. Further e-mails about errors caused by subsequent attempts to reopen the database are not sent.

An e-mail is also sent should the JobScheduler have to stop using a database after an error.

4.2.5 Installation with JavaMail

The following files must be added to the class_path directory when an e-mail is to be sent using JavaMail.

sos.mail.jar	(is delivered with the JobScheduler)
mail.jar	(Sun Microsystems, Inc.)
smtp.jar	(Sun Microsystems, Inc.)
activation.jar	(Sun Microsystems, Inc.)

See sos.ini (section [java], entry class_path= …).

4.3 Log Categories

category	default¹	
all		All categories (with exception of explicit categories)
com		COM-Operations
com.invoke		Calling a COM object method
env		Environment variables
exception		Particular error codes
exception.*	default	All error codes
exception.D310	explicit	End of file
exception.D311	explicit	Expression not found
factory		DocumentFactory

category	default[1]	
factory.parameter		
factory.processor		Factory Processor (hostole.dll)
file		
file.directory		
file.mmap		mmap()
hostole		HostOLE calls (with exception of the factory processor)
ini		Windows call GetPrivateProfile() Read the .ini file: `GetPrivateProfile()`.
java		Java
jdbc		JDBC Calls
mail		
mutex	explicit	Mutual exclusion locks, blocking for thread serialisation
object_server		JobScheduler interface for tasks running in their own processes
object_server.call		Method Call
object_server.continue		
object_server.Invoke		COM operation
object_server.push		Stack-Operation for nested method calls
object_server.QueryInterface		COM operation
object_server.wait		
odbc		ODBC calls
rtf		RTF Processor
rtf.learn		Learned (unknown) RTF codes
rtf.map_next_line		Positions of script line numbers found in RTF documents
rtf.merge		Merge different templates
scheduler	implicit	JobScheduler
scheduler.call	implicit	Start and end call of a job method, e.g. spooler_process()
scheduler.cluster		Cluster operation
scheduler.directory		Open a directory
scheduler.file_order		<file_order_source>
scheduler.http		HTTP server
scheduler.log		Access to protocol files
scheduler.nothing_done		When a job is idle

category	default¹	
scheduler.order	implicit	Operations on orders (temporary payload setting)
scheduler.service	implicit	Windows service controller
scheduler.signal		
scheduler.wait		The JobScheduler enters the waiting state
scheduler.xml		XML/DOM operations
socket		Socket operations (network)
socket.accept		System call accept(): accept TCP connections
socket.close		System call close(): close socket
socket.connect		System call connect(): establish TCP connection
socket.data	explicit	Data from recv() and send()
socket.listen		System call listen(): waiting for TCP connection request (listening)
socket.recv		System call send(): receive file
socket.select		System call select(): waiting for TCP result
socket.send		System call send(): send file
socket.setsockopt		System call setsockopt()
socket.shutdown		System call shutdown(): end connection
sossql		SQL-Processor (file type sossql)
sossql.get_key		Operation get_key()
spidermonkey		The Spidermonkey JavaScript implementation
spidermonkey.callback	explicit	Spidermonkey call backs
spidermonkey.idispatch	explicit	COM object management
windows		Betriebssystem Microsoft Windows
windows.PeekMessage		Aufruf PeekMessage()

Many of the entries made in the log file (`scheduler.log`) are categorised. These categories are used to regulate whether or not an entry is made in a log file.

A log category is specified in front of the file name, with a ">" symbol inserted between the category and the file name. Multiple categories are separated by an empty space. For example (in an .ini file): `log = scheduler.* socket.* >c:/tmp/scheduler.log`.

The special category `all` selects all categories other than those which must be explicitly selected.

Log file categories are organised in hierarchies. Categories can be selected together with their subcategories by appending an asterisk ("`.*`") to the category name thus: `scheduler.*`.

See also the commands:

5 Communication and Operation

5.1 HTTP Server and Web Services

5.1.1 Web Services

The JobScheduler can encapsulate the execution of jobs and job chains as web services. To do this, the JobScheduler responds to SOAP queries received by way of its own built-in HTTP server.

Web services are installed using the `<web_service>` element.

The use of the JobScheduler as a web service, together with example configurations is described in the »Web Service Tutorial«.

5.1.2 Operation with a Browser

The JobScheduler can be operated using its own built-in HTML interface and its own HTTP-Server in conjunction with a standard browser (Microsoft Internet Explorer and Firefox). This interface is accessed using the address `http://localhost:4444`, where `localhost` can also be another computer name and `4444` the TCP port number configured for the JobScheduler.

See `<config tcp_port="…">` (page 21).
See `factory.ini` (section `[spooler]`, entry `html_dir= …`) (page 100).

The user interface is described in the "Built-In User Interface" section of this chapter (below).

5.1.3 Security

The JobScheduler only allows TCP and HTTP connections to computers which have been given appropriate permissions with the `<allowed_host>` parameter.

In addition, HTTP authentication can arranged using the `<http.authentication>` element.

5.1.4 Show Protocols in a Browser

A protocol can be viewed whilst it is being written by the JobScheduler. The current status is always shown. The following URLs are used:

The functions necessary for this are provided by the JobScheduler's HTML interface. The JobScheduler uses the following URLs for protocols:

main protocol	`http://`*`localhost:4444`*`/show_log?`
job protocol	`http://`*`localhost:4444`*`/show_log?job=`*`jobname`*
task protocol	`http://`*`localhost:4444`*`/show_log?task=`*`task_id`*
order protocol	`http://`*`localhost:4444`*`/show_log?job_chain=`*`jobchain`*`&order=`*`order_id`*

where *localhost* is `localhost` or 127.0.0.1 or the hostname of the server on which the JobScheduler is running.

5.1.5 Job Descriptions

The job description from `<description>` can be called up using the following URL:

`http://localhost:4444/job_description?job=job`

Note that it is assumed here that the description is coded in HTML.

5.1.6 Built-In Graphical User Interface

The JobScheduler can be operated using its own, built-in, web-based graphical user interface (GUI). Instructions for accessing this interface were provided in the "Operation with a Browser (page 180)" section of this chapter (above).

This interface is automatically installed and updated with the JobScheduler installation package.

The JobScheduler GUI is configured in the `custom.js` datei, which is to be found in the `JobScheduler installation directory\config\html` directory

This interface is intended for the *operation* of the JobScheduler - i.e. starting, monitoring and stopping the JobScheduler itself, jobs & job chains, orders, and any locks and JobSchedulers operating in a cluster.
This interface should not be confused with the JobScheduler Editor (page 120), which is a GUI used to *configure* the JobScheduler itself, jobs & job chains, orders using XML. (Note that the JobScheduler Editor can be called up from the JobScheduler GUI. This is described below.)

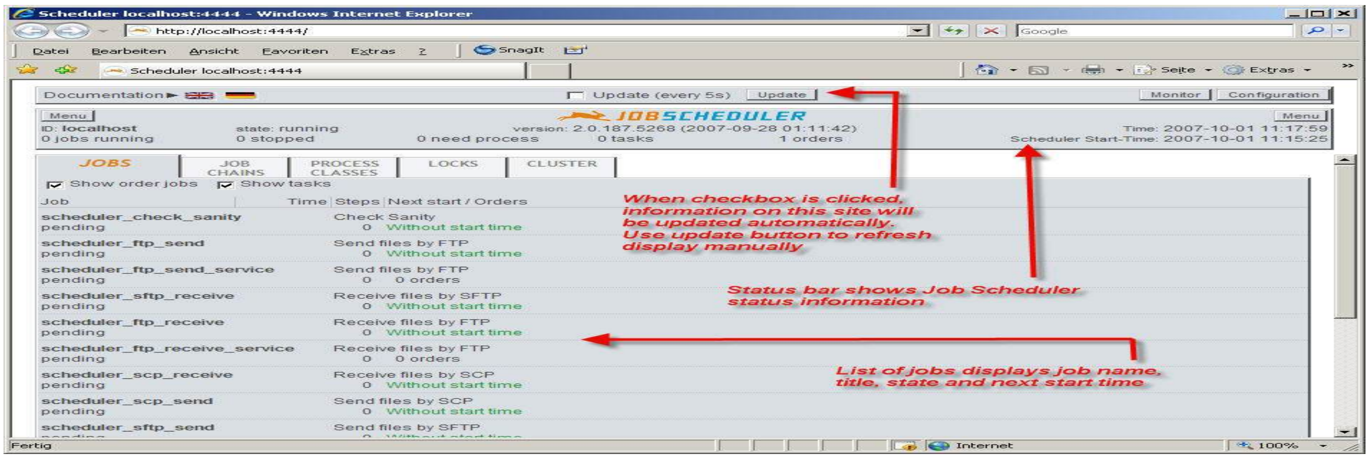
The JobScheduler GUI should also not be confused with the Managed Jobs Administration interface, which is a part of the JobScheduler *Managed Jobs* package. The Managed Jobs Administration interface is a GUI for the creation and monitoring jobs, job chains & orders and comes with a user administration. Further information about the Managed Jobs Administration interface can be found in the »Managed Jobs Documentation«.

When first opened, the JobScheduler GUI appears as shown in the screenshot below and comprises three areas:

- An upper menu bar, where the general configuration of the interface (language & update) is carried out; any monitoring functions the JobScheduler is to carry out is shown and the configuration of the JobScheduler itself in the XML configuration file (page 6) is shown.

  Note that here the XML configuration file is shown "read-only" - to change the configuration, the JobScheduler Editor (page 120) or an alternative XML editor should be used which has access to the JobScheduler installation directory.
- A central menu and status bar, providing access to the most important JobScheduler menus and status information about the operation of the JobScheduler.

  The "Update periodically" checkbox and the "Update" button are used to update the information shown in the web interface either at regular intervals (when the checkbox is selected) or manually, as required (using the button). The interval with which the regular updates of the interface are carried out is set using the Settings dialog, which is accessed via the "Extras" button.
- A working area initially showing a list of the jobs configured for the JobScheduler, which can be used to provide information about other aspects of the JobScheduler operation such as the job chains, process classes and locks by selecting the appropriately named menu tabs. Note that, should, for example, no locks be configured then the "Locks" menu tab will not be shown.



ID: localhost state: running version: 2.0.187.5268 (2007-09-28 01:11:42) Time: 2007-10-01 11:17:59
0 jobs running 0 stopped 0 need process 0 tasks 1 orders Scheduler Start-Time: 2007-10-01 11:15:25

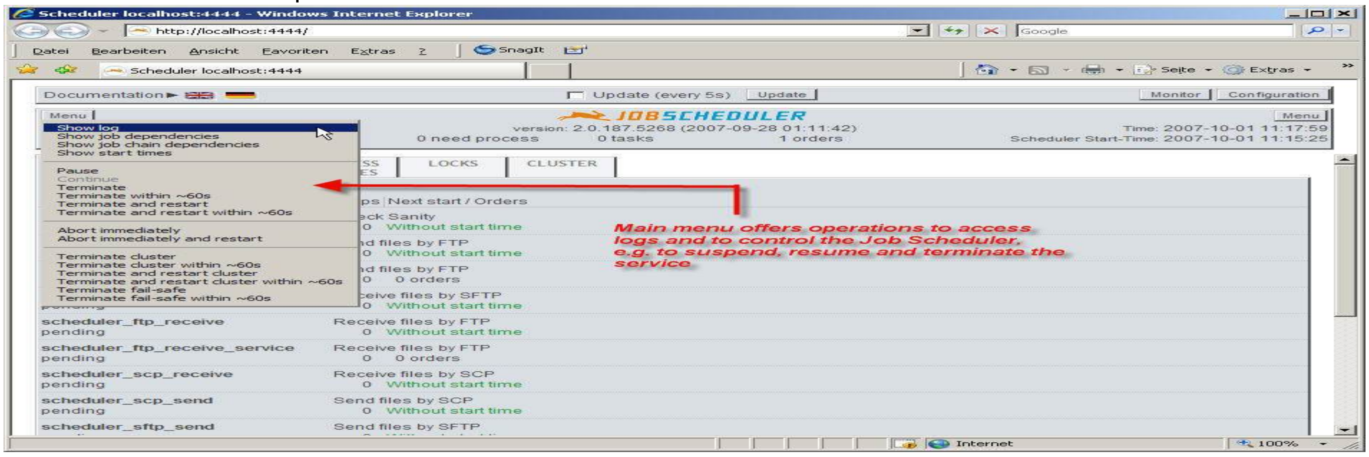
```
Job                              Time | Steps | Next start / Orders
scheduler_check_sanity                 Check Sanity
pending                                0   Without start time
scheduler_ftp_send                     Send files by FTP
pending                                0   Without start time
scheduler_ftp_send_service             Send files by FTP
pending                                0   0 orders
scheduler_sftp_receive                 Receive files by SFTP
pending                                0   Without start time
scheduler_ftp_receive                  Receive files by FTP
pending                                0   Without start time
scheduler_ftp_receive_service          Receive files by FTP
pending                                0   0 orders
scheduler_scp_receive                  Receive files by SCP
pending                                0   Without start time
scheduler_scp_send                     Send files by SCP
pending                                0   Without start time
scheduler_sftp_send                    Send files by SFTP
```

When checkbox is clicked, information on this site will be updated automatically. Use update button to refresh display manually

Status bar shows Job Scheduler status information

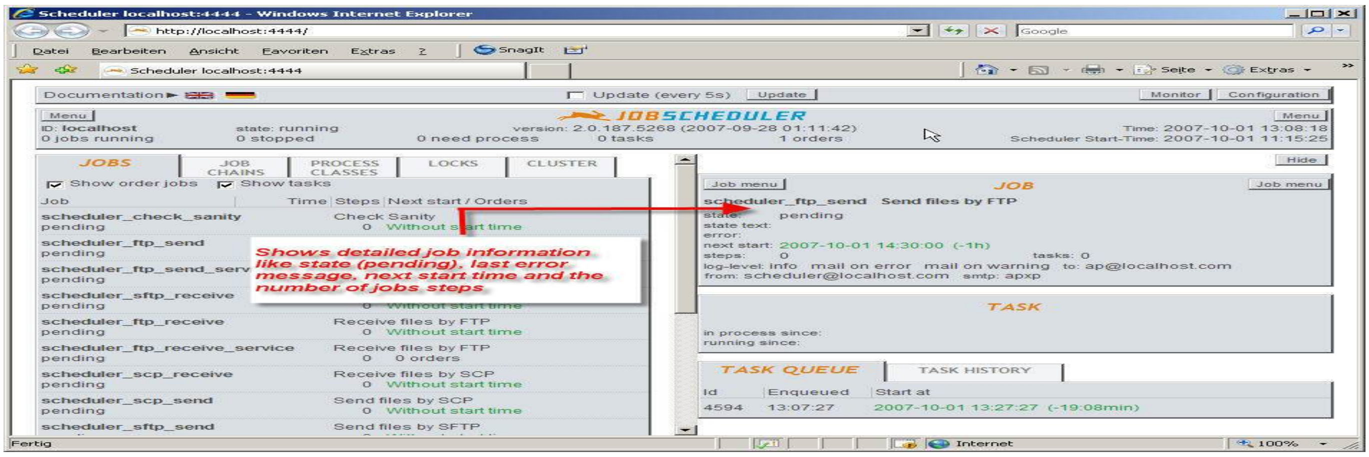
List of jobs displays job name, title, state and next start time

The main menu for the operation of the JobScheduler is shown in the next screenshot:


Menu
Show log
Show job dependencies
Show job chain dependencies
Show start times
Pause
Continue
Terminate
Terminate within ~60s
Terminate and restart
Terminate and restart within ~60s
Abort immediately
Abort immediately and restart
Terminate cluster
Terminate cluster within ~60s
Terminate and restart cluster
Terminate and restart cluster within ~60s
Terminate fail-safe
Terminate fail-safe within ~60s

Main menu offers operations to access logs and to control the Job Scheduler, e.g. to suspend, resume and terminate the service

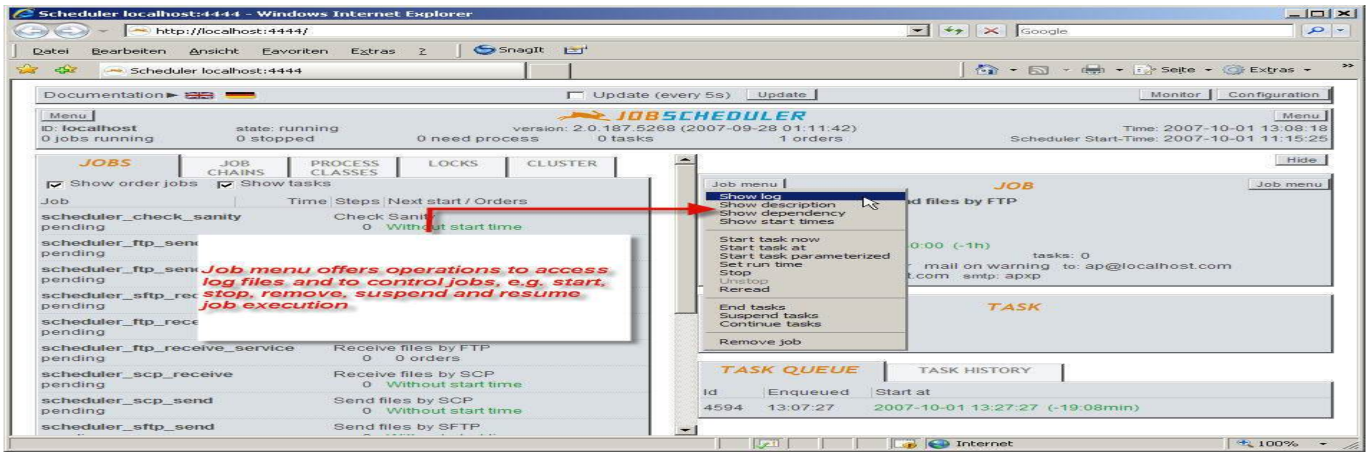
Detailed information about the state of one of the jobs listed in the main working area of the interface can be obtained by clicking on the name of the job. This causes the right hand part of this area to divide and additional areas showing status information about the job, about the task and about the taks queue to be displayed as can be seen in the screen shot below:



```
scheduler_ftp_send   Send files by FTP
state:      pending
state text:
error:
next start: 2007-10-01 14:30:00  (-1h)
steps:      0                               tasks: 0
log-level: info   mail on error   mail on warning   to: ap@localhost.com
from: scheduler@localhost.com   smtp: apxp

in process since:
running since:

Id     Enqueued    Start at
4594   13:07:27    2007-10-01 13:27:27  (-19:08min)
```

Shows detailed job information like state (pending), last error message, next start time and the number of jobs steps

The job status area also contains a menu providing the main commands necessary for controlling the job as shown in the next screen shot:

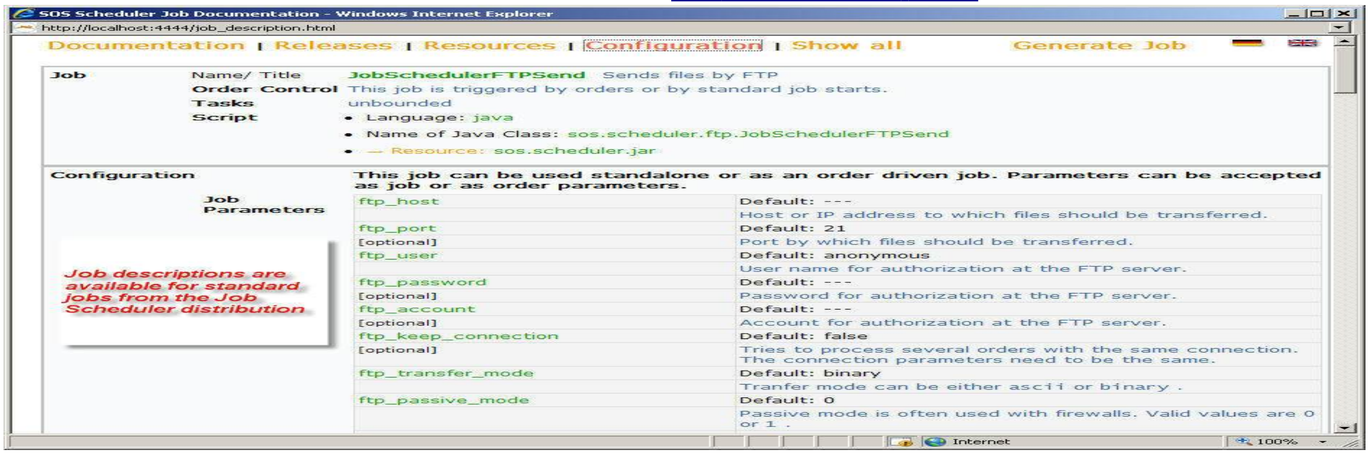


Job menu offers operations to access log files and to control jobs, e.g. start, stop, remove, suspend and resume job execution

```
Job                          Time  Steps  Next start / Orders
scheduler_check_sanity             Check Sanity
pending                            0  Without start time
scheduler_ftp_receive_service      Receive files by FTP
pending                            0  0 orders
scheduler_scp_receive              Receive files by SCP
pending                            0  Without start time
scheduler_scp_send                 Send files by SCP
pending                            0  Without start time
scheduler_sftp_send                Send files by SFTP

TASK QUEUE
Id     Enqueued   Start at
4594   13:07:27   2007-10-01 13:27:27 (-19:08min)
```

The "Show description" item in the menu opens a new browser window containing the job description. Job descriptions such as the one shown below can be written with the Job Editor. This editor combines documentation of the job, its configuration and a generate function, where the job can be newly generated after changes to its configuration have been made. It is so integrated in the JobScheduler installation that the JobScheduler automatically takes account of changes in a job configuration without having to be restarted.

The documentation for the Job Editor can be read in the Job Documentation Editor tutorial.



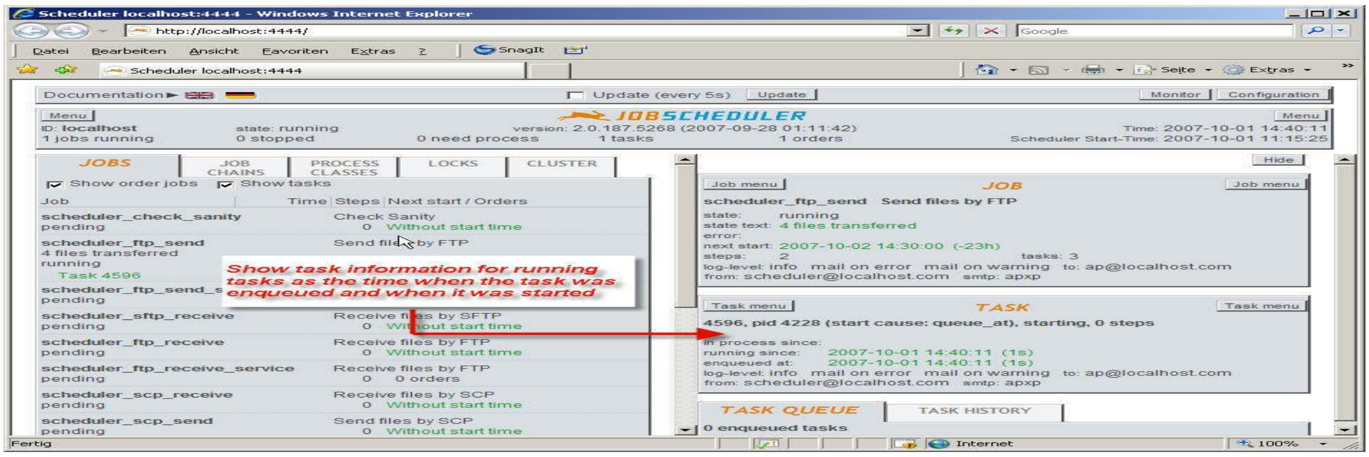
Job descriptions are available for standard jobs from the Job Scheduler distribution

Job	Name/ Title	JobSchedulerFTPSend Sends files by FTP
	Order Control	This job is triggered by orders or by standard job starts.
	Tasks	unbounded
	Script	• Language: java • Name of Java Class: sos.scheduler.ftp.JobSchedulerFTPSend • — Resource: sos.scheduler.jar

Configuration — This job can be used standalone or as an order driven job. Parameters can be accepted as job or as order parameters.

Job Parameters	
ftp_host	Default: --- Host or IP address to which files should be transferred.
ftp_port [optional]	Default: 21 Port by which files should be transferred.
ftp_user	Default: anonymous User name for authorization at the FTP server.
ftp_password [optional]	Default: --- Password for authorization at the FTP server.
ftp_account [optional]	Default: --- Account for authorization at the FTP server.
ftp_keep_connection [optional]	Default: false Tries to process several orders with the same connection. The connection parameters need to be the same.
ftp_transfer_mode	Default: binary Tranfer mode can be either ascii or binary .
ftp_passive_mode	Default: 0 Passive mode is often used with firewalls. Valid values are 0 or 1 .

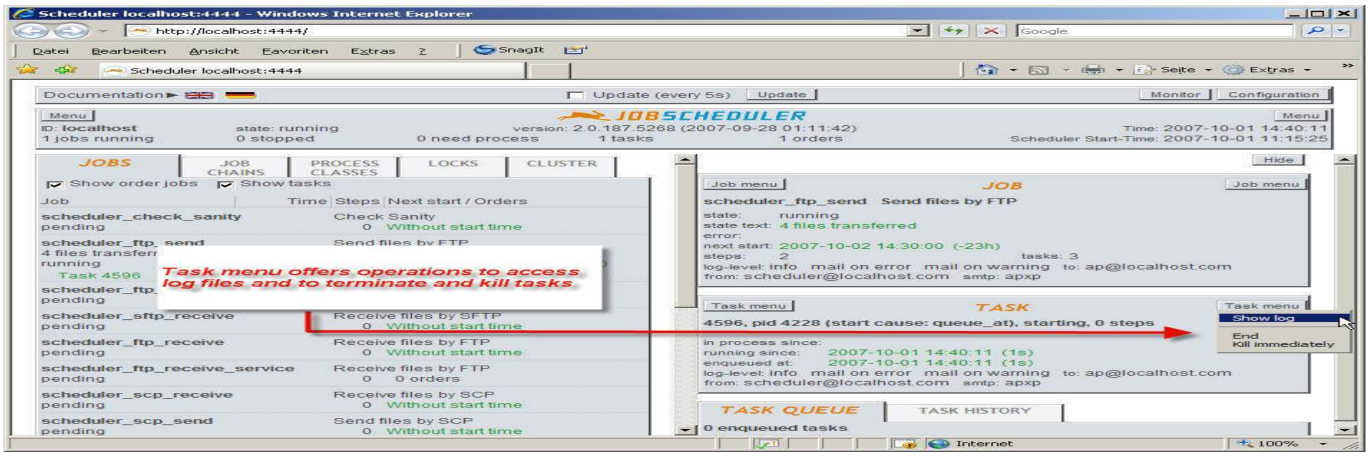
Information about tasks carrying out a job is shown in the task area of the interface as shown in the next screen shot:



Show task information for running tasks as the time when the task was enqueued and when it was started

```
JOB
scheduler_ftp_send   Send files by FTP
state:       running
state text:  4 files transferred
error:
next start:  2007-10-02 14:30:00 (-23h)
steps:       2                         tasks: 3
log-level: info  mail on error  mail on warning  to: ap@localhost.com
from: scheduler@localhost.com  smtp: apxp

TASK
4596, pid 4228 (start cause: queue_at), starting, 0 steps
in process since:
running since:   2007-10-01 14:40:11 (1s)
enqueued at:     2007-10-01 14:40:11 (1s)
log-level: info  mail on error  mail on warning  to: ap@localhost.com
from: scheduler@localhost.com  smtp: apxp

TASK QUEUE
0 enqueued tasks
```

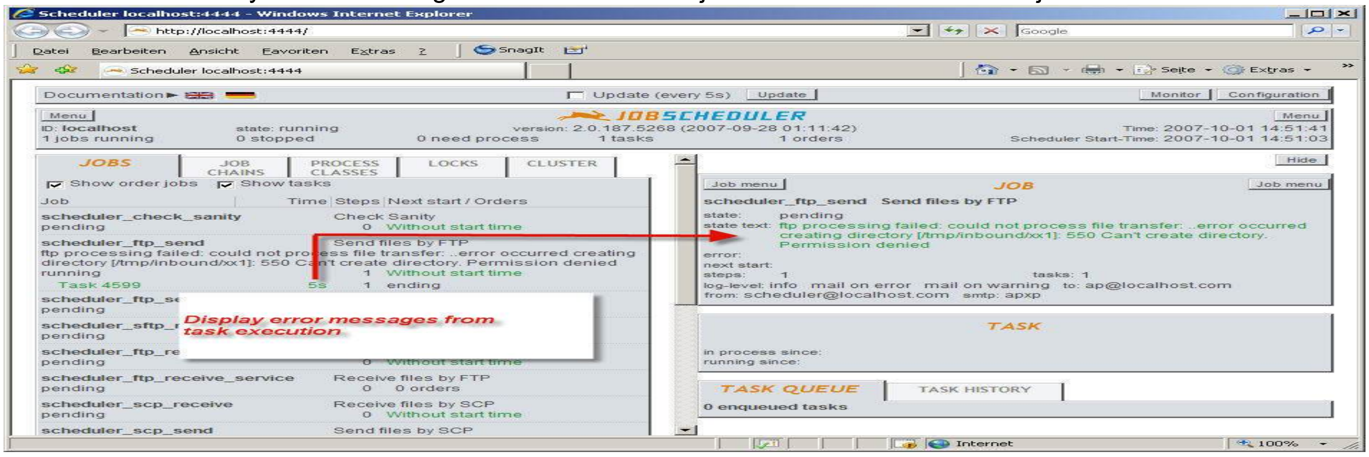
A task menu is provided as shown in the following screen shot:



```
ID: localhost          state: running                     version: 2.0.187.5268 (2007-09-28 01:11:42)          Time: 2007-10-01 14:40:11
1 jobs running         0 stopped        0 need process    1 tasks          1 orders          Scheduler Start-Time: 2007-10-01 11:15:25

Job                                Time | Steps | Next start / Orders
scheduler_check_sanity                     Check Sanity
pending                                      0   Without start time
scheduler_ftp_send                         Send files by FTP
4 files transferr
running
  Task 4596
scheduler_ftp_
pending
scheduler_sftp_receive                     Receive files by SFTP
pending                                      0   Without start time
scheduler_ftp_receive                      Receive files by FTP
pending                                      0   Without start time
scheduler_ftp_receive_service              Receive files by FTP
pending                                      0     0 orders
scheduler_scp_receive                      Receive files by SCP
pending                                      0   Without start time
scheduler_scp_send                         Send files by SCP
pending                                      0   Without start time

JOB
scheduler_ftp_send   Send files by FTP
state:       running
state text:  4 files transferred
error:
next start:  2007-10-02 14:30:00  (-23h)
steps:       2                                tasks: 3
log-level: info   mail on error   mail on warning   to: ap@localhost.com
from: scheduler@localhost.com   smtp: apxp

TASK
4596, pid 4228 (start cause: queue_at), starting, 0 steps
in process since:
running since:    2007-10-01 14:40:11  (1s)
enqueued at:      2007-10-01 14:40:11  (1s)
log-level: info   mail on error   mail on warning   to: ap@localhost.com
from: scheduler@localhost.com   smtp: apxp

0 enqueued tasks
```

Task menu offers operations to access log files and to terminate and kill tasks

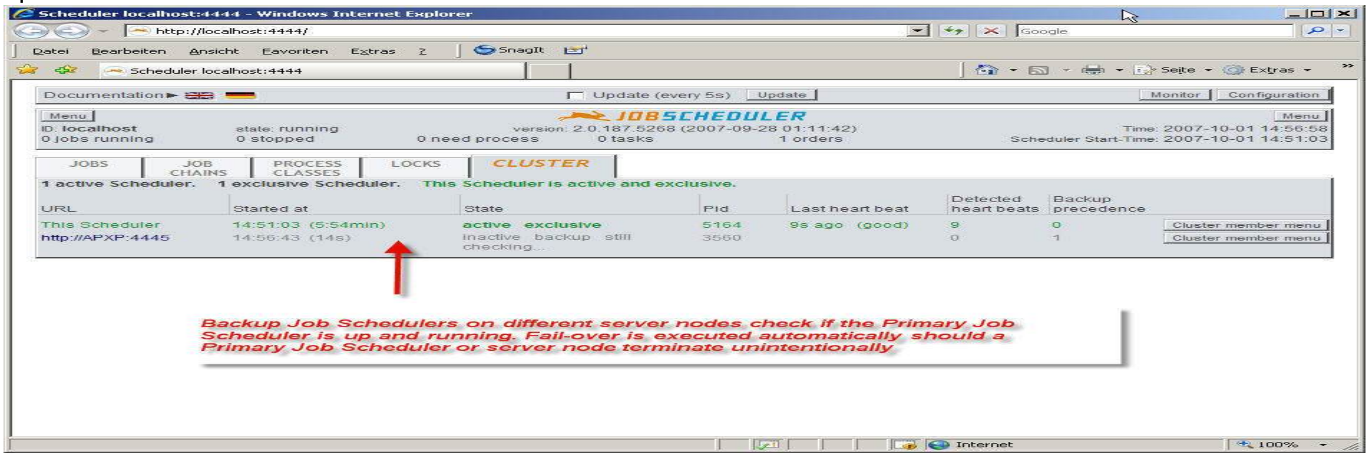
Information about any errors occuring in the execution of a job is shown in the interface job area:



```
ID: localhost          state: running                     version: 2.0.187.5268 (2007-09-28 01:11:42)          Time: 2007-10-01 14:51:41
1 jobs running         0 stopped        0 need process    1 tasks          1 orders          Scheduler Start-Time: 2007-10-01 14:51:03

Job                                Time | Steps | Next start / Orders
scheduler_check_sanity                     Check Sanity
pending                                      0   Without start time
scheduler_ftp_send                         Send files by FTP
ftp processing failed: could not process file transfer: ..error occurred creating
directory [/tmp/inbound/xx1]: 550 Can't create directory. Permission denied
running                                      1   Without start time
  Task 4599                          5s      1   ending
scheduler_ftp_se
pending
scheduler_sftp_r
pending
scheduler_ftp_re
pending                                      0   Without start time
scheduler_ftp_receive_service              Receive files by FTP
pending                                      0     0 orders
scheduler_scp_receive                      Receive files by SCP
pending                                      0   Without start time
scheduler_scp_send                         Send files by SCP

JOB
scheduler_ftp_send   Send files by FTP
state:       pending
state text:  ftp processing failed: could not process file transfer: ..error occurred
             creating directory [/tmp/inbound/xx1]: 550 Can't create directory.
             Permission denied
error:
next start:
steps:       1                                tasks: 1
log-level: info   mail on error   mail on warning   to: ap@localhost.com
from: scheduler@localhost.com   smtp: apxp

TASK
in process since:
running since:

0 enqueued tasks
```

Display error messages from task execution

If the JobScheduler is part of a JobScheduler cluster, providing backup, monitoriung and fail-over features, a corresponding "Cluster" menu tab will be shown in the main working area of the interface. As can be seen in the next screen shot, a button is provided to a menu for configuring cluster related aspects of the JobScheduler operation:



```
ID: localhost          state: running                     version: 2.0.187.5268 (2007-09-28 01:11:42)          Time: 2007-10-01 14:56:58
0 jobs running         0 stopped        0 need process    0 tasks          1 orders          Scheduler Start-Time: 2007-10-01 14:51:03

1 active Scheduler.    1 exclusive Scheduler.    This Scheduler is active and exclusive.
```

URL	Started at	State	Pid	Last heart beat	Detected heart beats	Backup precedence	
This Scheduler	14:51:03 (5:54min)	active exclusive	5164	9s ago (good)	9	0	Cluster member menu
http://APXP:4445	14:56:43 (14s)	inactive backup still checking...	3560		0	1	Cluster member menu

Backup Job Schedulers on different server nodes check if the Primary Job Scheduler is up and running. Fail-over is executed automatically should a Primary Job Scheduler or server node terminate unintentionally

The principles for using the JobScheduler GUI and which have been described in conjunction with the features shown in the preceeding screen shots are followed for the aspects of the interface which have not been shown in the screen shots. This means, for example, that use of the job chain part of the interface should follow intuitively.

Detailed information about the configuration and operation of the JobScheduler can be found in the following sections of this documentation:

- The XML Configuration (page 6) of the JobScheduler.
- Jobs and Job Chains (page 151).
- Orders (page 163).
- Log files and errors (page 174).
- Error messages (page 234).

5.2 XML Commands

These commands can be transmitted using TCP (e.g. telnet), HTTP and using the Command line options `-cmd` and `-send-cmd`.

Command	Answer (packed in `<spooler><answer>`)
`<add_jobs>`	`<ok>`
`<add_order>`	`<ok>` `<order>`
`<job.why>`	`<job>`
`<job_chain.modify>`	`<ok>`
`<job_chain_node.modify>`	`<ok>`
`<kill_task>`	`<ok>`
`<licence.use>`	`<ok>`
`<lock>`	`<ok>`
`<lock.remove>`	`<ok>`
`<modify_hot_folder>`	`<ok>`
`<modify_job>`	`<ok>`
`<modify_order>`	`<ok>`
`<modify_spooler>`	`<ok>`
`<param>`	`<ok>`
`<param.get>`	`<param>`
`<params>`	`<ok>`
`<params.get>`	`<params>`
`<process_class>`	`<ok>`
`<process_class.remove>`	`<ok>`
`<remove_job_chain>`	`<ok>`
`<remove_order>`	`<ok>`
`<schedule.remove>`	`<ok>`
`<scheduler_log.log_categories.reset>`	`<ok>`
`<scheduler_log.log_categories.set>`	`<ok>`
`<scheduler_log.log_categories.show>`	`<log_categories>`
`<show_calendar>`	`<calendar>`
`<show_history>`	`<history>`

Command	Answer (packed in `<spooler><answer>`)
`<show_job>`	`<job>`
`<show_job_chain>`	`<job_chain>`
`<show_job_chains>`	`<job_chains>`
`<show_jobs>`	`<jobs>`
`<show_order>`	`<order>`
`<show_state>`	`<state>`
`<show_task>`	`<task>`
`<start_job>`	`<ok>` `<task>`
`<terminate>`	`<ok>`

XML Elements

XML Element `<add_jobs>`

```
<add_jobs >
    job
</add_jobs>
```

Jobs are added with `temporary="yes"` and then started. Every job is deleted as soon as it ends.

Note that job names must be unique.

Parent Elements

<commands> -

XML Element `<add_order>`

```
<add_order
    job_chain                        = "name"
    id                               = "id"
    replace                          = "yes|no"
    priority                         = "number"
    title                            = "text"
    state                            = "text"
    web_service                      = "name"
    at                               = "timestamp"   Order Starting Time
    end_state                        = "text"        State before which the order should be successfully
                                                     completed and should leave the job chain
>
    params                       Parameters
    run_time
    xml_payload                  XML Payload
</add_order>
```

Adds a new order.

When the `<params>` element is specified, then the JobScheduler creates a `Variable_set` and makes it available in `Order.payload()`.

Example:

```
    <add_order job_chain="job_chain" id="1234" title="My First Order" state="100
at="now+3:00">

    <params>
        <param name="a_parameter" value="a value"/>

    </params>
</add_order>
```

Parent Elements

<commands> -

Attributes

`job_chain`=*"name"*

The name of the job chain in which the order is being processed.

`id`=*"id"*

The alphanumerical identification of the order. (*Note that this parameter may not be set to* `id` *- which is an XML reserved word.*)

`replace`=*"yes|no"* (Initial value:yes)

`replace="no"`: `Job_chain.add_order()` will be called.
`replace="yes"`: `Job_chain.add_or_replace_order()` will be called.

`priority`=*"number"*

If two orders should be started at the same time then orders with a higher priority are processed first.

`title`=*"text"*

The title of the order.

`state`=*"text"*

`web_service`=*"name"*

When an order has been completed and the end of the job chain reached, it is then transformed with a style sheet and forwarded to a Web Service.

See `<web_service>` (page 83).

`at`=*"timestamp"* (Initial value:now) Order Starting Time

`"now"`, `"yyyy-mm-dd HH:MM[:SS]"`, `"now + HH:MM[:SS]"` and `"now + SECONDS"` are possible.

See also `Order.at`.

`end_state`=*"text"* State before which the order should be successfully completed and should leave the job chain

See `Order.end_state`.

XML Element `<check_folders>`

```
<check_folders > </check_folders>
```

Checks the configuration directories for changes and updates the respective objects in the JobScheduler. On Win dows operating systems changes to a directory are noticed immediately, on Unix systems this could take up to 60 seconds.

The command is available in the built-in HTML interface and is used. for example, when developing jobs for Unix in order to shorten the waiting time until changes to the configuration are noticed by the JobScheduler.

Example:

```
<check_folders/>
```

Parent Elements

<commands> -

XML Element `<job.why>`

```
<job.why
    job                              = "job_name"
> </job.why>
```

This command determines possible reasons for a job not starting. As these reasons may not only lie in the job itself but also in the objects that are related to the job (job chains, orders, etc.), this command generally returns an XML element structure, referenced to the job. Common to all elements returned is that they have to end in one or more `obstacle` elements. Each of these obstacle elements will specifa a reason why the job has not started.

Example:

```
<job.why job="mein_job" />
```

Parent Elements

<commands> -

Attributes

`job`=*"job_name"*

The Job Name

XML Element `<job_chain.modify>`

```
<job_chain.modify
    job_chain                     = "name"                 Specifies a job chain
    state                         = "state"
> </job_chain.modify>
```

Example:

```
<job_chain.modify job_chain="my_job_chain" state="stopped"/>
```

Parent Elements

<commands> -

Attributes

`job_chain`=*"name"* Specifies a job chain

`state`=*"state"*

Not possible with distributed job chains.

Possible values are:

Individual job nodes can be stopped with `<job_chain_node.modify action="stop">`.

Messages

[ERROR]	SCHEDULER-384	job_chain is distributed and therefore does not support operation ''
[ERROR]	SCHEDULER-405	Setting state='' is not possible while job chain has state ''

XML Element `<job_chain_node.modify>`

```
<job_chain_node.modify
    job_chain                     = "name"              Specifies the job chain
    state                         = "state"             Specifies the job chain node
    action                        = "action"
> </job_chain_node.modify>
```

Example:

```
<job_chain_node.modify job_chain="my_job_chain" state="100" action="stop"/>
```

Parent Elements

<commands> -

Attributes

`job_chain`=*"name"* Specifies the job chain `state`=*"state"* Specifies the job chain node

`action`=*"action"*

Not allowed for distributed job chains.

Possible values are `action="process"`, `action="stop"` and `action="next_state"`, as described for `Job_chain_node.action`.

A complete job chain can be stopped using `<job_chain.modify state="stopped">`.

Messages

[info]	SCHEDULER-859	Due to action='next_state' the state '' has been skipped. Next state is ''

XML Element `<kill_task>`

```
<kill_task
    timeout                         = "duration_or_never"
    job                             = "job_name"
    id                              = "number"
    immediately                     = "yes|no"
> </kill_task>
```

Stops non-API tasks on Unix systems (`<script language="shell">`) together with any sub-processes. To do this, the JobScheduler determines all sub-processes using the operating system "ps -ef" command and stops them using `SIGKILL`. In addition, the process group task processes will be stopped.

Example:

```
<kill_task job="my_job" id="4711"/>
```

Example:

```
<kill_task job="my_job" id="4711" timeout="never"/>
```

Parent Elements

<commands> -

Attributes

`timeout`=*"duration_or_never"*

The period of time allowed for a process to end after receipt of a SIGTERM signal. If the process does not end within this time it will be killed with a SIGKILL signal.

Implementation on a JobScheduler Master

See How to Terminate Tasks for more information.

`job`=*"job_name"*

The job name.

`id`=*"number"*

The task Id. (*The attribute may not be named* `id` *- this term is a reserved word in XML.*)

`immediately`=*"yes|no"* (Initial value:no)

XML Element `<licence.use>`

```
<licence.use
    key                              = "licence_key"              Lizenzschlüssel
> </licence.use>
```

Adds a license key.

Parent Elements

<commands> -

Attributes

`key`=*"licence_key"* Lizenzschlüssel

The licence key is added to the licence keyring.

See also `sos.ini` (section `[licence]`, entry `key1= …`) (page 105).

XML Element `<lock>`

```
<lock
    name                             = "name"                   The lock name
    max_non_exclusive                = "integer"                Restricting Non-Exclusive Use
> </lock>
```

A lock can stop two tasks from running at the same time.

See `<lock.use>` (page 54) for information about the use of locks.

See also `<lock.remove>` (page 192)

Example:

```
<locks>
    <lock name="switching_database"/>
    <lock name="only_three_tasks" max_non_exclusive="3"/>
</locks>
```

Behavior with `<base>`

Supplements the `<lock>` element in the corresponding node of the basic XML configuration . Attributes specified here have precedence over those entered in the basic XML configuration.

Parent Elements

<locks> -

Attributes

`name`=*"name"* The lock name

`max_non_exclusive`=*"integer"* Restricting Non-Exclusive Use

The default setting is unlimited - which means that with `<lock.use exclusive="no">` an unlimited number of non-exclusive tasks can be started (but only one exclusive).

Messages

[ERROR]	SCHEDULER-887	More lock holders than new max_non_exclusive=: holders

XML Element `<lock.remove>`

```
<lock.remove
    lock                    = "path"                The lock name
> </lock.remove>
```

Removes a Lock.

See also `<lock>` (page 54).

Parent Elements

<commands> -

Attributes

`lock`=*"path"* The lock name

The lock can only be removed when it is not active - i.e. it is neither allocated a task nor being used by a job (`<lock.use>`).

See also `Lock.remove()`.

XML Element `<modify_hot_folder>`

```
<modify_hot_folder
    folder                  = "path"
>
    job
    job_chain
    lock                    Lock Declaration
```

```
   order
   process_class            Process class
   schedule
</modify_hot_folder>
```

This command adds a child element to a configuration directory. The file name is formed according to the rules for file-based configuration of jobs and tasks (see (page 88)).

Already existing files will be overwritten.

The child element (only one can be specified) must possess a `name=` attribute, or, if it is an order, `job_chain=` and `id=` attributes.

Parent Elements

<commands> -

Attributes

`folder`=*"path"* (Initial value:/)

Specifies the folder in which XML elements should be written. The JobScheduler creates this directory if it does not already exist (including all higher level directories, if necessary to the root directory).

XML Element `<modify_job>`

```
<modify_job
   job                           = "jobname"
   cmd                           = "cmd"
> </modify_job>
```

Example:

```
<modify_job job="my_job" cmd="wake"/>
```

Parent Elements

<commands> -

Attributes

`job`=*"jobname"*

The name of the job for which a command is intended.

`cmd`=*"cmd"*

The following sub-commands may be used:

XML Element `<modify_order>`

```
<modify_order
    job_chain                   = "state"
    order                       = "id"
    state                       = "state"
    title                       = "title"
    action                      = "action"
    setback                     = "no"
    priority                    = "number"
    suspended                   = "yes|no"
    at                          = "timestamp"
    end_state                   = "text"        State before which the order should be successfully
                                                completed and should leave the job chain
>
    params                  Parameters
    xml_payload             XML Payload
    run_time
</modify_order>
```

Example:

```
<modify_order job_chain="my_job_chain" order="42" priority="1"/>
```

Parent Elements

<commands> -

Attributes

`job_chain`=*"state"*

The order job chain.

`order`=*"id"*

The order identifier (alphanumerical).

`state`=*"state"*

Changes the state of an order and thereby its position in a job chain. Use of this attribute causes any setbacks (`Order.setback()`) made to be cancelled.

See also `Order.state`

`title`=*"title"*

Changes the order title

See also `Order.title`

`action`=*"action"*

`action="reset"` Resets the order: The order is returned to its original state, It is no longer suspended and setback is cancelled. The next start time for the order is calculated as if the order had successfully completed the job chain.

This operation is only possible, when the order is not being carried out by a job.

Messages

[ERROR]	SCHEDULER-217	order is being processed by task

`setback`=*"no"*

Is effective after an order has been setback using `Order.setback()`. This command ends all delays set using `<delay_order_after_setback>`, so that the order may be immediately executed. Note that the counter used to note how often an order has been setback remains unchanged.

See also `Order.setback()` and `<delay_order_after_setback>` (page 32).

`priority`=*"number"*

`suspended`=*"yes|no"*

Suspends or restarts the execution of an order - see `Order.suspended`.

`at`=*"timestamp"*

`"now"`, `"yyyy-mm-dd HH:MM[:SS]"`, `"now + HH:MM[:SS]"` and `"now + SECONDS"` can be used.

Changes the next start time of a waiting order,

- whose `Order.run_time` has not been reached, or
- which has been setback using `Order.setback()`.

`at="now"` restarts an order which has been waiting because of `Order.run_time` or `Order.setback()`.

See also `<add_order at="…">` (page 13) and `Order.at`.

`end_state`=*"text"* State before which the order should be successfully completed and should leave the job chain

See `Order.end_state`.

XML Element `<modify_spooler>`

```
<modify_spooler
    cmd                         = "cmd"
    timeout                     = "int"
> </modify_spooler>
```

The following sub-commands are not carried out after the JobScheduler has halted because of insufficient disc space for a log file (`<state waiting_errno="…">`): `pause`, `reload`, `terminate`, `terminate_and_restart`, `let_run_terminate_and_restart`. On the other hand `continue`, `abort_immediately` and `abort_immediately_and_restart` are immediately effective.

Example:

```
<modify_spooler cmd="abort_immediately_and_restart"/>
```

Parent Elements

<commands> -

Attributes

`cmd`=*"cmd"*

The following subcommands are allowed:

`timeout`=*"int"*

For `cmd="terminate"` and `cmd="terminate_and_restart"`: the time the JobScheduler will wait before stoping unterminated processes.

The default setting is an unlimited waiting time.

See `Spooler.terminate()` .

XML Element `<param>`

```
<param
    name                              = ""                    Unique Names
    value                             = ""
> </param>
```

See `<params>` (page 66).

Defines the individual parameters for the JobScheduler, a Job or an Order. In general all parameters are available with the API calls `Spooler.variables()` , `Task.params()` respectively `Order.payload()` or in shell Jobs as environment variable (with leading **SCHEDULER_PARAM_**)

Correspondent parameter in the JobScheduler configuration, at the Job and at the order are valid in the following sequence:
- order
- job
- scheduler

The use of individual parameters beginning with **scheduler.** is not recommended as this name space is reserved for the JobScheduler configuration settings.

The parameters can be overwritten and extended during run time.

See also `Variable_set` class.

JobScheduler Parameters

The following parameters can be used to configure the JobScheduler:

scheduler.variable_name_prefix	SCHEDULER_	Prefix of the JobScheduler einvironment variables (for compatibility reasons to older versions **SCHEDULER_VARIABLE_NAME_PREFIX** is allowed).
scheduler.max_kbyte_of_db_log_entry	unlimitet	Maximum size of the order logs in the database.
scheduler.order.keep_order_content_on_reschedule	TRUE	Preserves the order parameters after the execution of the order.
scheduler.order.distributed.balanced	FALSE	TRUE: distributed order tasks are allocated equally to the JobSchedulers in a cluster.. FALSE: an additional JobScheduler of the cluster takes over the execution of a distributed order when the JobScheduler initially processing the order is working at maximum capacity - i.e. the allowed number of processes reaches a maximum value.
scheduler.agent.keep_alive	-	Prevents connections between a JobScheduler Master and Classic Agent from timing out. The value attribute sets the interval in seconds between keep-alive packets. Keep-alive packets will not be sent if the parameter is not set or if the value attribute is empty.

- **SCHEDULER_VARIABLE_NAME_PREFIX**
- **scheduler.max_kbyte_of_db_log_entry**
- **scheduler.order.keep_order_content_on_reschedule**
- **scheduler.order.distributed.balanced**
- **scheduler.agent.keep_alive**

The JobScheduler Master and Classic Agent can be configured to prevent connections from timing out by adding a `scheduler.agent.keep_alive` parameter to the `<params>` section of the Master's scheduler.xml file. This file is located in the `$SCHEDULER_DATA/config` folder, where `$SCHEDULER_DATA` is the directory used for JobScheduler's configuration and log files.

Example:

```
<params>
   <param name="scheduler.agent.keep_alive" value="300" />
</params>
```

- The value attribute sets the interval in seconds between keep-alive packets.

  A duration lower than 30s will be silently replaced by 30s.
- Keep-alive packets will not be sent if the parameter is not set or if the value attribute is empty.
- The keep-alive parameter will be forwarded to the Agent along with other task configuration parameters for use when the Agent initiates a connection.
- Keep-alive packets will be sent across the network by the JobScheduler (either Master or Agent) that initiates a task.

JobScheduler Master

- The Master sends keep-alive packets to Classic Agents (up to and including Classic Agent release 1.9) via TCP connections.
- The Master log will show a `SCHEDULER-711` message at the info level stating that a keep-alive packet has been successfully sent.

Connection Keep-Alive for Master and Agent

- See the Connection Keep-Alive for Master and Agent article for more detailed information.

Delimitation

- Keep-alive packets are not created if Remote File Watching is performed by the Agent.
- Keep-alive packets are only sent for running jobs.

Important

- The JobScheduler Universal Agent (available with JobScheduler 1.10 and later) does not use keep-alive packets. Instead, the Universal Agent sends so-called heartbeats using a secure HTTP connection. See the `<remote_scheduler>` (page 71) element for information about the configuration of heartbeats.

Job Parameters

Job parameters can be called using the `Task.params()` method.

Order Parameters

Order parameters can be called using the `Order.params()` method.

Behavior with `<base>`

Supplements the `<param>` element in the corresponding node of the basic XML configuration with the attribute `name=` . Attributes specified here have precedence over those entered in the basic XML configuration.

Parent Elements

<params> - Parameters

Attributes

`name`="" Unique Names

`value`=""

Environment variables (e.g. `$HOME`) are replaced by this attribute (see Settings which Allow Environment Variables to be Called (page 106)).

XML Element `<param.get>`

```
<param.get
    name                         = "name"              The parameter name
> </param.get>
```

Returns `<param>` with the JobScheduler parameter or `<ok>` , when the parameter is not known.

Attributes

`name`=*"name"* The parameter name

XML Element `<params>`

```
<params >
    param                       Individual Parameters
    copy_params
    include
</params>
```

Specifies the parameters for the JobScheduler, a job or an order. The parameters can be overwritten and extended whilst the JobScheduler is running.

JobScheduler parameters can be called up using the `Spooler.variables()` method.

Job parameters are called using the `Task.params()` method.

The parameters for an order can be called using the `Order.payload()` method.

See also the `Variable_set` and `<sos.spooler.variable_set>` (page 79) classes.

Behavior with `<base>`

Supplements the `<params>` element in the corresponding node of the basic XML configuration . Attributes specified here have precedence over those entered in the basic XML configuration.

Parent Elements

<job> -

<add_order> - Add an order

<config> -

<modify_order> -

<payload> - Payload

<queued_task> -

<web_service> -

XML Element `<params>`

```
<params >
    param                       Individual Parameters
    copy_params
    include
</params>
```

Specifies the parameters for the JobScheduler, a job or an order. The parameters can be overwritten and extended whilst the JobScheduler is running.

JobScheduler parameters can be called up using the `Spooler.variables()` method.

Job parameters are called using the `Task.params()` method.

The parameters for an order can be called using the `Order.payload()` method.

See also the `Variable_set` and `<sos.spooler.variable_set>` (page 79) classes.

Behavior with `<base>`

Supplements the `<params>` element in the corresponding node of the basic XML configuration . Attributes specified here have precedence over those entered in the basic XML configuration.

Parent Elements

<job> -

<add_order> - Add an order

<config> -

<modify_order> -

<payload> - Payload

<queued_task> -

<web_service> -

XML Element `<params.get>`

```
<params.get
    name                            = "name"                  The parameter name
> </params.get>
```

Returns `<params>` including all JobScheduler variables.

Attributes

`name`=*"name"* The parameter name

XML Element `<payload>`

```
<payload >
    params                     Parameters
</payload>
```

If the payload is not a `Variable_set` object, then it will be added as text.

Parent Elements

<order> -

XML Element `<process_class>`

```
<process_class
    spooler_id                      = "scheduler_id"
    name                            = "name"
    max_processes                   = "number"
    remote_scheduler            = "host:port"           Task execution on remote
                                                        computers
    replace                     = "yes|no"
>
    remote_schedulers
</process_class>
```

Defines or modifies a process class.

See also `<process_class.remove>` (page 202).

Behavior with `<base>`

Supplements the `<process_class>` element in the corresponding node of the basic XML configuration . Attributes specified here have precedence over those entered in the basic XML configuration.

Parent Elements

<process_classes> -

Attributes

`spooler_id`=*"scheduler_id"*

An element having this attribute is only active when the attribute is either:

- empty
- set to the `-id=` JobScheduler start parameter
- or when the JobScheduler `-id` option is not specified when starting the JobScheduler.

`name`=*"name"*

The name of the process class. Should this attribute be missing or empty (`""`) then the default process class will be changed.

See the `process_class=` attribute of the `<job>` (page 42) element.

See the `process_class=` attribute of the `<job_chain>` (page 47) element.

`max_processes`=*"number"* (Initial value:30)

Limits the number of processes.

Some operating systems limit the number of processes which the JobScheduler can start. The number of processes configured here should not exceed the number allowed by the operating system. A value below 64 is usually safe.

For Microsoft Windows systems, the maximum number of processes that are allowed to be executed in parallel is currently 30.

`remote_scheduler`=*"host:port"* Task execution on remote computers

Specifies the remote computer on which the tasks of this process class are to be executed. This computer is specified using its host name or IP number and TCP port (see `<config tcp_port="…">` (page 21)).

The remote computer must allow access with `<allowed_host level="all">`.

Tasks executed communicate with the controlling JobScheduler via the API. However, the following points should be noted:

- `<include>` within `<script>` is executed by a task process. The file to be included is thereby read by the computer which carries out the task.
- The `Subprocess.timeout` and `Task.add_pid()` methods do not work. The JobScheduler cannot terminate remote subprocesses whose time limits have been exceeded.
- `Log.log_file()` is, as with almost all methods, carried out on the computer on which the JobScheduler is running and thereby accesses the files of its local file system.

Some settings are taken from the remote instead of from the controlling JobScheduler:

- `sos.ini` (section `[java]`, entry `javac= …`)
- `factory.ini` (section `[spooler]`, entry `tmp= …`)
- `<config java_options="…">`
- `<config java_class_path="…">`
- `<config include_path="…">`

Messages

`[warn]`	`SCHEDULER-849`	Timeout is not possible for a subprocess running on a remote host (it cannot be killed), pid=
`[warn]`	`SCHEDULER-850`	After lost connection to remote scheduler, process is going to be killed
`[info]`	`SCHEDULER-848`	Task pid= started for remote scheduler

`replace`=*"yes|no"* (Initial value:yes)

`replace="yes"` replaces the existing process class.

`replace="no"` only changes the attributes which are set by the process class.

XML Element `<process_class.remove>`

```
<process_class.remove
    process_class                 = "path"                    Name of the lock
> </process_class.remove>
```

Removes a process class.

See also `<process_class>` (page 68).

Parent Elements

<commands> -

Attributes

`process_class`=*"path"* Name of the lock

The JobScheduler delays deletion of a process class as long as tasks are still running in it. No new tasks will be started during this time.

See also `Process_class.remove()`.

XML Element `<remove_job_chain>`

```
<remove_job_chain
    job_chain                     = "name"
> </remove_job_chain>
```

REmoves a Job Chain.

This command uses the `Job_chain.remove()` method.

Example:

```
<remove_job_chain job_chain="my_job_chain" />
```

Parent Elements

<commands> -

Attributes

`job_chain`=*"name"*

XML Element `<remove_order>`

```
<remove_order
    job_chain                     = "name"
    order                         = "id"
> </remove_order>
```

Removes an order from the job chain. Note that a job which is in the process of carrying out an order will not be affected by this command.

The command uses the `Order.remove_from_job_chain()` method.

Example:

```
<remove_order job_chain="my_jobchain" order="42" />
```

Parent Elements

<commands> -

Attributes

`job_chain`=*"name"*

`order`=*"id"*

The (alphanumerical) order identifier.

XML Element `<schedule.remove>`

```
<schedule.remove
    schedule                     = "path"                    The schedule path
> </schedule.remove>
```

Removes a schedule.

See also `<schedule>` (page 75).

Parent Elements

<commands> -

Attributes

`schedule`=*"path"* The schedule path

All tasks for jobs that use the schedule will be stopped.

Orders that use the schedule complete their job chain and will not be restarted, as long as the schedule is not available.

XML Element `<scheduler_log.log_categories.reset>`

```
<scheduler_log.log_categories.reset
    delay                        = "integer"                 Delay in Seconds
> </scheduler_log.log_categories.reset>
```

Resets all log categories back to the values set as the JobScheduler started.

See also

Example:

```
<scheduler_log.log_categories.reset delay="360"/>
```

Parent Elements

<commands> -

Attributes

`delay`=*"integer"* Delay in Seconds

Delays resetting of the categories by the time specified.

XML Element `<scheduler_log.log_categories.set>`

```
<scheduler_log.log_categories.set
    category                     = "name"                    Category Name
    value                        = "0|1"
> </scheduler_log.log_categories.set>
```

See also

Example:

```
<scheduler_log.log_categories.set category="scheduler" value="1"/>
```

Parent Elements

<commands> -

Attributes

`category`=*"name"* Category Name

`value`=*"0|1"*

`1` activates the category, `0` deactivates it.

XML Element `<scheduler_log.log_categories.show>`

```
<scheduler_log.log_categories.show > </scheduler_log.log_categories.show>
```

See also

Example:

```
<scheduler_log.log_categories.show/>
```

Parent Elements

<commands> -

XML Element `<show_calendar>`

```
<show_calendar
    what                          = "what"
    limit                         = "number"
    from                          = "yyyy-mm-ddThh:mm:ss"
    before                        = "yyyy-mm-ddThh:mm:ss"
> </show_calendar>
```

This command returns the start times of jobs and orders. The results are not sorted.

The `limit=""` attribute is per defafult set to `limit="100"` and thereby limits the number of results. The limit should be set to a large enough value, so that no gaps appear in the (calender) list.

Example:

```
<show_calendar/>
<show_calendar limit="1000" what="orders" from="2007-04-03T00:00:00"
before="2007-05-01T00:00:00"/>
```

Parent Elements

<commands> -

Attributes

`what`=*"what"*

`what="orders"` also returns the start times of orders.

`limit`=*"number"* (Initial value:100)

Limits the number of entries that are returned in order avoid too large a result. Because calender entries are not sorted according to time but by object, this command does not return the next 100 entries but effectively 100 random entries.

The limit should be set high enough so that entries are not lost.

`from`=*"yyyy-mm-ddThh:mm:ss"*

Returns calender entries after a given date & time.

`before`=*"yyyy-mm-ddThh:mm:ss"*

Returns calender entries before a given date & time. The default setting is a week after `from=` or a week after `from`.

XML Element `<show_history>`

```
<show_history
   job                          = "job_name"
   id                           = "number"
   next                         = "number"
   prev                         = "number"
   what                         = ""
> </show_history>
```

Example:

```
<show_history id="4711" next="10"/>
```

Parent Elements

<commands> -

Attributes

`job`=*"job_name"*

The job name.

`id`=*"number"*

The identifier for the history. (*This attribute may not be called `id`, as this is a reserved term in XML.*)

`next`=*"number"*

The first *number* entries made after the task id should be returned.

`prev`=*"number"*

The last *number* entries made before the task id should be returned.

`what`=""

XML Element `<show_job>`

```
<show_job
    job                        = "name"
    job_chain                  = "name"
    what                       = ""
    max_orders                 = ""
    max_task_history           = ""
> </show_job>
```

Example:

```
<show_job job="my_job"/>
```

Parent Elements

<commands> -

Attributes

`job`=*"name"*

The job name.

`job_chain`=*"name"*

Neither orders which are in this job chain nor tasks which process such orders are returned.

`what`=""

`max_orders`=""

`max_task_history`=""

XML Element `<show_job_chain>`

```
<show_job_chain
    job_chain                    = ""                              The job chain name
    max_orders                   = ""
    max_order_history            = ""
    what                         = ""
> </show_job_chain>
```

Example:

```
<show_job_chains/>
```

Shows the job chain.

Parent Elements

<commands> -

Attributes

`job_chain`="" The job chain name

`max_orders`=""

`max_order_history`=""

`what`=""

XML Element `<show_job_chains>`

```
<show_job_chains
    what                       = ""
    max_orders                 = ""
    max_order_history          = ""
> </show_job_chains>
```

Example:

```
<show_job_chains/>
```

Invisible job chains (`<job_chain visible="no">`) will not be shown.

Parent Elements

<commands> -

Attributes

`what=""`

`max_orders=""`

`max_order_history=""`

XML Element `<show_jobs>`

```
<show_jobs
    what                      = ""
    max_orders                = ""
    max_task_history          = ""
> </show_jobs>
```

Example:

```
<show_jobs/>
```

Invisible jobs (`<job visible="no">`) will not be shown.

Parent Elements

<commands> -

Attributes

`what=""`

`max_orders=""`

`max_task_history=""`

XML Element `<show_order>`

```
<show_order
    job_chain                 = "name"
    order                     = "id"
    history_id                = "integer"
    what                      = ""
> </show_order>
```

Shows either the current order or an order out of the history.

Example:

```
    <show_order job_chain="my_job_chain" order="E4711"/>
```

Parent Elements

<commands> -

Attributes

`job_chain`=*"name"*

The name of the job chain.

`order`=*"id"*

The order identifier.

`history_id`=*"integer"*

Completed orders are saved in the JobScheduler database history. This can lead to there being more than one entry being made in the history under a particular order id, for example, when an order repeated because of the `<run_time>` setting. Each history entry has a history id which can be used to access the entry. The `order` attribute is then no longer required.

The history ID is returned by `<show_job_chain what="order_history">`.

`what`=*""*

XML Element `<show_state>`

```
<show_state
    what                          = "what"
    max_orders                    = "integer"
> </show_state>
```

The name of the `show_state` element may be shortened to `s`.

Example:

```
<show_state/>
<show_state what="job_chain_orders,job_orders"/>
```

Parent Elements

<commands> -

Attributes

`what`=*"what"*

(Note that this description applies to all commands; note however that some keywords will not make sense for all commands.)

The JobScheduler filters command replies so that they do not become unnecessarily large. The following key words may be used (if combined, then separated with a comma) to extend command replies.

`max_orders`=*"integer"*

Limits the number of orders that can be returned.

XML Element `<show_task>`

```
<show_task
   id                          = "number"
   what                        = ""
> </show_task>
```

Shows information for a current task or for a task out of the history.

Example:

```
<show_task id="4711"/>
```

Parent Elements

<commands> -

Attributes

`id`=*"number"*

The task id.

`what`=*""*

XML Element `<start_job>`

```
<start_job
   job                         = "job_name"
   name                        = "name"
   after                       = "number"
   at                          = "yyyy-mm-dd hh:mm:ss | now |
                                 period"
   force                       = "yes|no"
   web_service                 = "name"
>
   environment
   params                    Parameters
</start_job>
```

Example:

```
    <start_job job="my_job" at="now">

    <params>
        <param name="number" value="100"/>

    </params>
</start_job>
```

Parent Elements

<commands> -

Attributes

`job`=*"job_name"*

The job name.

`name`=*"name"*

A task can be given a name here.

`after`=*"number"*

A delay - the number of seconds after which a task should be started.

`at`=*"yyyy-mm-dd hh:mm:ss* | *now* | *period"* (Initial value:now)

The time at which a task is to be started. `<run_time>` is deactivated.

Relative times - `"now"`, `"now + HH:MM[:SS]"` and `"now + SECONDS"` - are allowed.

`at="period"` allows a job to start when allowed by `<run_time>` (that is in the current or next period).

`force`=*"yes|no"* (Initial value:yes)

`force="no"`:

- A job that with the "stopped" state will remain in this state.
- If a start time has been specified with either `<run_time>` or `<schedule>` and this does not allow a job to start then the JobScheduler will postpone the start to the next period.
- This means that `at="now"` has the same effect as `at="period"`.

`force="yes"`:

- This means that `at="now"` has the same effect as `at="period"`. A job that with the "stopped" state will be immediately started.
- A job with a start time specified using `at=` will be started at this time, regardless of whether or not a run-time period has been specified using `<run_time>` or `<schedule>`.

`web_service`=*"name"*

After a task has been executed, it is transformed with a style sheet and handed over to a Web Service.

See `<web_service>` (page 83).

XML Element `<terminate>`

```
<terminate
    all_schedulers               = "yes|no"
    restart                      = "yes|no"
    continue_exclusive_operat    = "yes|no"
    ion
    timeout                      = "seconds"
> </terminate>
```

Correct termination of the JobScheduler (see Terminating the JobScheduler (page 233)).

Example:

```
<terminate>
```

Parent Elements

<commands> -

Attributes

`all_schedulers`=*"yes|no"* (Initial value:no)

Causes all participating JobSchedulers to be stopped when used in conjunction with `-exclusive`

`restart`=*"yes|no"* (Initial value:no)

Causess the JobScheduler to restart.

`continue_exclusive_operation`=*"yes|no"* (Initial value:no)

Causes an inactive JobScheduler to take over operation when set in conjunction with `-exclusive`.

`timeout`=*"seconds"*

When specified, causes the JobScheduler to break off all tasks still running after the time specified.

XML Element `<xml_payload>`

```
<xml_payload > </xml_payload>
```

The `<xml_payload>` contains a single XML element (with unlimited child elements). The `Order.xml_payload` property contains this element as XML text.

Parent Elements

<order> -

<add_order> - Add an order

<modify_order> -

5.2.1 XML Answers

XML Elements

XML Element `<answer>`

```
<answer
    time                            = "yyyy-mm-dd hh:mm:ss.mmm"
>
    ERROR                       Error Messages
    history                     Task-History
    job                         Job
    job_chain                   Job Chain
    order                       Order
    process_classes             Process Classes
    ok                          Simple Answer
    log_categories              Log Categories
    state                       General Status of the JobScheduler
    task                        Tasks
</answer>
```

The JobScheduler only returns one of the child elements.

Parent Elements

<spooler> -

Attributes

`time`=*"yyyy-mm-dd hh:mm:ss.mmm"*

The answer time stamp.

XML Element `<ERROR>`

```
<ERROR
    time                  = "yyyy-mm-dd hh:mm:ss"   Timestamp
    class                 = "name"                  Name of the (C++) exception class
    code                  = "text"                  Error code
    text                  = "text"                  Error code and error text
    source                = "filename"              Name of the file containing the error
    line                  = "number"                Line Number
    col                   = "number"                Column Number
> </ERROR>
```

Parent Elements

<answer> - Answer

<job> - Job

<task> - Tasks

Attributes

`time`=*"yyyy-mm-dd hh:mm:ss"* Timestamp

`class`=*"name"* Name of the (C++) exception class

`code`=*"text"* Error code

`text`=*"text"* Error code and error text

`source`=*"filename"* Name of the file containing the error

the error file name, should it be possible to associate the error with a file.

`line`=*"number"* Line Number

the line number (starting from 1) in which the error occurs, should it be possible to associate the error with a particular line.

`col`=*"number"* Column Number

the number of the column in which the error occurs (starting from 1), should it be possible to associate the error with a particular column.

XML Element `<file_based>`

```
<file_based
   filename                   = ""                 File name
   last_write_time            = ""                 The file timestamp in UTC
   state                      = ""
>
   ERROR                 Error Messages
   removed               The incomplete removal of an object
</file_based>
```

Parent Elements

<process_class> -

<lock> -

<job> - Job

<job_chain> - Job Chain

<order> - Order

Attributes

`filename`="" File name

`last_write_time`="" The file timestamp in UTC

`state`=""

one of the following values: `undefined`, `not_initialized`, `initialized`, `loaded`, `active` and `closed`.

XML Element `<history>`

```
<history >
   history. entry              Entry in the Task History
</history>
```

The History is read from the database.

Parent Elements

<history> - Task-History

XML Element `<history. entry>`

```
<history. entry
   task                        = "number"                 Task id
   id                          = "number"                 Task id (out of date)
   spooler_id                  = "name"                   Scheduler id
   job_name                    = "name"                   Job name
   start_time                  = "yyyy-mm-dd hh:mm:ss"    The task start time
   end_time                    = "yyyy-mm-dd hh:mm:ss"    The task end time
   cause                       = "cause"                  The Reason for the start
   steps                       = "number"                 Number of job steps
   error                       = "0|1"                    "1" for a job error
   error_code                  = "text"                   Error code
   error_text                  = "text"                   Error code with error text
>
   ERROR                       Error Messages
</history. entry>
```

The History is read from the database.

> **Example:**

Parent Elements

<history> - Task-History

Attributes

`task`=*"number"* Task id

`id`=*"number"* Task id (out of date)

Use the attribute `task=`.

`spooler_id`=*"name"* Scheduler id

The JobScheduler id (or `"-"`, should the scheduler not have an id).

`job_name`=*"name"* Job name

`start_time`=*"yyyy-mm-dd hh:mm:ss"* The task start time

`end_time`=*"yyyy-mm-dd hh:mm:ss"* The task end time

`cause`=*"cause"* The Reason for the start

See `<task cause="…">` (page 230)

`steps`=*"number"* Number of job steps

`error`=*"0|1"* "1" for a job error

`error_code`=*"text"* Error code

The `<ERROR>` element will be returned instead of this attribute within `<job>`.

`error_text`=*"text"* Error code with error text

The `<ERROR>` element will be returned instead of this attribute within `<job>`.

XML Element `<job>`

`<job`

`job`	= *"name"*	Job name
`state`	= *"name"*	State
`waiting_for_process`	= *"yes|no"*	When a job is waiting for a process
`all_steps`	= *"number"*	The number of job steps for all tasks
`all_tasks`	= *"number"*	The number of tasks already started
`state_text`	= *"text"*	The properties of job.state_text
`log_file`	= *"dateiname"*	The name of the protocol file
`order`	= *"yes|no"*	For order controlled jobs
`tasks`	= *"number"*	Maximum number of tasks allowed
`next_start_time`	= *"yyyy-mm-dd hh:mm:ss.mmm"*	Next planned start time

```
    delay_after_error          = "yyyy-mm-dd hh:mm:ss.mmm"  delay_after_error takes effect
                                                            after an error
    in_period                  = "yes|no"                   When a <period> is valid
    has_description            = "yes"                      yes, when a job has a <
                                                            description>
    remove                     = "yes"                      yes, when a job is removed
    temporary                  = "yes"                      yes, when a job is temporary
    enabled                    = "yes|no"                   Disable a Job.
>
    tasks                  List of Tasks Currently Running
    description
    commands
    params
    lock.requestor
    queued_tasks           Number of Queued Tasks
    history                Task-History
    order_queue            Order Queue
    ERROR                  Error Messages
</job>
```

Example:

Parent Elements

<jobs> -

<answer> - Answer

Attributes

`job`=*"name"* Job name

`state`=*"name"* State

`state="none"`	The initial status of a job as the JobScheduler starts.
`state="stopping"`	The job has been stopped and not all tasks have ended.
`state="stopped"`	The job has been stopped and all tasks have ended.
`state="read_error"`	The status when `<script>` cannot be read.
`state="pending"`	No task is running.
`state="running"`	At least one task is running.

`waiting_for_process`=*"yes|no"* (Initial value:no) When a job is waiting for a process

This occurs when a job attempts to start a task but the process class is not large enough.

See `<job process_class="…">` (page 42).

`all_steps`=*"number"* The number of job steps for all tasks `all_tasks`=*"number"* The number of tasks already started

`state_text`=*"text"* The properties of job.state_text

See `Job.state_text`.

`log_file`=*"dateiname"* The name of the protocol file

`order`=*"yes|no"* (Initial value:no) For order controlled jobs

See `<job order="…">` (page 42).

`tasks`=*"number"* Maximum number of tasks allowed

See `<job tasks="…">` (page 42).

`next_start_time`=*"yyyy-mm-dd hh:mm:ss.mmm"* Next planned start time

`delay_after_error`=*"yyyy-mm-dd hh:mm:ss.mmm"* delay_after_error takes effect after an error

`in_period`=*"yes|no"* When a <period> is valid

`has_description`=*"yes"* yes, when a job has a <description>

`remove`=*"yes"* yes, when a job is removed

See `<modify_job cmd="remove">` (page 193).

`temporary`=*"yes"* yes, when a job is temporary

See `<job temporary="yes">` (page 42).

`enabled`=*"yes|no"* (Initial value:yes) Disable a Job.

See `<job enabled="yes">` (page 42).

XML Element `<job_chain>`

```
<job_chain
    name                         = "name"                  Name of the job chain
    orders                       = "number"                Number of orders in the job chain
    state                        = "The job chain state"
>
    job_chain_node            Position in a Job Chain
</job_chain>
```

Parent Elements

<job_chains> -

Attributes

`name`=*"name"* Name of the job chain

`orders`=*"number"* Number of orders in the job chain

`state`=*"The job chain state"*

`state="under_construction"`	The initial state after `Spooler.create_job_chain()`.
`state="finished"`	After `Spooler.add_job_chain()`.
`state="removing"`	After `Job_chain.remove()`, and as long as orders are being executed.

XML Element `<job_chain_node>`

```
<job_chain_node
    state                          = "text"      Order state which has been assigned to this
                                                 position in the job chain
    next_state                     = "text"      Next state for this order after successful
                                                 processing
    error_state                    = "text"      Order state after an error
    orders                         = "number"    Number of orders at a given position in the job
                                                 chain
>
    job                        Job
</job_chain_node>
```

Parent Elements

<job_chain> - Job Chain

Attributes

`state`=*"text"* Order state which has been assigned to this position in the job chain

`next_state`=*"text"* Next state for this order after successful processing

`error_state`=*"text"* Order state after an error

`orders`=*"number"* Number of orders at a given position in the job chain

XML Element `<log>`

```
<log
   level                          = "log_level"       The minimum level at which output is
                                                      allowed
   highest_level                  = "log_level"       The highest level used
   last_error                     = "string"          The last output produced by the error
                                                      level
   last_warning                   = "string"          The last output produced by the
                                                      warning level
   mail_on_error                  = "boolean"
   mail_on_warning                = "boolean"
   mail_on_success                = "boolean"
   mail_on_process                = "integer"
   smtp                           = "name"
   mail_from                      = "email_adresse"
   mail_to                        = "email_adresse"
   mail_cc                        = "email_adresse"
   mail_bcc                       = "email_adresse"
   mail_subject                   = "email_subject"
> </log>
```

The content of `<log>` is the protocol (when requested with `what="log"`).

Parent Elements

<job> - Job

<task> - Tasks

Attributes

`level`=*"log_level"* The minimum level at which output is allowed

See

`highest_level`=*"log_level"* The highest level used

`last_error`=*"string"* The last output produced by the error level

`last_warning`=*"string"* The last output produced by the warning level

`mail_on_error`=*"boolean"*

See `factory.ini` (section `[spooler]`, entry `mail_on_error= …`) (page 102)

`mail_on_warning`=*"boolean"*

See `factory.ini` (section `[spooler]`, entry `mail_on_warning= …`) (page 102)

`mail_on_success`=*"boolean"*

See `factory.ini` (section `[spooler]`, entry `mail_on_success= ...`) (page 102)

`mail_on_process`=*"integer"*

See `factory.ini` (section `[spooler]`, entry `mail_on_process= ...`) (page 102)

`smtp`=*"name"*

See `factory.ini` (section `[spooler]`, entry `smtp= ...`) (page 104)

`mail_from`=*"email_adresse"*

See `factory.ini` (section `[spooler]`, entry `log_mail_from= ...`) (page 101)

`mail_to`=*"email_adresse"*

See `factory.ini` (section `[spooler]`, entry `log_mail_to= ...`) (page 101)

`mail_cc`=*"email_adresse"*

See `factory.ini` (section `[spooler]`, entry `log_mail_cc= ...`) (page 101)

`mail_bcc`=*"email_adresse"*

See `factory.ini` (section `[spooler]`, entry `log_mail_bcc= ...`) (page 101)

`mail_subject`=*"email_subject"*

See `factory.ini` (section `[spooler]`, entry `log_mail_subject= ...`) (page 101)

XML Element `<log_categories>`

```
<log_categories >
    log_category
</log_categories>
```

See also

Parent Elements

<answer> - Answer

XML Element `<ok>`

```
<ok > </ok>
```

An empty element, with which commands which do not return any other result are acknowledged.

Example:

Parent Elements

<answer> - Answer

XML Element `<order>`

```
<order
   order                    = "id"                        Order identifier
   title                    = "text"                      Title
   state                    = "text"                      State
   job_chain                = "name"                      Job chain name
   job                      = "name"                      Job
   task                     = "id"                        Current task identifier
   in_process_since         = "yyyy-mm-dd hh:mm:ss.mmm"   Task start Time
   state_text               = "text"                      State text
   priority                 = "number"                    Priority
   created                  = "yyyy-mm-dd hh:mm:ss.mmm"   Point in time when an order is
                                                          created
   log_file                 = "file name"                 Name of the protocol file
   setback                  = "hh:mm:ss.mmm"              Point in time when an order is to
                                                          be cancelled
   next_start_time          = "yyyy-mm-dd hh:mm:ss.mmm"   Next starting time
   web_service              = "name"                      Name of the assigned Web
                                                          Service
>
   log                  Protocol
   run_time
   payload
</order>
```

Example:

Parent Elements

<answer> - Answer

<order_queue> - Order Queue

Attributes

`order`=*"id"* Order identifier

See `Order.id`

`title`=*"text"* Title

See `Order.title`

`state`=*"text"* State

See `Order.state`

`job_chain`=*"name"* Job chain name

See `Job_chain.add_order()`

`job`=*"name"* Job

See `Order.job`

`task`=*"id"* Current task identifier

`in_process_since`=*"yyyy-mm-dd hh:mm:ss.mmm"* Task start Time

`state_text`=*"text"* State text

See `Order.state_text`.

`priority`=*"number"* Priority

See `Order.priority`.

`created`=*"yyyy-mm-dd hh:mm:ss.mmm"* Point in time when an order is created

`log_file`=*"file name"* Name of the protocol file

`setback`=*"hh:mm:ss.mmm"* Point in time when an order is to be cancelled

`next_start_time`=*"yyyy-mm-dd hh:mm:ss.mmm"* Next starting time

Should this attribute not be specified, then the order will be carried out as soon as possible.

`web_service`=*"name"* Name of the assigned Web Service

XML Element `<order_queue>`

```
<order_queue
    length                          = "number"          Number of orders in the order queue
    next_start_time                 = "date"            Starting time for the next order
>
    order                       Order
</order_queue>
```

Parent Elements

<job> - Job

Attributes

`length`=*"number"* Number of orders in the order queue

The number of `<order>` elements can be zero or limited by the value that is given with the `max_orders=` or the `what=""` attributes.

`next_start_time`=*"date"* Starting time for the next order

XML Element `<process_classes>`

```
<process_classes >
   process_class
</process_classes>
```

Example:

Parent Elements

<answer> - Answer

XML Element `<processes>`

```
<processes >
   process
</processes>
```

Example:

Parent Elements

<process_class> -

XML Element `<queued_task>`

```
<queued_task
   task                        = "number"                   Task Id
   enqueued                    = "yyyy-mm-dd hh:mm:ss.mmm"  Time of entry in the task queue
   start_at                    = "yyyy-mm-dd hh:mm:ss.mmm"  Task start time
   name                        = "text"                     Task Name
>
   params
</queued_task>
```

Parent Elements

<queued_task> - Tasks in the Task Queue

Attributes

`task`=*"number"* Task Id

`enqueued`=*"yyyy-mm-dd hh:mm:ss.mmm"* Time of entry in the task queue

`start_at`=*"yyyy-mm-dd hh:mm:ss.mmm"* Task start time

`name`=*"text"* Task Name

XML Element `<queued_tasks>`

```
<queued_tasks
   length                          = "number"          Number of tasks in the task queue
>
   queued_task             Tasks in the Task Queue
</queued_tasks>
```

Parent Elements

<job> - Job

Attributes

`length`=*"number"* Number of tasks in the task queue

XML Element `<removed>`

```
<removed >
   ERROR                   Error Messages
</removed>
```

Parent Elements

<file_based> - File Based Objects

XML Element `<replacement>`

```
<replacement >
   process_class
   lock
   job                     Job
   job_chain               Job Chain
   order                   Order
</replacement>
```

The child element has the same name as the parent element. In the case of the replacement of a job, the nesting appears as `<job>_<replacement>_<job>_`. The child element contains the new object, with which the JobScheduler is not able to replace the current object. The JobScheduler replaces the object to be replaced (i.e. the current object) as soon as it is "free".

Parent Elements

<process_class> -

<lock> -

<job> - Job

<job_chain> - Job Chain

<order> - Order

XML Element `<state>`

```
<state
    config_file                 = "file name"                   XML configuration
    spooler_id                  = "name"                        Scheduler id
    id                          = "name"                        Out of date
    spooler_running_since       = "yyyy-mm-dd hh:mm:ss"         The start time of the
                                                                JobScheduler
    wait_until                  = "yyyy-mm-dd hh:mm:ss|never"   The next action of the
                                                                JobScheduler
    state                       = "state"                       The state of the JobScheduler
    log_file                    = "file name"                   Name of the main protocol file
    version                     = "vv.vv.vv yyyy-mm-dd"
    pid                         = "number"                      The JobScheduler process id
    db                          = "hostware_file_name"          Database
    db_waiting                  = "yes|no"                      Renew the JobScheduler
                                                                database connection
    waiting_errno               = "number"                      Error on writing a protocol (full
                                                                disc)
    waiting_errno_text          = "text"                        The text for the waiting_errno
                                                                attribute
    waiting_errno_filename      = "file name"                   The name of the file for which
                                                                the error occured
    loop                        = "number"                      Internal value: number of server
                                                                cycles
    waits                       = "number"                      Internal value: the number of
                                                                waiting states
    time                        = "yyyy-mm-dd hh:mm:ss"         Out of date
    tcp_port                    = "integer"
    udp_port                    = "integer"
>
    locks
    jobs
    job_chains
    process_classes         Process Classes
    subprocesses            Sub-Processes (dependent processes)
    remote_schedulers
</state>
```

Parent Elements

<answer> - Answer

Attributes

`config_file`=*"file name"* XML configuration

The name of the XML configuration file.

`spooler_id`=*"name"* Scheduler id

The JobScheduler Identifier - see the `-id` option.

`id`=*"name"* Out of date

Out of Date. Use the `spooler_id=` attribute instead.

`spooler_running_since`=*"yyyy-mm-dd hh:mm:ss"* The start time of the JobScheduler

`wait_until`=*"yyyy-mm-dd hh:mm:ss|never"* The next action of the JobScheduler

`state`=*"state"* The state of the JobScheduler

The state of the JobScheduler can be changed using the `<modify_spooler cmd="…">` command.

`log_file`=*"file name"* Name of the main protocol file

`version`=*"vv.vv.vv yyyy-mm-dd"*

The JobScheduler version.

`pid`=*"number"* The JobScheduler process id

`db`=*"hostware_file_name"* Database

The hostware file name of the database.

`db_waiting`=*"yes|no"* (Initial value:no) Renew the JobScheduler database connection

The `db_waiting` element is set to `"yes"`, should the JobScheduler lose its connection with the database and be in the process of reconnecting itself. In this situation the JobScheduler does not execute any jobs.

`waiting_errno`=*"number"* Error on writing a protocol (full disc)

When this attribute is set, then the JobScheduler waits and only reacts to commands via TCP, UDP or HTTP.

The JobScheduler can be restarted using `<modify_spooler cmd="continue">`.

`waiting_errno_text`=*"text"* The text for the waiting_errno attribute

`waiting_errno_filename`=*"file name"* The name of the file for which the error occured

`loop`=*"number"* Internal value: number of server cycles

This value increases as the JobScheduler executes operations such as a job step or a method callup.

`waits`=*"number"* Internal value: the number of waiting states

This value increases by one each time the JobScheduler does not have to execute a task and enters a waiting state.

`time`=*"yyyy-mm-dd hh:mm:ss"* Out of date

Out of Date. Use `<answer time="…">` instead.

`tcp_port`=*"integer"*

`udp_port`=*"integer"*

XML Element `<subprocess>`

```
<subprocess
    pid              = "number"                   The operating system process id
    priority         = "priority"
    timeout_at       = "yyyy-mm-dd hh:mm:ss.mmm"  Process time limit
    killed           = "yes|no"                   Set when a task has just been broken
                                                  off (killed).
> </subprocess>
```

Example:

Parent Elements

<subprocesses> - Sub-Processes (dependent processes)

Attributes

`pid`=*"number"* The operating system process id

`priority`=*"priority"*

The process priority: on Unix systems -20 to 20, on Windows `idle`, `below_normal`, `normal`, `above_normal` or `high`.

`timeout_at`=*"yyyy-mm-dd hh:mm:ss.mmm"* Process time limit

Is set when a process `Task.add_pid()` is set a time limit.

`killed`=*"yes|no"* (Initial value:no) Set when a task has just been broken off (killed).

XML Element `<subprocesses>`

```
<subprocesses >
   subprocess                 Subprocesses (dependent processes)
</subprocesses>
```

Example:

Parent Elements

<state> - General Status of the JobScheduler

<task> - Tasks

XML Element `<task>`

```
<task
   task                       = "number"                    Task Id
   id                         = "number"                    (out of date)
   state                      = "state"                     State of a Task
   name                       = "name"
   running_since              = "yyyy-mm-td hh:mm:ss.mmm"
   enqueued                   = "yyyy-mm-td hh:mm:ss.mmm"
   start_at                   = "yyyy-mm-td hh:mm:ss.mmm"
   idle_since                 = "yyyy-mm-td hh:mm:ss.mmm"
   in_process_since           = "yyyy-mm-td hh:mm:ss.mmm"
   cause                      = "cause"
   steps                      = "number"
   log_file                   = "file name"
   calling                    = "text"
   pid                        = "number"
   delayed_after_error_task   = "zahl"
   web_service                = "name"                      Name of the assigned
                                                            Web Service
>
   subprocesses               Sub-Processes (dependent processes)
   log                        Protocol
   order                      Order
   ERROR                      Error Messages
</task>
```

Example:

Parent Elements

<tasks> - List of Tasks Currently Running

Attributes

`task`=*"number"* Task Id

`id`=*"number"* (out of date)

The task id.

("id" is a reserved term in XML and therefore `task` should be used instead.)

`state`=*"state"* State of a Task

The following values may be returned:

`name`=*"name"*

The name of the task (see `<start_job>` (page 80)).

`running_since`=*"yyyy-mm-td hh:mm:ss.mmm"*

The time when the task was loaded or started: the task start time.

`enqueued`=*"yyyy-mm-td hh:mm:ss.mmm"*

The time when the task entered the task queue.

`start_at`=*"yyyy-mm-td hh:mm:ss.mmm"*

The time at which the task should be started (see `<start_job at="…">` (page 80)).

`idle_since`=*"yyyy-mm-td hh:mm:ss.mmm"*

The time at which the task state changed to `waiting_for_order`.

`in_process_since`=*"yyyy-mm-td hh:mm:ss.mmm"*

The time when the `spooler_process()` started, that is the time at which the current active job step started.

`cause`=*"cause"*

The cause of a job start.

`steps`=*"number"*

The number of completed job steps or calls of the `Job.spooler_process()` method.

`log_file`=*"file name"*

The name of the protocol file. This attribute is only set so long as the task is running.

`calling`=*"text"*

The name of the job method which has just been called up.

`pid`=*"number"*

The identifier for the process which the task is carrying out.

`delayed_after_error_task`=*"zahl"*

Wenn diese Task eine Wiederholung einer mit Fehler abgebrochen Task ist (s. `Job.delay_after_error_`), dann gibt dieses Attribut die Kennung der fehlerhaften Task an.

`web_service`=*"name"* Name of the assigned Web Service

XML Element `<tasks>`

```
<tasks
    count                          = "number"
>
    task                       Tasks
</tasks>
```

Parent Elements

<job> - Job

Attributes

`count`=*"number"*

The number of tasks currently running.

XML Element `<waiting_jobs>`

```
<waiting_jobs
    job                            = "name"
>
    job                        Job
</waiting_jobs>
```

Parent Elements

<process_class> -

Attributes

`job`=*"name"*

5.3 Terminating the JobScheduler

5.3.1 Correct Stopping

The JobScheduler stops when all jobs have come to an end.

See `Spooler.terminate()` and `<modify_spooler cmd="terminate">` (page 195).

5.3.2 Correct Stopping with a Time Limit

A time limit may be specified within which the JobScheduler must stop. Should tasks still be running at the end of this time limit, because for example they remain too long in the `spooler_process()` then the JobScheduler terminates all processes still running.

However the JobScheduler first sends out an e-mail listing the tasks being terminated.

The JobScheduler then waits a maximum 30 seconds for the tasks to terminate (this usually takes place immediately) before stopping itself.

See `Spooler.terminate()` and `<modify_spooler cmd="terminate">` (page 195).

5.3.3 Termination

The JobScheduler breaks off all processes (including the foreign processes registered with add_pid()) immediately. (Unix: `kill -s KILL`, Windows: `TerminateProcess()`). The JobScheduler then shuts itself down.

Advantage: The termination is immediate.

Disadvantage: the software, in particular the jobs, do not have the possibility to react. This means, for example, that temporary files cannot be deleted.

Appendix A: Messages

A.1 Messages for the Scheduler Package

Code	Text
`SCHEDULER-104`	There is an error such as the start time set in the begin="*(1)*" attribute is later than the end time set in end="*(2)*" *- Example -* An error results from, for example `<period begin="23:00" end="01:00"/>` as the start and end times must lie within the same day, i.e. between 00:00 und 24:00. See `<period>` (page 67).
`SCHEDULER-105`	Unknown command: <*(1)*> See XML commands (page 185).
`SCHEDULER-106`	Unknown command for task: *(1)*
`SCHEDULER-107`	Missing script
`SCHEDULER-110`	Unknown task state: *(1)*
`SCHEDULER-113`	Unexpected XML tag: *(1)*
`SCHEDULER-115`	Missing XML configuration file, option -config=FILENAME.XML
`SCHEDULER-116`	XML configuration contains no <config spooler_id=*(1)*> element
`SCHEDULER-117`	Cannot use different scripting languages in the same scripting engine
`SCHEDULER-119`	There is no security level "*(1)*"
`SCHEDULER-120`	Script has reported an error: *(1)*
`SCHEDULER-121`	The security settings do not allow this operation (or JobScheduler is waiting for database)
`SCHEDULER-122`	Task terminated
`SCHEDULER-125`	task.wait_until_terminated() cannot be called in its own thread. Deadlock.
`SCHEDULER-130`	Job *(1)* has already been defined
`SCHEDULER-136`	No history
`SCHEDULER-139`	Operation is not possible when using a plain text file for history

Code	Text
SCHEDULER-140	Task logged error: *(1)*
SCHEDULER-141	History has been suppressed for job *(1)* (history=no)
SCHEDULER-143	Task cannot start because it has no start time and no valid <run_time>
SCHEDULER-145	<script> com_class= and language= cannot be combined
SCHEDULER-146	Missing script
SCHEDULER-147	*(1)* cannot accept orders because it has not been declared with <job order="yes">
SCHEDULER-148	'*(1)*' has been initialized already and cannot be modified
SCHEDULER-149	There is no job in job chain "*(1)*" for the state "*(2)*"
SCHEDULER-150	The state "*(1)*" has already assigned a job in job chain *(2)*
SCHEDULER-151	Job chain is not in use (missing add_job_chain() or job chain is to be removed)
SCHEDULER-152	Job "*(1)*" is not in job chain "*(2)*"
SCHEDULER-153	'*(1)*' is not in state '*(2)*'
SCHEDULER-155	A temporary job cannot accept orders
SCHEDULER-156	*(1)* is not in *(2)*
SCHEDULER-157	*(1)* is not in a job chain
SCHEDULER-159	Id and priority of an order placed in an order queue cannot be changed
SCHEDULER-160	There is already a *(1)* '*(2)*'
SCHEDULER-161	There is no *(1)* '*(2)*'
SCHEDULER-162	There is no order *(1)* in job chain "*(2)*"
SCHEDULER-163	*(1)* is not in an order queue
SCHEDULER-164	<show_... what="*(1)*">: invalid value for attribute what=
SCHEDULER-166	<script> java_class= and language= cannot be combined
SCHEDULER-167	A program source cannot be set with <script> com_class= or java_class=

Code	Text
SCHEDULER-168	com_class= and java_class= cannot be combined in <script>
SCHEDULER-172	This kind of script (scripting engine, Perl, COM or Java) is not possible here
SCHEDULER-173	Missing script
SCHEDULER-174	Method "*(1)*" is missing in class "*(2)*"
SCHEDULER-179	<process> task did not terminate. pid=*(1)*
SCHEDULER-182	Unexpected XML element <*(1)*>
SCHEDULER-183	Scheduler cannot start because spooler_init() in '*(1)*' returned false
SCHEDULER-184	Scheduler database cannot be accessed due to a database problem
SCHEDULER-186	*(1)* is already in job chain *(2)*
SCHEDULER-187	Order.setback(): An order can only be set back when it is being executed by a task
SCHEDULER-188	Order.setback() and Order.state= cannot be combined
SCHEDULER-191	separate_process: Unexpected state *(2)* in *(1)*
SCHEDULER-194	process_class and separate_process cannot be combined
SCHEDULER-196	use_engine="*(1)*" is deprecated
SCHEDULER-197	Module::java_method_id(*(1)*): unknown Java class
SCHEDULER-198	Missing variable name
SCHEDULER-199	Module instance has been closed
SCHEDULER-200	Remote instance has been closed
SCHEDULER-201	Missing working directory when calling Java compiler
SCHEDULER-202	Connection to task has been lost, state=*(1)*: *(2)*
SCHEDULER-203	Module instance in the server has been closed
SCHEDULER-204	Job "*(1)*" is in error state. *(2)*
SCHEDULER-205	No database given (entry db= in factory.ini)

Code	Text
SCHEDULER-207	Task *(1)* is unknown (and not noted in the history)
SCHEDULER-208	Only a date without a time is allowed
SCHEDULER-209	Too many objects for Microsoft Windows MsgWaitForMultipleObjects(). MAXIMUM_WAIT_OBJECTS-1=*(1)* has been reached. Operation is not executable
SCHEDULER-210	Too many processes. The number of processes is limited to *(1)*
SCHEDULER-212	Missing entry [spooler] html_dir= in file factory.ini
SCHEDULER-213	Error in HTTP header: expecting *(1)*
SCHEDULER-214	Forbidden path in HTTP request
SCHEDULER-215	Unknown or terminated task *(1)*
SCHEDULER-217	*(1)* is being processed by *(2)*
SCHEDULER-218	Subprocess pid=*(2)* has been terminated with exit code=*(1)*. Command=*(3)*
SCHEDULER-219	Subprocess pid=*(2)* has been terminated with signal *(1)*. Command=*(3)*
SCHEDULER-220	<run_time *(1)*> cannot be carried out for an order
SCHEDULER-221	<day day="*(1)*">: only the days from *(2)* to *(3)* are possible
SCHEDULER-222	This command <*(1)*> is only executable by way of a TCP connection
SCHEDULER-223	Supervisor has reported an error: *(1)*
SCHEDULER-224	Supervisor has closed the connection
SCHEDULER-225	start_new_file() is only applicable for the main protocol
SCHEDULER-226	Order has not been processed because the task has been terminated before the processing step (spooler_process())
SCHEDULER-227	Scheduler script has logged an error: *(1)*
SCHEDULER-230	Job "*(1)*" is to be removed - it is not longer usable
SCHEDULER-231	Missing attribute *(2)* in XML element <*(1)*>
SCHEDULER-232	Attribute <*(1) (2)*="*(3)*"> is not allowed here
SCHEDULER-234	The <job *(1)*="..."> attribute cannot be overwritten

Code	Text
SCHEDULER-235	Unknown web service: "*(1)*"
SCHEDULER-236	Web service "*(1)*" has already been declared
SCHEDULER-237	XSLT stylesheet "*(1)*" does not deliver a result
SCHEDULER-238	Web service or alias for url_path="*(1)*" has already been declared
SCHEDULER-239	XML root element <*(1)*> expected
SCHEDULER-240	Order has not been assigned to a web service
SCHEDULER-241	Task has not been assigned to a web service
SCHEDULER-242	forward_xslt_stylesheet *(1)* does not deliver <service_request>
SCHEDULER-243	Property *(1)* is fixed and is no longer modifiable
SCHEDULER-244	Element <content> is missing or empty
SCHEDULER-245	Element <content> has more than one child element
SCHEDULER-246	Order does not belong to a web service operation
SCHEDULER-247	HTTP response has already been sent `Web_service_response` cannot be changed after `Web_service_response.send()` .
SCHEDULER-248	web_service_operation has finished and is not longer valid
SCHEDULER-249	Order id cannot be represented as a character string: *(1)*
SCHEDULER-250	Order state cannot be represented as a character string: *(1)*
SCHEDULER-251	Order payload cannot be represented as character string: *(1)*
SCHEDULER-252	Invalid url path - it has to begin with '/': *(1)*
SCHEDULER-254	*(1)* tasks did not terminate in spite of kill *(2)* seconds ago. Extending deadline for *(3)*s
SCHEDULER-255	*(1)* tasks did not terminate in spite of kill *(2)* seconds ago. JobScheduler is aborting
SCHEDULER-256	Deadline for terminating JobScheduler has been reached, but *(1)* tasks are still running
SCHEDULER-257	Job is being removed now

Code	Text
SCHEDULER-258	Job will be removed after all its tasks have terminated
SCHEDULER-260	Error in JobScheduler Skript, spooler_exit(): *(1)* Error in JobScheduler script, spooler_exit(): *(1)*
SCHEDULER-261	Nothing done, event=*(1)*, operations=*(2)*
SCHEDULER-262	Kill signal (*(1)*) received. JobScheduler is terminating
SCHEDULER-263	Kill signal (*(1)*) #*(2)* received. *(3)*
SCHEDULER-264	SCHEDULER IS TERMINATING AFTER SERIOUS ERROR
SCHEDULER-266	Error when writing the history: *(1)*
SCHEDULER-267	Error when storing the log in table *(1)*: *(2)*
SCHEDULER-268	Log of task *(1)* is not readable from database: *(2)*
SCHEDULER-269	Error when closing the job history: *(1)*
SCHEDULER-271	Task is being terminated in favour of job *(1)*
SCHEDULER-272	Terminating task after reaching deadline <job timeout="*(1)*">
SCHEDULER-273	Killing subprocess *(1)*
SCHEDULER-274	Error when killing subprocess *(1)*: *(2)*
SCHEDULER-275	Deadline reached, killing subprocess *(1)*
SCHEDULER-276	Could not kill task
SCHEDULER-278	<period> ended, terminating task
SCHEDULER-279	Process terminated with signal *(1)* (*(2)*)
SCHEDULER-280	Process terminated with exit code *(1)* (0x*(2)*)
SCHEDULER-281	Killing process
SCHEDULER-283	Error when loading task *(1)* from database: *(2)*
SCHEDULER-284	Ignoring cmd='*(1)*' because job is to be removed

Code	Text
`SCHEDULER-285`	Max. *(1)* history records are being read
`SCHEDULER-287`	TCP connection not allowed
`SCHEDULER-289`	*(1)* is blocked. Trying a further *(2)* seconds before giving up
`SCHEDULER-290`	Timeout: No job started to execute *(1)*, cancelling order and HTTP operation See `<web_service timeout="…">` (page 83)
`SCHEDULER-291`	Error when removing protocol file: *(1)*
`SCHEDULER-292`	Missing from, to, subject or body. Suppressing email
`SCHEDULER-293`	Waiting for closing module instance ...
`SCHEDULER-295`	Error when loading order *(1)* from database: *(2)*
`SCHEDULER-297`	Order has been carried out without web_service_operation.send(), operation cancelled
`SCHEDULER-298`	Ignoring job=*(1)* because order is removed from job chain
`SCHEDULER-299`	Removing process *(1)* after error
`SCHEDULER-300`	Monitoring of directory *(1)* has ended after error: *(2)*
`SCHEDULER-301`	UDP message from *(1)* is not allowed
`SCHEDULER-302`	Error sending e-mail: *(1)*
`SCHEDULER-303`	Problem with database: *(1)*
`SCHEDULER-304`	Error when reading next id (column "*(2)*"): *(1)*
`SCHEDULER-305`	Error when inserting into table *(1)*: *(2)*
`SCHEDULER-306`	Error when updating table *(1)*: *(2)*
`SCHEDULER-307`	Not writing history record because of already existing database problem
`SCHEDULER-308`	Ignoring cancel() because response has already been sent
`SCHEDULER-309`	Error when opening database: *(1)*
`SCHEDULER-310`	Error when closing the database: *(1)*

Code	Text
SCHEDULER-311	Error when processing HTTP request *(1)*: *(2)*
SCHEDULER-312	Variable "*(1)*" (vt=*(2)*) cannot be converted to Text: *(3)*
SCHEDULER-313	Cannot store order in database: *(1)*
SCHEDULER-316	<web_service name="*(1)*"/>: attributes job_chain=/timeout= and request/response/forward_xslt_stylesheet= cannot be combined
SCHEDULER-318	Option -env=NAME=VALUE: missing '=' between name and value: -env="*(1)*"
SCHEDULER-319	XML-Element <*(1)*> is not possible here
SCHEDULER-320	Could not send mail
SCHEDULER-321	Mail has been sent, but an error occurred when processing the mail queue: *(1)*
SCHEDULER-322	min_tasks=*(1)* is greater than max_tasks=*(2)* See `<job min_tasks="…">` (page 42).
SCHEDULER-323	exit_code is available only when task module is loaded See `Task.exit_code`.
SCHEDULER-324	Invalid value for attribute exit_code="" in <commands> See `<job><commands>` (page 42).
SCHEDULER-325	Attribute exit_code is not applicable here See `<commands>` (page 20).
SCHEDULER-326	<commands on_exit_code="">: <commands> for exit code *(2)* is already defined See `<job><commands>` (page 42).
SCHEDULER-327	Last error occurred when executing command: *(1)* See `<job><commands>` (page 42).
SCHEDULER-328	Executing <commands on_exit_code="*(1)*">: See `<job><commands>` (page 42).
SCHEDULER-329	<copy_params from=""/>: requested parameters are not available See `<copy_params from="order">` (page 26).
SCHEDULER-330	*(1)* could not be initialized:*(2)*
SCHEDULER-331	Scheduler is terminating after error (see last error line)
SCHEDULER-332	Error in *(1)* while switching to state=*(2)*: *(3)*

Code	Text
SCHEDULER-333	"now+SECONDS" expected in "*(1)*"
SCHEDULER-334	Error when building mail: *(1)*
SCHEDULER-335	Only "yes", "no" and a number are allowed with *(1)*="*(2)*": *(3)*
SCHEDULER-336	Only "yes", "no" and a list of signal names are allowed with *(1)*="*(2)*": *(3)*
SCHEDULER-337	Signal *(1)* is unknown on this operating system and is ignored
SCHEDULER-338	Order.payload is not a Variable_set
SCHEDULER-339	File does not exist and can therefore neither be moved nor removed: *(1)*
SCHEDULER-340	File still exists. Order has been set on the blacklist
SCHEDULER-342	There is no job between order source and order sink in *(1)*
SCHEDULER-343	This is no file order: *(1)*
SCHEDULER-344	order.id has more than *(2)* characters: *(1)*
SCHEDULER-345	order.id has more than *(2)* characters, the limit of your database column *(3)* is: *(1)*
SCHEDULER-346	Due to previous error the path will be ignored: *(1)*
SCHEDULER-347	Path has been removed from the blacklist: *(1)*
SCHEDULER-348	Database doesn't deliver width of column *(1)*.*(2)*
SCHEDULER-349	Database column *(1)*.*(2)* could not be expanded
SCHEDULER-350	Width of database column *(1)*.*(2)* is unknown. We assume *(3)*
SCHEDULER-351	<modify_order_... setback="*(1)*">: Invalid value for attribute setback=
SCHEDULER-353	No immediate response from command <*(1)*>
SCHEDULER-354	Scheduler Java classes (sos.spooler.jar *(1)* expected) are not up to date: *(2)*
SCHEDULER-355	SQL update statement changed more than one record, *(1)*: *(2)*
SCHEDULER-357	This is a member of a cluster (option -exclusive or -distributed-orders), and therefore needs role 'scheduler'
SCHEDULER-359	For an exclusive or distributed Scheduler, the database *(1)* is not supported

Code	Text
SCHEDULER-360	Error when accessing database table *(1)*
SCHEDULER-361	No database
SCHEDULER-362	Scheduler is aborting because it has become inactive
SCHEDULER-363	Error when creating SQL table *(1)*: *(2)*
SCHEDULER-365	Illegal character in -id=*(1)*
SCHEDULER-367	Scheduler is aborting because it has lost its exclusivity
SCHEDULER-368	Option *(1)* needs *(2)*
SCHEDULER-369	Option *(1)* conflicts with *(2)*
SCHEDULER-370	Operation can only be performed on a distributed orders Scheduler
SCHEDULER-371	DATABASE INTEGRITY IS BROKEN
SCHEDULER-372	Exclusivity has been stolen by JobScheduler member '*(1)*'
SCHEDULER-373	UNEXPECTED DEACTIVATION BY JOBSCHEDULER MEMBER *(1)*
SCHEDULER-374	In *(1)*, state '*(2)*' has no job
SCHEDULER-375	Order is distributed and therefore does not support operation '*(1)*'
SCHEDULER-376	For a distributed order, this operation is not possible
SCHEDULER-377	After own late heart beat, JobScheduler member '*(1)*' has taken exclusiveness
SCHEDULER-378	After own late heart beat, this JobScheduler has been deactivated and the occupied orders have been freed by JobScheduler member '*(1)*'
SCHEDULER-379	*(1)* is occupied by JobScheduler member '*(2)*'
SCHEDULER-380	job_chain orders_recoverable="no" cannot be combined with distributed="yes", in *(1)*
SCHEDULER-381	Scheduler is not yet active and cannot execute the operation
SCHEDULER-384	*(1)* is distributed and therefore does not support operation '*(2)*'
SCHEDULER-385	Deletion of order in database has failed
SCHEDULER-386	Last heart beat was *(1)*, *(2)* seconds ago. Something is delaying JobScheduler execution, the JobScheduler is aborted immediately

Code	Text
SCHEDULER-389	This order stored in database is distributed and cannot be used in a non-distributed job chain
SCHEDULER-390	More than *(1)* nested includes in *(2)*
SCHEDULER-391	Invalid value '*(2)*' for setting '*(1)*'. Allowed values are: *(3)*
SCHEDULER-392	heart_beat_warn_timeout=*(1)* seconds has to be lower than heart_beat_own_timeout=*(2)*, which has to be lower than heart_beat_warn_timeout=*(3)* seconds
SCHEDULER-393	Error when calling <run_time> function '*(1)*': *(2)*
SCHEDULER-394	<run_time>-Function '*(1)*' returned start time *(2)* which is earlier than the requested beginning *(3)*
SCHEDULER-396	Job has not yet reached state '*(1)*' for operation *(2)*
SCHEDULER-397	A distributed order does not support *(1)*
SCHEDULER-398	After last error, next_start_function='*(1)*' will no longer be executed for this <run_time>
SCHEDULER-399	Error in *(1)*:*(2)*
SCHEDULER-401	Unknown '*(1)*'
SCHEDULER-403	There is no valid state for job chain node state='*(1)*', action='next_state' (circular next_state)
SCHEDULER-405	Setting state='*(1)*' is not possible while job chain has state '*(2)*'
SCHEDULER-406	Error in job chain node state='*(1)*': *(2)*
SCHEDULER-407	Attribute *(1)* is empty
SCHEDULER-408	<*(1) (2)*=...> cannot be changed
SCHEDULER-409	XML tag <*(1)*> expected, instead of <*(2)*>
SCHEDULER-412	Job chains can only be nested once
SCHEDULER-413	Nested job chains cannot be combined with distributed orders
SCHEDULER-414	There is a circular nesting of job chains: *(1)*
SCHEDULER-415	Error when trying to continue in next job chain: *(1)*
SCHEDULER-416	'*(1)*' is already known
SCHEDULER-417	Invalid name: '*(1)*'

Code	Text
SCHEDULER-419	New Process_class.max_processes=*(1)* is below current number of processes=*(2)*
SCHEDULER-420	Negative value is not allowed for *(1)*
SCHEDULER-422	'*(1)*' has already been added
SCHEDULER-424	'*(1)*' is being removed before '*(2)*'
SCHEDULER-426	'*(2)*' and '*(3)*' cannot be combined, because of duplicate order id '*(1)*'
SCHEDULER-427	Invalid MD5 hexadecimal coded password for HTTP user '*(1)*'
SCHEDULER-428	Error when reading base file '*(1)*' (*(2)*):*(3)*
SCHEDULER-429	Current name of '*(1)*' is fixed and cannot be changed to '*(2)*'
SCHEDULER-430	'*(1) (2)*' is unknown
SCHEDULER-431	Error when processing configuration directory:*(1)*
SCHEDULER-432	*(1)* has no name
SCHEDULER-433	'*(1)*' is not in any folder and cannot be replaced or removed
SCHEDULER-434	Error when closing '*(1)*':*(2)*
SCHEDULER-435	Standing order filename has not expected syntax jobchain,orderid.order.xml: *(1)*
SCHEDULER-436	Order id in <order> is different from order id in configuration filename
SCHEDULER-437	Attribute job_chain="*(1)*" is not empty and differs from Job_chain '*(2)*' in configuration filename
SCHEDULER-438	Invalid Job_chain_node for add_order() with state='*(1)*'
SCHEDULER-439	Error when removing after base file '*(1)*' has been removed:*(3)*
SCHEDULER-440	Definition of *(1)* is incomplete
SCHEDULER-441	Invalid value in <*(1) (2)*="*(3)*">
SCHEDULER-442	Attribute *(1)* cannot be combined with *(2)*
SCHEDULER-443	<month month="*(1)*"> has been defined already
SCHEDULER-445	Invalid weekday '*(1)*'

Code	Text
SCHEDULER-446	<weekday day="*(1)*" which="*(2)*" has been defined already
SCHEDULER-447	Error in *(1)*: *(2)*
SCHEDULER-448	File '*(1)*' cannot be read, because it is bigger than *(2)*MB
SCHEDULER-449	Syntax error in order parameters file *(1)*: Missing '=' after variable name
SCHEDULER-450	Java file '*(1)*' could not be deleted
SCHEDULER-451	Database check failed, database is not useable. *(1)*
SCHEDULER-452	Database does not correctly store test values:*(1)* (written value)*(2)* (read value)
SCHEDULER-453	Database does not respect rollback command
SCHEDULER-454	Remote configuration directories '*(1)*' and '*(2)*' refer to the same IP number
SCHEDULER-455	No configuration directory for '*(1)*'
SCHEDULER-456	Invalid response from supervisor
SCHEDULER-457	Remote JobScheduler '*(1)*' has not been registered
SCHEDULER-458	Remote JobScheduler has not sent a notification for *(1)*s and has been deregistered
SCHEDULER-459	Error in dependant after '*(1)*' has been loaded:*(2)*
SCHEDULER-460	*(1)* is centrally defined and cannot be locally overwritten
SCHEDULER-461	Path reaches beyond root (too many '..'): *(1)*
SCHEDULER-462	Attribute live_file= is not possible here. Use attribute file=
SCHEDULER-463	*(1)*: substituted '*(2)*' is a substitute
SCHEDULER-464	*(1)*: valid_from=""> is not before valid_to=""
SCHEDULER-465	'*(1)*' overlaps *(2)*
SCHEDULER-466	'*(1)*' is a substitute for another schedule and cannot be used directly
SCHEDULER-467	Attribute '*(1)*' is only valid with attribute '*(2)*'
SCHEDULER-468	Using this call is not possible in this context

Code	Text
SCHEDULER-469	try_hold_lock() or try_hold_lock_non_exclusive() has failed and call_me_again_when_locks_available() has not been called
SCHEDULER-470	Path is not absolute: *(1)*
SCHEDULER-471	Not supported character for encoding '*(1)*'
SCHEDULER-472	Unknown subsystem "*(1)*"
SCHEDULER-473	operation is not supported for distributed orders: '*(1)*'
SCHEDULER-474	Workload JobScheduler is not responding to UDP message since *(1)* (*(2)*s). Workload scheduler's configuration can not be updated.
SCHEDULER-475	java_options and java_class_path can only be set in primary XML configuration
SCHEDULER-477	Could not re-open log file. Following lines are not written to this file. *(2)*
SCHEDULER-478	Could not re-open '*(1)*' log file. Following lines are lost. *(2)*
SCHEDULER-479	No entry config/@spooler_id='*(1)*' found in configuration file 'scheduler.xml'.
SCHEDULER-480	No appropriate config element found in scheduler.xml - please remove attribute @spooler_id of the config-element or start JS with the appropriate -id option.
SCHEDULER-481	Method '*(1)*' with signature '*(2)*' is not implemented.
SCHEDULER-483	Order parameter 'scheduler.remote_scheduler' cannot be used in a cluster (option -exclusive or -distributed-orders)
SCHEDULER-484	Order parameter 'scheduler.remote_scheduler' can be used only for shell script jobs and is ignored for API jobs
SCHEDULER-485	Missing JobScheduler ID (spooler_id="" or -id)
SCHEDULER-486	Self-referencing dependency *(1)*
SCHEDULER-487	Role '*(1)*' needed for this operation
SCHEDULER-488	This remote JobScheduler '*(1)*' is not accessible: *(2)*
SCHEDULER-489	No remote JobScheduler is accessible. Waiting before trying again
SCHEDULER-490	The Windows operating system does not support Unix signals (like SIGTERM). kill_task with timeout > 0 is not supported
SCHEDULER-491	The process class (*(1)*) of an API task cannot be changed (to *(2)*) between two order steps
SCHEDULER-900	Scheduler *(1)* is starting with *(2)*, pid=*(3)*

Code	Text
SCHEDULER-902	state=*(1)*
SCHEDULER-904	Tasks have *(1)* seconds to terminate
SCHEDULER-906	Restarting Scheduler: *(1)*
SCHEDULER-907	Opening database: *(1)*
SCHEDULER-908	Column *(2)* is being added to database table *(1)*: *(3)*
SCHEDULER-909	Creating database table *(1)*
SCHEDULER-912	add_pid(*(1)*), process deadline is *(2)*
SCHEDULER-913	Previous history has been archived under *(1)*
SCHEDULER-914	end() called, task will terminate
SCHEDULER-915	Process event
SCHEDULER-916	idle_timeout is over, terminating task
SCHEDULER-917	Enqueued task has been loaded from database: id=*(1)* start_at=*(2)*
SCHEDULER-918	state=*(1) (2)*
SCHEDULER-919	Task *(1)* enqueued
SCHEDULER-920	Reading script again (processing again <include>)
SCHEDULER-921	Next period is *(1)*, Schedule '*(2)*'
SCHEDULER-922	No further period
SCHEDULER-923	Repeating at *(1)* due to delay_after_error
SCHEDULER-924	First start at the start of the period: *(1)*
SCHEDULER-925	Next start at *(2)* due to spooler_job.repeat=*(1)*
SCHEDULER-926	Next start at *(2)* due to <period repeat="*(1)*">
SCHEDULER-927	Next start time will be determined at beginning of the next period. This is at *(1)*
SCHEDULER-928	Next single_start at *(1)*

Code	Text
SCHEDULER-930	Task *(1)* started - cause: *(2)*
SCHEDULER-931	state=*(1)*
SCHEDULER-933	TCP connection accepted
SCHEDULER-934	Compiling Java source: *(1)*
SCHEDULER-935	*(1)* orders read from database
SCHEDULER-936	Replacing *(1)* now
SCHEDULER-937	Removing *(1)* now
SCHEDULER-938	Order will be processed at *(1)*
SCHEDULER-939	Replacing order with same id: *(1)*
SCHEDULER-940	Removing order from job chain
SCHEDULER-941	Removing order from job chain. Order is still being executed by *(1)*
SCHEDULER-942	add_or_replace_order(): order delayed until *(1)* has executed *(2)*
SCHEDULER-943	setback() called *(1)* times. Setting order to error state
SCHEDULER-944	End state reached - order will be repeated at *(2)* with state=*(1)*
SCHEDULER-945	No further job in job chain - order has been carried out
SCHEDULER-946	setback(): order has been set back *(1)* times, until *(2)*
SCHEDULER-947	setback(): order has been set back *(1)* times. This is more than *(2)* - the maximum for this job
SCHEDULER-948	Process *(1)* started
SCHEDULER-949	Job will be executed when a process of process class '*(1)*' becomes available
SCHEDULER-950	Scheduler has been registered
SCHEDULER-951	Daylight saving time has begun
SCHEDULER-952	Daylight saving time has ended, now it's standard time
SCHEDULER-953	Still sleeping *(1)* seconds ...

Code	Text
SCHEDULER-954	XSLT stylesheet *(1)*="*(2)*" returns:
SCHEDULER-955	UDP message from *(1)*: *(2)*
SCHEDULER-956	Scheduler expects commands from *(1)* port *(2)*
SCHEDULER-957	Closing database
SCHEDULER-958	Waiting *(1)* seconds before reopening the database
SCHEDULER-960	Windows service handler: control=*(1)* event=*(2)*
SCHEDULER-961	Protocol starts in *(1)*
SCHEDULER-962	Protocol ends in *(1)*
SCHEDULER-963	Copying protocol file to *(1)*
SCHEDULER-964	New *(1)* The web service has created a new order and handed it over to the job chain.
SCHEDULER-965	Executing command *(1)*
SCHEDULER-966	Command answer: *(1)*
SCHEDULER-967	start_new_file(): protocol file is being closed now
SCHEDULER-968	start_new_file(): intermediate protocol file is: *(1)*
SCHEDULER-969	Less than min_tasks=*(1)* are running. New tasks will be started. Reason: *(2)*
SCHEDULER-970	*(1)* ended immediately after start, so min_tasks=*(2)* doesn't lead to new tasks
SCHEDULER-972	Still waiting until *(1)* (*(2)*) ...
SCHEDULER-973	Killed task does not stop the job because of ignore_signals="..."
SCHEDULER-974	Last error does not stop the job if the task aborts (after kill or crash) with any signal listed in ignore_signals="". In this case, expect warning SCHEDULER-279
SCHEDULER-975	This file path with ';' will not be included in changed_directory: *(1)*
SCHEDULER-976	This directory path with ';' will not be included in triggered_files: *(1)*
SCHEDULER-977	Job is not stopping because of <job stop_on_error="no">. Task error was: *(1)*

Code	Text
SCHEDULER-978	Job is stopping because of <job stop_on_error="yes">. Task error was: *(1)*
SCHEDULER-979	Removing file *(1)*
SCHEDULER-980	Moving file *(1)* to *(2)*
SCHEDULER-981	File on blacklist has been removed
SCHEDULER-982	File has been removed, so the file order is being removed too
SCHEDULER-983	New file, *(2)*: Added *(1)*
SCHEDULER-984	Recovered from previous error in directory *(1)*
SCHEDULER-987	Starting process: *(1)*
SCHEDULER-989	*(1)* cannot be removed now, it will be done later
SCHEDULER-990	Adding *(1)*
SCHEDULER-991	Order has been suspended
SCHEDULER-992	Order is no longer suspended, next start *(1)*
SCHEDULER-994	No heart beat for *(2)* seconds (*(1)*), expecting heart beat within *(3)* seconds
SCHEDULER-995	No heart beat for *(2)* seconds (*(1)*), ignored for *(3)* seconds because of recent database reconnect
SCHEDULER-996	No heart beat for *(2)* seconds (*(1)*) - JobScheduler seems to be dead
SCHEDULER-997	Making up an extra heart beat
SCHEDULER-999	Scheduler has been terminated properly
SCHEDULER-805	No JobScheduler is active
SCHEDULER-806	This JobScheduler is active and exclusive now
SCHEDULER-807	Using database product *(1)*
SCHEDULER-809	Dead JobScheduler record has been removed
SCHEDULER-811	Executing command read from database: *(1)*
SCHEDULER-812	Just before processing, order record in database has been occupied or removed

Code	Text
SCHEDULER-813	Order is occupied by JobScheduler '*(1)*'
SCHEDULER-814	Inactive JobScheduler '*(2)*' has the higher backup precedence *(1)* (*(3)*)
SCHEDULER-815	Task should end but it has just been started with an order attached, so one step will be done
SCHEDULER-816	Unable to release occupation in database
SCHEDULER-817	Missing order record in database
SCHEDULER-819	Scheduler becomes active
SCHEDULER-820	Watching heart beat of that Scheduler
SCHEDULER-821	Scheduler member '*(1)*' seems to be active, claiming its last heart beat was *(2)*s ago (pid=*(4)*, *(3)*)
SCHEDULER-822	Scheduler member '*(1)*' seems to be exclusive, claiming its last heart beat was *(2)*s ago (pid=*(4)*, *(3)*)
SCHEDULER-823	Dead JobScheduler member '*(1)*' has resurrected
SCHEDULER-825	No exclusive JobScheduler is running
SCHEDULER-826	That JobScheduler has terminated
SCHEDULER-827	Own heart beat is late: next_heart_beat has been announced for *(1)* (this is *(2)* seconds late)
SCHEDULER-829	Releasing occupied order *(1)*:*(2)*
SCHEDULER-830	Because all JobScheduler tasks have been killed, the order in database has not been updated. Only the occupation has been released
SCHEDULER-831	Waiting *(1)*s for start of non-backup Scheduler
SCHEDULER-832	This is a backup Scheduler, waiting for a non-backup Scheduler
SCHEDULER-833	Watching heart beat of that exclusive Scheduler, standing by to take over operation
SCHEDULER-834	Active JobScheduler has terminated properly
SCHEDULER-835	This JobScheduler is becoming exclusive now
SCHEDULER-836	Deactivating that dead Scheduler
SCHEDULER-837	Taking exclusiveness from that Scheduler
SCHEDULER-838	*(1)*. heart beat detected

Code	Text
SCHEDULER-839	This order has been replaced
SCHEDULER-841	It has been requested that the exclusive operation will not be continued
SCHEDULER-842	Task is going to process *(1)*, state=*(2)*, on JobScheduler *(3)*
SCHEDULER-843	Task has ended processing of *(1)*, state=*(2)*, on JobScheduler *(3)*
SCHEDULER-844	Scheduler script is not yet active. Operation is executed without the call of function '*(1)*'
SCHEDULER-845	Task ended without processing the order. The order remains in job's order queue in the same state
SCHEDULER-846	After task exception and due to stop_on_error='no', the order has been moved to error_state='*(1)*'
SCHEDULER-847	Please use new name '*(1)*' instead of old '*(2)*'
SCHEDULER-848	Task pid=*(1)* started for remote scheduler *(2)*
SCHEDULER-849	Timeout is not possible for a subprocess running on a remote host (it cannot be killed), pid=*(1)*
SCHEDULER-850	After lost connection to remote scheduler, process *(1)* is going to be killed
SCHEDULER-851	After possibly volatile error (number *(1)*), the statement which caused the error will be repeated: *(2)*
SCHEDULER-852	Order not found in database
SCHEDULER-853	Order in database could not be updated or deleted.
SCHEDULER-854	Closing lock *(1)* while held by *(2)*
SCHEDULER-855	Task is now holding *(1)* *(2)*
SCHEDULER-856	*(1)* has been released
SCHEDULER-857	*(1)* has been released, *(2)* non-exclusive holders remaining
SCHEDULER-858	Not longer waiting for lock *(1)*
SCHEDULER-859	Due to action='next_state' the state '*(2)*' has been skipped. Next state is '*(1)*'
SCHEDULER-860	Waiting for '*(1)*', *(2)*, on place *(3)* (*(4)*)
SCHEDULER-861	'*(1)*' has been removed
SCHEDULER-862	Continuing in '*(1)*'

Code	Text
SCHEDULER-863	Continuing from '*(1)*'
SCHEDULER-871	Closing process class before *(1)*
SCHEDULER-872	New '*(1)*' shares order IDs with *(2)*
SCHEDULER-873	Job_chains *(2)* share order IDs because '*(1)*' has been removed
SCHEDULER-874	Order_id_space has been closed
SCHEDULER-875	Order_id_space has been closed because '*(1)*' has been removed
SCHEDULER-876	Temporary files cannot be deleted. Still trying ..., *(1)*
SCHEDULER-877	Temporary files have been deleted now
SCHEDULER-878	Temporary files cannot be deleted: *(1)* In a windows environment it is possible that files can not be deleted, because another process (e.g. a virus scanner) blocked them. JobScheduler try to delete files for a while and if it's failed you'll get the message SCHEDULER-878. This behaviour should not be the normal case, but it is not excluded.
SCHEDULER-879	Deactivating old cluster member with same ID
SCHEDULER-880	Error when creating database index is ignored
SCHEDULER-881	Error when reading stdout file is ignored: *(1)*, *(2)*
SCHEDULER-882	Configuration directory '*(1)*' has been removed:*(2)*
SCHEDULER-884	Configuration directory has been deleted
SCHEDULER-885	Task ends because '*(1)*' is being removed
SCHEDULER-886	Lock will be removed later, it is held by '*(1)*'
SCHEDULER-887	More lock holders than new max_non_exclusive=*(1)*: *(2)*
SCHEDULER-888	*(1)* cannot be replaced now, it will be done later
SCHEDULER-889	These filenames yield the same object name: '*(1)*' and '*(2)*'. The second is ignored
SCHEDULER-890	*(2)* is going to be removed because its configuration file '*(1)*' has been removed
SCHEDULER-891	Configuration file '*(1)*' (*(2)*) is going to be be loaded into this new *(3)*

Code	Text
SCHEDULER-892	This *(3)* is going to be replaced due to changed configuration file '*(1)*' (*(2)*)
SCHEDULER-893	*(1)* is '*(2)*' now
SCHEDULER-894	'*(1)*' will be removed later, it is used by *(2)*
SCHEDULER-895	Configuration directory '*(1)*' does not exist, it will be ignored
SCHEDULER-897	Error in replacement ignored::*(2)*
SCHEDULER-898	Folder is going to be removed because its configuration directory has been removed
SCHEDULER-899	Scheduler has no UDP port, so updated configuration files on supervisor will not be noticed
SCHEDULER-701	Replicating '*(1)*'
SCHEDULER-702	Removing '*(1)*'
SCHEDULER-703	Local configuration file is ignored, the central configuration file is used instead
SCHEDULER-704	Order has reached its end_state='*(1)*'
SCHEDULER-705	Substitute '*(1)*' is valid now
SCHEDULER-706	Standard '*(1)*' is valid now
SCHEDULER-707	Creating configuration directory '*(1)*'
SCHEDULER-708	Adding '*(1)*'
SCHEDULER-709	Killing descendant pid=*(1)* (*(2)*)
SCHEDULER-710	Delayed command <scheduler_log.log_categories.reset> has been executed
SCHEDULER-711	Step has run for *(2)*s, which is shorter than the expected duration of *(1)*
SCHEDULER-712	Task runs longer than the expected duration of *(1)*
SCHEDULER-713	Process handle *(1)* is already registered
SCHEDULER-714	Scheduler *(1)* is already registered via connection *(2)*. This connection will be closed now. New connection is *(3)*
SCHEDULER-715	The command-line option configuration-directory '*(1)*' is not a directory
SCHEDULER-717	Remote commands are not allowed for the current licence-key(s)

Code	Text
SCHEDULER-718	Using '*(1)*' configuration directory '*(2)*'
SCHEDULER-720	job_chain *(1)* is deactivated (set max_orders to activate it)
SCHEDULER-721	Scheduler is not responding quickly, a microstep took *(1)*s
SCHEDULER-722	Preceding content truncated (field length is limited to *(1)* bytes)
SCHEDULER-723	Cluster member '*(1)*' has been advised for processing of this order
SCHEDULER-724	Error ignored when trying to obtain directory change notifications for *(1)*: *(2)*
SCHEDULER-725	Persistent Order deleted from database
SCHEDULER-727	Keep-alive package sent to Agent

A.2 Messages for the Separate processes Package

Code	Text
Z-REMOTE-100	Separate process: *(1)*
Z-REMOTE-101	Separate process: pid=*(1)*: Connection lost
Z-REMOTE-102	Separate process: Unknown proxy id *(1)*
Z-REMOTE-103	Separate process: Not a registered object: *(1)*
Z-REMOTE-104	Separate process: Invalid session id *(1)*
Z-REMOTE-105	Separate process: Unknown session command, failure in communication
Z-REMOTE-106	Separate process: Unknown session class, failure in communication
Z-REMOTE-107	Separate process pid=*(1)*: Invalid response
Z-REMOTE-108	Separate process: Variant type *(1)* is not transmittable
Z-REMOTE-109	Separate process: Failure in communication (message is too short)
Z-REMOTE-111	Separate process: Thread *(1)* tries to use connection owned by thread *(2)*
Z-REMOTE-112	Separate process: Program did not fetch operation's result: *(1)*

Code	Text
Z-REMOTE-113	Separate process: pop_operation with different method "*(2)*" ("*(1)*" expected) or different object
Z-REMOTE-114	Separate process: Program did not fetch result of an finished operation, method=*(1)*
Z-REMOTE-115	Separate process pid=*(1)*: Message length *(2)* too big or invalid
Z-REMOTE-118	Separate process pid=*(1)*: No response from new process within *(2)*s
Z-REMOTE-119	Separate process: Two operations on the same connection. First: *(1)*, second: *(2)*
Z-REMOTE-120	Separate process: Unknown object id *(1)*
Z-REMOTE-121	Separate process: pop_operation() on empty stack, method=*(1)*
Z-REMOTE-122	Separate process pid=*(1)*: Caller has killed process
Z-REMOTE-123	Separate process pid=*(1)*: Process lost
Z-REMOTE-124	Separate process: Named arguments not implemented
Z-REMOTE-125	Separate process: DISPID *(1)* unknown
Z-REMOTE-126	Separate process: Default proxy for *(1)* has no properties
Z-REMOTE-127	Separate process: More data as announced received. Failure in communication
Z-REMOTE-128	Separate process: v.vt is not the value of SafeArrayGetVartype()
Z-REMOTE-129	Separate process: Blocking invocation on asynchronous connection rejected. Object=*(1)*, dispid=*(2)*
Z-REMOTE-130	Connection from unexpected host *(1)*

A.3 Messages for the Charset Package

Code	Text
Z-CHARSET-001	Unknown character encoding: *(1)*
Z-CHARSET-002	Missing character encoding

Appendix B: Change Log

Revision	Date	Note
2.1.2.6164	2010-07-07	**JS-538: job_chain should be not run parallel** The new attribute max_orders for element job_chain determine how many orders can execute a job_chain simultaneous. Is this attribute missing the number of running orders for the chain is unlimited. max_orders="1" executes a job_chain exclusive for one order, before a new order for this chain will be run.
2.1.1.6132	2010-06-22	**JS-462: The command-line option configuration-directory (live folder) is available as documented** The live folder can also be set by the command-line option configuration-directory.
2.1.1.6103	2010-05-28	**JS-506: Command <show_state what='folders no_subfolder'> shows names of the subdirectories.** The command <show_state what='folders no_subfolder'> responses the names of the subdirectories without its contents.
2.1.1.6101	2010-05-07	**JS-343: The attribute @absolute_repeat supports now the first start time too.** If @absolute_repeat is set in a run_time then sometimes the first start time was not correct calculated.
2.1.1.6096	2010-05-04	**JS-380: JobScheduler not 'freeze' any longer if a job is set to 'suspend'.** API jobs had once a while the problem that the JobScheduler 'freezed' if a job is set to 'suspend'. This was happend when the 'suspend' was called inside of an operation (spooler_start, spooler_process ...).

Revision	Date	Note
2.1.1.6089	2010-04-28	**JS-481: TCP connections which are required no more are now unfailing released.** On open a new TCP connection there is checked if an already registered connection for the same host/port is no longer required.
2.1.1.6078	2010-04-27	**JS-471: More than 64 parallel processes are possible on Windows.** On Windows existed a limit of 64 parallel running processes. The possible number of processes is now analog to Unix.
2.1.0.6078	2010-04-13	**JS-409: Start time of a job is arbitrary set.** The start time of a job can be set more than 24 hours in the future.
2.1.0.6078	2010-04-13	**JS-444: Includes are extended in the running JobScheduler instance for monitor scripts too.** Files which are denotes with the live_file attribute are reloaded after changes. For this the file must be LOCAL stored.
2.1.0.6078	2010-04-13	**JS-457: Timestamp of the configuration files are marked as UTC** The timestamp of the configuration files are marked as UTC.
2.1.000.6070	2010-03-23	**JS-421: SCHEDULER_RETURN_VALUES works at UNIX when a monitor is assigned too**
2.1.0.6065	2010-04-13	**JS-421: Environment variables are passed properly between scripts.** Shell scripts can read environment variables which are set from preceding scripts.
2.0.223.6010	2009-06-12	**New API Calls: Order.setback_count and Job.setback_max** • `Order.setback_count` • `Job.setback_max`

Revision	Date	Note
2.0.222.6007	2009-05-07	**<job warn_if_shorter_than and warn_if_longer_than>** The JobScheduler issues a warning when the time a job step takes lies outside a predefined time span. • `<job warn_if_shorter_than="…">` • `<job warn_if_longer_than="…">`
2.0.221.5985	2009-03-20	**Windows: 'kill' Stops All Child Proceses** `<kill_task>` applied to a shell job now stops all child processes as already happens on Unix systems.
2.0.221.5980	2009-03-10	**Monitor for Shell-Jobs** A `<monitor>` Can Now Also be Specified for Non-API Jobs. • `<monitor>` **Database Problems Caused by the Winter / Summer Time Change Have Now Been Solved**
2.0.220.5972	2009-03-02	**Standing Orders are No Longer Immediately Started With the Next JobScheduler Run After kill -9** **<settings> in <job>** • `<settings>` • `<mail_on_error>` • `<mail_on_warning>` • `<mail_on_success>` • `<mail_on_process>` • `<mail_on_delay_after_error>` • `<log_mail_to>` • `<log_mail_cc>` • `<log_mail_bcc>` • `<log_level>` • `<history>` • `<history_on_process>` • `<history_with_log>`

Revision	Date	Note
2.0.220.5970	2009-02-25	**JobSchedulers Running as Unix-Daemons Now Divert stdout/stderr to a File** As one would expect from a program riunning as a Daemon, the JobScheduler now closes stdout and stderr diverts the output to the `scheduler.out` file. Note that this file must be writable. • See the `-service` option.
2.0.219.5968	2009-01-27	**libhostjava.so for HP-UX/Itanium is now Integrated in the JobScheduler** In contrast to HP-UX/PA-RISC, the Itanium libhosthava.so is now integrated in the JobScheduler. This means that the "com_construct problem" no longer occurs.
2.0.219.5966	2009-01-21	**Tasks Stopped on Unix Systems With 'kill' No Longer Lead to a Task Error** This means that behavious on Unix systems is now the same as on Windows. **JS-321: Backup JobSchedulers Take Over Job, Task, Job Chain and Order Settings Once More** The cluster_member_id database columns are no longer filled when operating with a backup JobScheduler. They are only filled when operating with distributed orders, so that records are now only directly allocated to cluster members.
2.0.219.5963	2008-12-12	**JS-305: Resetting an Order** • `<modify_order action="reset">`

Revision	Date	Note
2.0.218.5961	2008-11-18	**<script language="shell">: .cmd file Generated in OEM Coding** • `<script language="script">` Windows requires that verlangt OEM coding (CP437) is used for batch files. When a character from `<script>` cannot be interpreted, the JobScheduler terminates the job with a `SCHEDULER-471` message. Note, however, that when reading the XML configuration file, the JobScheduler ignores chararcters that are not in Latin1: see here . `[ERROR]` `SCHEDULER-471` Not supported character for encoding "
2.0.217.5956	2008-11-12	**JS-316: New Environment Variables for Non-API Jobs** • `<process>`
2.0.216.5953	2008-11-05	**Transaction Isolation Changed for Sybase** *read committed* is now set for Sybase. **<start_job force="no">: Start a Task Without Unstoppping the Job** • `<start_job force="no">` **Windows Z-ASYNC-SOCKET-001 Error Message Corrected** An internal security test, which was only intended for Unix systems, sometimes stopped jobs from starting on Windows systems. **Errors from 2.0.214.5927 Corrected (Jobs in the Database)** In addition, the number of characters per column in the `SCHEDULER_JOB_CHAIN_NODES` table has been reduced.

Revision	Date	Note
2.0.215.5938	2008-10-28	**Minor Changes Made to the Log Categories**
2.0.215.5937	2008-10-22	**Special License Keys Now Allow the JobScheduler to Return the Database Password** `<licence.use>` can return the license key. The JobScheduler no longer supresses the characters which begin with `-password=` in its response to `<show_state>`. • `<licence.use>`. • `<state db="…">`.
2.0.214.5931	2008-10-20	**The General, Central Configuration Directory Can Be Deleted** When the general, central configuration directory is deleted, then the supervisory JobScheduler deletes the corresponding objects in all connected JobSchedulers. • `<config central_configuration_directory="…">`. • `<config supervisor="…">`.
2.0.214.5927	2008-10-16	**Jobs, Job Chains and Job Chain Nodes that have been Stopped Remain Stopped after a New Start** The state of jobs, job chains and job chain nodes is now stored in the database. When one of these object types is deleted then the corresponding database entry will also be deleted. • `<modify_job cmd="stopped">`. The JobScheduler is being shut down or restarted and any job modifications made will not be noted in the database. • `<job_chain.modify state="stopped">` • `<job_chain_node.modify action="stop">`: The value of `action` is retained.

Revision	Date	Note
2.0.213.5922	2008-10-07	**Nested Job Chains With Capital Letters Now Work** Capital letters in the names of parent job chains could cause the JobScheduler to fail. This has now been cured. Capital letters in job chain names are now treated the same as lower case ones. Job chain names are left in upper case in the database to ensure that orders can be correctly identified. (This requires that the case in which the job name is written remains unchanged.) Job chain names should be written in the database in standard form - i.e. in lower case.
2.0.213.5918	2008-10-04	**Viewing on the Main Protocol in a Browser No Longer Cut Off** The main protocol is no longer cut off in a browser window when the main protocol had been sent because of an error which had taken place as the JobScheduler was starting.

Revision	Date	Note
2.0.213.5914	2008-10-02	**$-Substitition Can Be Switched Off Using ** Placing a backward slash (»`\`«) before »`$`« returns a »`$`« character without a variable having to be set. See (page 106). **<kill_task> now stops all child processes of non-API processes running on Unix systems** • `<kill_task>` The `SCHEDULER-277` and `SCHEDULER-282` messages will no longer be sent. **New Commands for Controlling Log Categories** • `<scheduler_log.log_categories.show>` • `<scheduler_log.log_categories.set>` • `<scheduler_log.log_categories.reset>`
2.0.212.5882	2008-09-11	**".." is Possible in all Paths** Paths to JobScheduler objects (job, job chain, etc.) can now use ".." to point to a higher directory level. For example, in • `<job_chain_node job="…">`
2.0.211.5876	2008-09-07	**Implementation of Nested Job Chains Revised** **scheduler.log Shows the Day of the Month in the First Column** **Diagnose Report on Process Starting Error** When a non-API process cannot be started on a Unix system (error on `execvp`), then the JobScheduler returns the `execvp` parameter list and environment variabes. • `<process>` • `<script language="shell">`

Revision	Date	Note
2.0.210.5864	2008-08-22	**Modification for Nested Job Chains** `<add_order state="…">` and `<add_order end_state="…">` can now be used on nested job chains.

2.0.210.5860	2008-08-14	**Perl-API for Unix (libsosperlscript.so) Systems Improved**
		
		• `Log.level` : $spooler_log->LetProperty('level', ...) functions.
		• The Perl interface now attempts to recognise each unknown type as a string, so that subsequent sequences, whose second `$a` command converts a SVt_IV into a SVt_PVIV, function:
		
		`my $a = 1;`
		`my $x = "$a";`
		`$spooler_log->info( $a )`
		
		Run_time Class 'Possible for Java
		
		`Run_time` can now be used in Java.

2.0.210.5838	2008-07-25	**New SCHEDULER_TASK_ID Environment Variable** See `<process>` (page 0).

Revision	Date	Note
2.0.210.5837	2008-07-22	**XML Commands for Setting and Reading JobScheduler Parameters Added** • `<param>` • `<param.get>` • `<params>` • `<params.get>` **Treatment of Holidays: <period when_holiday="…">** • `<period when_holiday="…">` • `<run_time when_holiday="…">` The JobScheduler can now handle holidays up to 2038. **Weekdays such as Saturday and Sunday can now be Declared as Holidays** • `<holidays><weekdays>` **Orders with Included Parameter Files are Now Possible** `<order>` `<params><include>` is now possible. Included files can now be monitored for file based orders. **<job timeout="…"> is Now Valid for Non-API Jobs** • `<job timeout="…">`
2.0.208.5817	2008-07-02	**File Orders Speeded Up** The JobScheduler now feeds file orders more quickly to a job chain when the first job in the job chain allows more than one task (`<job tasks="…">`).

Revision	Date	Note
2.0.208.5814	2008-06-30	**Double "To:" Removed From Mail Directory** In order to ensure compatibility with an older mail program, files written to the (`factory.ini` (section `[spooler]`, entry `mail_queue_dir=…`)) e-mail directory were given an additional "To:" line. This is no longer added. The old mail program should be replaced by a current version (mail.jar oder w3jmail44.dll). **Variable_set.set_value() Also Possible in Java** The `Variable_set.value` call can now be made in Java. This call can now be made in all supported script languages and replaces `Variable_set.var`, which was not possible in JavaScript. **spooler_log.new_filename** `Log.new_filename` can now be used on Unix systems. **Documentation Extended** `Log.mail_on_process` extended and texts from settings.xml are shown on the API pages.
2.0.208.5811	2008-06-26	**API Extended to Include Locks** • `Task.try_hold_lock()` • `Task.try_hold_lock_non_exclusive()` • `Task.call_me_again_when_locks_available()` • The following states have been added to the `<task state="…">` response element: `opening`, `opening_waiting_for_locks` and `running_waiting_for_locks`.
2.0.207.5799	2008-06-09	**Changed to Microsoft Visual Studio 2008**

Revision	Date	Note

2.0.206.5789	2008-05-25	**spidermonkey.dll and libspidermoneky.so: Behaviour with Boolean true Improved**
		
		Boolean true now always returns the same internal value, so that the following code will now work:
		
		`spooler_log.mail_on_success = true;`
		`var a = spooler_log.mail_on_success;`
		`if( a != true ) spooler_log.error( a + "!=" + true );  // Okay ab Spidermonkey 1.7.0.5789`
		
		Documentation Corrected
		
		• `Log.last` is only readable.
		• `Log.last_error_line` is only readable.

2.0.206.5787	2008-05-19	**Conversion of the SCHEDULER_HISTORY.END_TIME Database Column to Accept NULL** Modification to suit MySQL 5, which writes "0000-00-00" in the event of missing field values, when a column is declared as NOT NULL. **Order.priority is now interpreted correctly** • `Order.priority`: a higher priority causes an order to be positioned further *forward* in the order queue. **Distributed Orders Make a Single History Entry Once Again** The JobScheduler has been writing history entries for every step of distributed orders completed since version 2.0.194.8488 (2008-01-05). This has now been corrected.

Revision	Date	Note
2.0.205.5781	2008-05-09	**Order.job deleted** • `Order.job` extended. **<show_state what="source"** • `<show_state what="source">` shows the XML source. **<schedule.remove> Remove a Schedule** • `<schedule.remove>` deletes a schedule.
2.0.204.5773	2008-05-06	**JS-279: Additional Environment Variables for Non-API Jobs** • `<process>`
2.0.203.5764	2008-04-29	**JS-271: Adding an XML Document to a Configuration Directory: <modify_hot_folder>** • `<modify_hot_folder>` **JS-245: New XML Element <schedule>: Named <run_time>** • `<schedule>` • `<schedules>` • `<show_state what="schedules">` **JS-274: Job_chain.title** • `<job_chain title="…">` • `Job_chain.title`
2.0.202.5728	2008-04-21	**Problem When Starting Two Cluster JobSchedulers Simultaneously Cured** When two JobSchedulers in the same clusters are started at the same time, one of them could fail to start. This has now been corrected.
2.0.202.5723	2008-04-18	**JS-265: Order.end_state** • `<order end_state="…">` • `<add_order end_state="…">` • `<modify_order end_state="…">` • `Order.end_state`
2.0.201.5696	2008-04-02	**Modifications for Sybase Adaptive Server Enterprise**

Revision	Date	Note
2.0.201.5692	2008-03-31	**JS-60: <script> </script> functions as a <script/>** The JobScheduler now ignores empty text nodes (only blank spaces, new line and tabulators) in the `<script>` XML element. When the `<script>` element only contains one empty text node, this is now treated as unspecified source code.
2.0.201.5684	2008-03-14	**Job Log Files are no Longer Held Open** The JobScheduler now closes job log files after writing in them and reopens them as required.
2.0.201.5679	2008-03-13	**JS-150: Modification of the SCHEDULER_ORDER_HISTORY.END_TIME Column in an Access Database** With JS-150, the `SCHEDULER_ORDER_HISTORY.END_TIME` database column needs to be able to accept `NULL`. The JobScheduler checks this on starting and modifies the column. Access demands that a special procedure is followed: the JobScheduler creates a copy of the column, which it then copies back into the modified column. This operation is carried out once as the JobScheduler is starting and should not be interrupted.
2.0.201.5678	2008-03-07	**JS-150: New ERROR, ERROR_CODE and ERROR_TEXT columns in the Order History** When a task error occurs, the JobScheduler now records information about the error in the new columns in the `SCHEDULER_ORDER_STEP_HISTORY` table.

Revision	Date	Note

2.0.201.5671	2008-02-28	**The day= attribute can now be specified more than once**
		
		Days in `<weekdays>` and `<monthdays>` can now be repeated. The JobScheduler now combines the times specified. The `SCHEDULER 444` message has been dropped.
		
		• `<weekdays>` `<day day="…">`
		• `<monthdays>` `<day day="…">`
		• `<ultimos>` `<day day="…">`
		
		`<monthdays>`
		`    <day day="12 13">`
		`        <period`
		`single_start="13:12"/>`
		`    </day>`
		`    <day day="12">`
		`        <period`
		`single_start="12:00"/>`
		`    </day>`
		`</monthdays>`
		
		This results in starts at 12:00 and 13:12 on the 12th of the month.

Revision	Date	Note
2.0.200.5665	2008-02-25	**<include include_path="...">: Environment variables can now be called** • <config include_path="$HOME/..."> **JS-215: <params> mit <include>** Parameters saved in files can now be included. <param> can now be addressed using an XPath expression. • <params><include> **JS-240: <include live_file="…">** • <include live_file="…"> The new live_file= attribute can be directly used under: • <job><description> • <job><params> • <job><run_time> • <job> <run_time><holidays> • <holidays> and • <script> . A change in the file specified in live_file= under <job><params> causes the job to be re-read. **JS-222: New job.folder_path() call** • Job.folder_path
2.0.199.5618	2008-02-11	**JS-221: Cache for central configuration and handling of local configurations** Configurations replicated by a supervisory JobScheduler are stored in the new ./config/cache directory of a local JobScheduler. The JobScheduler merges central and local configurations. A centrally configured object cannot be overwritten locally. Only 25 TCP connections are now available on Windows systems because of the additional directory monitoring. • <config supervisor="…">

Revision	Date	Note
2.0.198.5596	2008-02-04	**JS-222: New methods added: Spooler.configuration_directory and Job.folder.path** • `Spooler.configuration_directory` • `Job.configuration_directory`

Revision	Date	Note
2.0.197.5594	2008-02-04	**JS-255: Process classes also now possible for non-API jobs** A process class can now be assigned to non-API jobs (default: the process class with an empty name `""`). This means that the process limits set in this process class also applies to non-API jobs. The default limit has been increased from 10 to 30. • `<job process_class="…">` • `<process_class max_processes="30">` **JS-213: Non-API jobs can be executed on remote JobSchedulers** A process class for job execution by a remote JobScheduler can now be assigned to non-API jobs. • `<process>` • `<script language="script">` • `<process_class remote_scheduler="…">` **JS-205: Job.name now returns the job path** `Job.name` no longer returns the name but instead the job path. **JS-204: Central Configuration** A JobScheduler running as Supervisor can replicate configuration files & directories from a central configuration (page 92) to other JobSchedulers. Configuration changes made in this manner on a local host are immediately forwarded by the Supervisor to remote JobSchedulers. Any changes become effective without the remote JobScheduler having to be restarted. • `<config supervisor="…">` • `<config central_configuration_directory="…">` • (page 92)

Revision	Date	Note
		JS-203: stdout and stderr are logged in real time Output to `stdout` and `stderr` are no longer written at the end of a task but as the task is being executed. This happens each second. Should no output be generated within this time then the interval is extended to 10 seconds. For API tasks, which are executed on a remote JobScheduler (`<process_class remote_scheduler="…">`), output is not written to the log file after the script has ended (that is just before the process is ended). **The Adaptive Server Enterprise Database from Sybase is now supported** The order identifier and job chain name columns have been reduced from 255 to 250 characters for all databases. **JS-227: Orders repeated with <run_time> are now continued after midnight** Up till now orders had ended at 24:00. (a result of revision JS-123 in October 2007.)
2.0.196.5515	2008-01-14	**JobScheduler Ready for AIX**
2.0.196.5515	2008-01-14	**JS-219: HTML Pages are Now Possible via a HTTP-Proxy Virtual Directory**
2.0.196.5515	2008-01-13	**JS-220: Spidermonkey 1.7** The `libspidermonkey.so` and `spidermonkey.dll` modules have been updated to version 1.7.0. SpiderMonkey 1.7.0 Change Log
2.0.195.5499	2008-01-11	**JS-212: Element <params> under <config>** `<params>` under `<config>` can be used to set parameters which are valid Scheduler-wide and which can be called up using `Spooler.variables`.

Revision	Date	Note

2.0.194.5488	2008-01-05	**JS-150: New scheduler_order_step_history Database Table**
		
		A new entry is made to this table for each step made in an order. The entry is made at the beginning of the step and completed at its end.
		
		The first order step is now added to the `scheduler_order_history` table as soon as the order is started and additional entries to the order history are completed after the order itself has been carried out.
		
		The `scheduler_order_history.end_time` column now accepts `NULL`.
		
		The `scheduler_order_step_history` table now has the following structure (syntax for PostgresQL):
		
		`CREATE TABLE SCHEDULER_ORDER_STEP_HISTORY`
		`(`
		`    "HISTORY_ID" integer not null,          // Link to SCHEDULER_ORDER_HISTORY.HISTORY_ID`
		`    "STEP"       integer not null,          // 1, 2, 3, ...`
		`    "TASK_ID"    integer not null,          // Link to SCHEDULER_TASKS.TASK_ID`
		`    "STATE"      varchar(100) not null,     // Order state`
		
		`    "START_TIME" timestamp not null,`
		`    "END_TIME"   timestamp,`
		`    primary key( "HISTORY_ID", "STEP" )`
		`)`
		
		• `factory.ini` (section `[spooler]`, entry `db_order_step_history_table= …`)

Revision	Date	Note
2.0.194.5476	2008-01-05	**New scheduler_history.pid Database Column** The new `scheduler_history.pid` database column is used for the identification of the operating system processes. Database declaration: `"PID" integer`
2.0.193.5476	2008-01-02	**Central Configuration** `<config central_configuration_directory="…">` The supervising JobScheduler can now overwrite the local configurations of other JobSchedulers by a central configuration (page 92) • `<config supervisor="…">` • `<config configuration_directory="…">`
2.0.192.5444	2007-12-21	**<file_order_source> Without regex= Could Cause the JobScheduler to Abort** An empty `regex=` attribute in `<file_order_source>` could cause the JobScheduler to abort, after a file name had been set on the blacklist (because it is still present after an order has been completed). This has been resolved. Interim solution in the current JobScheduler version: `regex="."`
2.0.192.5442	2007-12-20	**JS-178: Large Responses to an API Method Cause a Task to Fail** A call of, for example, `Spooler.execute_xml()` witha large return value (over 100kB) could cause the call to fail. This has now been cured.

Revision	Date	Note
2.0.192.5438	2007-12-19	**Changes to a Job Configuration no Longer Stops File Orders on Unix Systems** On Unix systems, a change to the configuration of the first job in a job chain after `<file_order_source>` would stop the file order source. This has been sorted. **<show_order>** The `<show_order>` command could cause the JobScheduler to abort, when an order was no longer available. This has now been cured. **JS-208: modify_job cmd='reread' has been Re-included as a Null-Operation** The command does not do anything as `<include>` is re-read at the start of every job.
2.0.192.5429	2007-12-11	**JS-52: XML Errors on Reading an Order's XML Document** XML errors could occur when reading orders, depending on the block size used in the database. This should now be cured.
2.0.192.5428	2007-12-11	**JS-206: <script language="perl"> Returns Errors with Line Number and Path** An error in a (`<script language="perl">`) will now generally be reported with the path and line number (for `<include>`). On Unix systems, the position appears in the stderr output. On Windows systems, in the event of some errors ("undefined value"), the number of the first line of the section will be returned but not that of the line at which the error occurs. This appears to be a peculiarity of ActivePerl.

Revision	Date	Note
2.0.192.5407	2007-11-30	**Job.name Now Returns the Complete Path** `Job.name` no longer returns the job name, but its path, without the initial backslash.
2.0.192.5402	2007-11-27	**Database Test Now Reduced to ASCII Characters** Because some PostgresQL installations do not support Latin-1 characters, the JobScheduler now only tests the ASCII character set (basic latin).
2.0.192.5397	2007-11-20	**Variables With Non-Latin1 Characters** Internal XML documents were being made invalid because variables such as `ERASE` can contain characters which are not recognised in UNICODE. The JobScheduler now codes these characters in hexadecimal.

Revision	Date	Note
2.0.192.5394	2007-11-18	**PostgresQL is Better Supported - the JobScheduler Now Tests the Database** On starting, the JobScheduler now carries out simple tests on the database: • whether '\' (backslash) is correctly dealt with, • whether the Latin1 character set and the TB (0x08), CR (0x0D) and NL (0x0A) control keys can be written and read and • whether the `rollback` SQL command works on the `SCHEDULER_VARIABLES` table. The JobScheduler does not start should one of these tests fail. `[ERROR]` `SCHEDULER-451` Database check failed, database is not useable. error `[ERROR]` `SCHEDULER-452` Database does not correctly store test values: (written value) (read value) `[ERROR]` `SCHEDULER-453` Database does not respect rollback command **Remote Task Execution: Spooler.directory, .log_dir, .ini_path and .include_path Return Values for the Remote JobScheduler** • `Spooler.directory` • `Spooler.log_dir` • `Spooler.ini_path` • `Spooler.include_path`

Revision	Date	Note
		JS-195: Job History Corrected for Jobs in Folders `<show_job what="task_history">` now takes account of the complete job path. **JS-196: Entries in the factory.ini File are also Possible for Jobs in Folders** • `factory.ini` (section `[job]`)
2.0.191.5386	2007-11-15	**Deletion of a Job Chain, which is Linked from a Higher Level Chain, No Longer Results in a Warning** When a job chain, which is linked to by a higher level job chain, is deleted, then the higher level chain is made useless. This was the basis of the `SCHEDULER-424 'Job_chain xxx' is being removed before 'yyy'` warning. The message is now given out as information instead of as a warning. **JS-180: Java Source Code Can Now be Translated With a Blank in the Classpath** The JobScheduler sets the `CLASSPATH` environment variable. **JS-146: Jobstarts With New, Changed and Deleted Configuratin Files** • `<config configuration_add_event="…">` • `<config configuration_modify_event="…">` • `<config configuration_delete_event="…">`

Revision	Date	Note
2.0.190.5382	2007-11-12	**JS-181: Change to the Capitalization of a Java Class Written in Source Code** On Windows systems, a change in the capitalization of the name of a Java class lead to an error when the source code of the class was directly specified in `<script language="java">`. The JobScheduler now deletes the generated `.java` file, in order to force Windows to accept the changed capitalization.
2.0.190.5379	2007-11-06	**Obselete <script process_class="..."> is no longer supported** Instead use `<job process_class="…">`.
2.0.190.5375	2007-11-03	**JS-177: Java-Methode Order.priority() Changed** `Order.priority` can now be used under Java.
2.0.190.5370	2007-10-29	**JS-155: Loading and Activation Phases of Dynamic Configuration Seperated** Objects in the dynamic configuration directory are now loaded completely before they are activated. This means that jobs from the configuration directory can now be used for `<config configuration_add_event="…">` etc. **JS-155: Short-term Deletion and Readdition of a File is Seen as a Change** When a file is deleted and then re-added a short time later then the JobScheduler interprets this as the modification of the object.

Revision	Date	Note
2.0.190.5365	2007-10-28	**JS-136: Authentification for SMTP Servers (E-mail Transmission)** The JobScheduler now supports SMTP authentification when sending e-mails. The JobScheduler now only runs with the new `sos.mail.jar` file! • `factory.ini` (section `[smtp]`) **JS-153: New <check_folders> Command** Checks the configuration directories for changes and updates the respective objects in the JobScheduler. On Windows operating systems changes to a directory are noticed immediately, for Unix this could take up to 60 seconds. • `<check_folders>` **JS-149: New<job_chain_node on_error="setback|suspend" Attributes** • `<job_chain_node on_error="…">`

Revision	Date	Note
2.0.189.5354	2007-10-22	**JS-146: Job Starts With New, Modified and Deleted Configuration Files** • `<config configuration_add_event="…">` • `<config configuration_modify_event="…">` • `<config config_configuration_delete_event="…">` **JS-147: Environment Variables Can be Set for Tasks** • `<job><environment>` • `<start_job><environment>` **JS-148: Task and Order Parameters for Non-API Jobs** Non-API jobs are handed over task and order parameters as environment variables. The names are written in block capitals. The prefix "`SCHEDULER_PARAM_`" must be added to the names. Note that order parameters overwrite task parameters of the same name.

Revision	Date	Note
2.0.187.5325	2007-10-10	**JS-134: Scripts for Multiple JobSchedulers** The new `<scheduler_script>` XML element allows scripts to be submitted to more than one JobSchedulers. **JS-68: Multiple <monitor> Job-Monitors** • `<monitor>` **JS-121: Months in <run_time>** • `<month>` **JS-123: absolute_repeat= in < run_time>** • `<period>` **JS-124: Multiple values in day=, monthday= usw.** • `<day>` **New Default Value for <add_order replace='yes'>** • `<add_order replace="yes">` : `replace="yes"` is now the default value. **Dynamic Configuration** • (page 88) **JS-126: HTTP Authentication now Required for Web Services** The `<http.authentication>` authentication is now also required for HTTP `<web_service>`. **JS-129: The EXIT_CODE Database Column now Added Retrospectively** **JS-116: <run_time> for orders in nested job chains** `<run_time>` now correctly repeats orders in nested job chains (`<job_chain_node.job_chain>`).

Revision	Date	Note
2.0.186.5179	2007-08-27	**JS-103: HTTP Authentication Added** The `<http.authentication>` XML element has been added. (Thanks here to Michael Collard, iinet.net.au, for permission to use the algorithm). **Message for delayed heartbeat is now given out as a warning** • `[warn]` `SCHEDULER-827` Own heart beat is late: next_heart_beat has been announced for (this is seconds late) • `[ERROR]` `SCHEDULER-386` Last heart beat was time, seconds ago. Something is delaying JobScheduler execution, the JobScheduler is aborted immediately • `[ERROR]` `SCHEDULER-377` After own late heart beat, JobScheduler member '' has taken exclusiveness

Revision	Date	Note
2.0.185.5170	2007-08-24	**Changes to the Database** • A new index - `SCHEDULER_O_HISTORY_JOB_CHAIN` - is now created for the `JOB_CHAIN` column when the `SCHEDULER_ORDER_HISTORY` table is created. • The former `SCHEDULER_HISTORY_CLUSTMEMBER` index is now created as `SCHEDULER_H_CLUSTER_MEMBER`. • The `SCHEDULER_ORDERS.SPOOLER_ID` database column is now created with `not null`. • The MySQL primary key fields for the `SCHEDULER_ORDERS` is now created with the Latin1 character set.

Revision	Date	Note
2.0.184.5167	2007-08-23	**JS-92: idle_time="5" is now preset** • `<job idle_timeout="never">` **JS-88: New Attribute max_order_history** • `<show_state max_order_history="…">` • `<show_job_chains max_order_history="…">` • `<show_job_chain max_order_history="…">` **JS-97: Messages about files which could not be deleted are now sent as debug3** The following messages are now given out as debug3 instead of info: • `[debug3]` `SCHEDULER-876` Temporary files cannot be deleted. Still trying ..., paths • `[debug3]` `SCHEDULER-877` Temporary files have been deleted now **JS-96: Logs coded in HTML and sent via HTTP can contain the characters '<', '>' and '&' once more.** **JS-93: Message SCHEDULER-846 corrected**

Revision	Date	Note
		• SCHEDULER-846 After task exception and due to stop_on_error='no', the order has been moved to error_state=''

Revision	Date	Note
2.0.183.5153	2007-08-10	**JS-48: In event of an error, non-API-Jobs set an order to the error state** In the event of an error in an order controlled non-API job (`<process>` or `<script language="shell">`), the state of the order being executed is set to the (`<job_chain_node error_state="…">`). This causes the job to stop, (except where `<job stop_on_error="no">` or `<delay_after_error>` are set). **JS-82: Unix: numerous instances of < start_when_directory_changed> no longer block the JobScheduler** On Unix systems where many directories are being monitored with `<start_when_directory_changed>`, the JobScheduler was so occupied that tasks were no longer being processed. This has now been cured. **JS-81: Date header missing in unsent e-mails created with JavaMail** E-mails, which the JobScheduler creates with JavaMail and stores in the mail directory, i.e. which have not been sent, now have a date header.
2.0.182.5146	2007-07-27	**JS-78: Scheduler-Script is now executed before <commands>** Scheduler-Script `<config><script>` is now executed first, before `<config><commands>`. This means that the `<run_time start_time_function="…">` can be calculated before orders are saved in the database. **JS-78: Distributed Orders are now possible with <run_time>** Distributed orders can now use `<run_time>`.

Revision	Date	Note
2.0.181.5143	2007-07-24	**JS-49: For DB2 Databases the numeric(1) data type is used instead of Boolean** **JS-55: <job tasks='0'> no longer starts tasks** In the `pending` state the JobScheduler checks the `<job tasks="…">` attribute. It does not start a task when `tasks="0"` is set.
2.0.180.5139	2007-07-23	**JS-39: scheduler.log.gz is no longer generated** When creation of the `scheduler.log.gz` cannot run in the background, then the JobScheduler does not create it at all, because this delays starting too much. **JS-41: URLs in the HTML Log Window can now be Clicked** URLs in HTML log files which have been transferred by HTTP now function as links. The JobScheduler recognises the `http://` identifier and places the URL in an `<a href="…" target="_blank">...</a>` tag.
2.0.179.5136	2007-07-23	**JS-52: Characters not allowed in XML are replaced by ¿** When creating an XML document, the Job Sche4duler now replaces the characters U-0 to U-1F (with the exception of U-9, U-A, U-D) and U-80 to U-9F with ¿, so that the file remains valid. All other characters of the Latin1 character set (ISO-8859-1) remain supported.

Revision	Date	Note
2.0.178.5132	2007-07-17	**JS-42: Process Jobs now set Job.state_text and Order.state_text** After process jobs (`<process>` and `<script language="shell">` , i.e. non API-Jobs) have ended, the JobScheduler fills `Job.state_text` and `Order.state_text` with the first line of stdout, up to a maximum of 100 characters. **Database indices are now automatically created** The JobScheduler now creates indices for database tables. **Create-tables has now been included in the documentation** • »SQL Instructions Used by the JobScheduler« (page 341) **JS-45: Handling of Backslashes in MySQL** The JobScheduler now takes account of the peculiarities of the backslash in MySQL strings when operating with a MySQL database.

Revision	Date	Note
2.0.177.5128	2007-07-13	**JS-23: Tasks now wait for blocked stdout and stderr temporary files** Only for Windows systems: The JobScheduler delays the end of a Task by half a second, until the temporary files created for `stdin`, `stdout` and `stderr` can be deleted. Note that these files could also be blocked by child processes, which the task process has started and which will continue to run after the task itself has come to an end. Or by a virus scanner holding the files for a moment. **JS-47: <security> in <base> usable** When used at a higher configuration level, `<security>` no longer replaces settings from the `<base>` configuration, but is merged with it. **abort_immediately_and_restart now delayed 2 seconds** `<modify_spooler cmd="abort_immediately_and_restart">` now waits for two seconds after closing the TCP connections, in order to give the Internet Explorer time to close the TCP connections itself, in order to ensure that the connections do not remain blocked. **distributed='no', Cluster-Member-ID now only needs to be unique for a computer name and port** A distributing JobScheduler (`-distributed-orders`) executes jobs locally on other JobSchedulers. `<job_chain distributed="…">` can be set so that a job chain is either executed locally on a single JobScheduler or divided amongst the JobSchedulers in a cluster. Local job chains *must* have a unique name in a cluster, although this is not tested. Up to now, the Cluster-Member-ID for each run of a JobScheduler

Revision	Date	Note
		operating in a cluster had to be unique. This ID is now assembled from the JobScheduler ID, the computer name and the TCP port number. This means that it is important that two computers working in the same cluster do not have the same name.
2.0.176.5116	2007-07-01	**Linux-Scheduler are now compiled under Suse 9.1** The binary distribution files now require Suse version 9.1 or higher (previously 8.0). Up till now, the Linux version was compiled on Suse 8.0. **orders_recoverable='no' no longer suppresses the order history** `<job_chain orders_recoverable="no">`.

Revision	Date	Note
2.0.176.5105	2007-06-21	**<file_order_source> added during run time now works** A <job_chain> with <file_order_source> added during run time did not start automatically. This has now been corrected. **need_db=no is no longer allowed for Cluster Operation** • factory.ini (section [spooler], entry need_db= …) • -exclusive • -distributed-orders • SCHEDULER-357 This is a member of a cluster (option -exclusive or -distributed-orders), and therefore needs role 'scheduler'
2.0.176.5091	2007-06-18	**-distributed-orders and Order.setback()** A distributed order with a start time (for example, set with Order.setback()) could bring the JobScheduler in a loop after a restart. This problem has been solved.

Revision	Date	Note
2.0.176.5087	2007-06-06	**Order Identifiers are tested for Uniqueness in all Nested Job Chains** • `<job_chain_node.job_chain>` Job chains which are nested together form a common *Order_id_space* in order to ensure the uniqueness of an order's identifier when it submitted. The JobScheduler rejects the submission of an order to a job chain, when the order identifier has already been allocated in the Order_id_space. When an order is replaced in a job chain, all other orders in the same Order_id_space but in other job chains will also be replaced. [ERROR] SCHEDULER-426 'job_chain' and 'job_chain' cannot be combined, because of duplicate order id '' [info] SCHEDULER-872 New 'Job_chain' shares order IDs with job_chains [info] SCHEDULER-873 Job_chains job_chains share order IDs because 'Job_chain' has been removed [info] SCHEDULER-874 Order_id_space has been closed

Revision	Date	Note
		[info] SCHEDULER-875 Order_id_space has been closed because 'job_chain' has been removed **orders_recoverable='no' No Longer Suppresses the Order History** When `<job_chain orders_recoverable="no">` is set, it only suppresses the saving of the active orders. **-send-cmd= The -ip-address= is Now Considered** `-send-cmd` considers `<config ip_address="…">`.
2.0.175.5031	2007-05-09	**<add_order at='...'>** The `<add_order at="…">` start time handed over with an order is no longer overwritten by `<add_order><run_time>`. **New Order.log Property** • `Order.log` returns the order log. **New Supervisor_client Class** • `Spooler.supervisor_client` • `Supervisor_client`
2.0.175.5024	2007-05-04	**XML Commands and API for Process Classes** • `Spooler.process_classes` • `Process_classes` • `Process_class` • `Job.process_class` • `<process_class>` • `<process_class.remove>`
2.0.174.5001	2007-04-27	**API for Locks** • `Spooler.locks` • `Locks` • `Lock`

Revision	Date	Note
2.0.172.4983	2007-04-16	**Nested Job Chains** An order can be processed along a number of job chains, one after the other. These job chains are themselves made up into a superordinate job chain. • `<job_chain_node.job_chain>`
2.0.171.4981	2007-04-16	**Multiple Locks and <lock.remove>** A job can set more than one lock and locks can be removed. • `<lock.remove>` • `<job>`_`<lock.use>`
2.0.171.4961	2007-04-09	**Stop Job Chains and Job Chain Nodes** A job chain can be stopped, so that no orders can be processed on this chain. A job chain node can be stopped, so that orders can no longer be processed at a node or so that orders jump over the node. • `<job_chain.modify>` • `<job_chain_node.modify>` • `Job_chain_node.action`
2.0.170.4951	2007-04-03	**<show_calendar>** A new command returns the start times of jobs and orders. • `<show_calendar>`
2.0.169.4940	2007-04-01	**Locks** Locks can be used to serialize the processing of tasks - see chapter (page 158). • `<lock>` • `<job>`_`<lock.use>`
2.0.168.4912	2007-03-19	**The Execution of Tasks on a Remote JobScheduler** • `<process_class remote_scheduler="…">`

Revision	Date	Note
2.0.167.4877	2007-03-08	**Order Log Files are not Always Written in the JobScheduler Log** An error corrected since the 15th February.
2.0.167.4861	2007-02-22	**The <modify_order> XML Command has been Extended** • `<modify_order>` can now change parameters, the Xml_payload and titles. • `<xml_payload>` may now be empty.

Revision	Date	Note
2.0.166.4860	2007-02-19	**The <web_services> XML Element is Now Called <http_server>** • <http_server> • [info] SCHEDULER-847 Please use new name " instead of old " **mail_on_delay_after_error Only Works With <delay_after_error>** factory.ini (section [job], entry mail_on_delay_after_error= …) now only works when <delay_after_error> is set for a job. **Message SCHEDULER-949** • [info] SCHEDULER-949 Job will be executed when a process of process class " becomes available This message now appears in the job history and shows the task ID, once the task has started. **<script> and <base>** The <script> element, which is defined in a <base> configuration, is no longer replaced by a superordinate configuration but is extended. (This does not function with Java because of its syntax.) This means that functions which are defined in a <base> configuration, can be used in a superordinate configuration. **Order Logs When the JobScheduler is Restarted** Log files from orders which have not been completed persist after a restart of the JobScheduler and are continued. To this end they are saved in the database.

Revision	Date	Note
2.0.165.4846	2007-02-12	**Order.state** When `Order.state` is given the value it had before a job was completed (i.e. the state effectively remains unchanged) then this prohibits - as does every other value - the automatic changing to the following state after `spooler_process()` has been completed. Instead, the order repeats the job once more. **HTTP-Server With Alias Directories** • `<http_server><http_directory>` **Job Chain Node Delays** • `<job_chain_node delay="…">` **<show_order> Can Differentiate Between History Entries for an Order** • `<show_order history_id="…">` This means that the HTML interface now shows the correct log when an order has more than one history entry. The command now returns the new `start_time=` attribute (first execution) and `end_time=` (completion of the order). The integrated HTML interface(`http://host:4444/z`) shows the time of completion. **The HTML Interface Now Shows Times More Exactly** The integrated HTML interface (`http://host:4444/z`) now shows seconds up to 99s and in addition quarter-seconds. **<job stop_on_error='no'> sets orders in the error state after an exception occurs**

Revision	Date	Note
		`<job stop_on_error="no">` allocates the error state (`<job_chain_node error_state="…">`) to an order after the task ends with an exception. Documentation change: `stop_on_error="no"` now applies for tasks ending with an exception.
2.0.163.4829	2007-02-01	**The Windows Address-in-use-Problem Has Been Solved** Sub-processes, which the JobScheduler did not know and could not terminate, blocked JobScheduler ports. These ports can no longer be inherited and no longer blocked by subprocesses. In such a situation on Windows, the stdout and stderr temporary files remain. **Temporary files remain as long as subprocesses are running** Windows: Subprocesses, which the JobScheduler did not know and could not terminate, blocked JobScheduler ports. These ports can no longer be inherited and no longer blocked by subprocesses.
2.0.163.4810	2007-01-30	**<file_order_sink move_to="" Accepts Environment Variables** • `<file_order_sink move_to="…">`
2.0.163.4793	2007-01-27	**Script Functions for the Start Time** `<run_time start_time_function="…">` can be used to specify a function in the JobScheduler script (i.e. the <config ><script> script), which returns the next start time.

Revision	Date	Note
2.0.163.4792	2007-01-25	**A Weekday in the Month: < monthdays><weekday>** `<run_time>` has been extended: `<weekday day="…">` can be used to specify particular week days in a month. `<run_time start_time_function="…">` can be used to specify a function in the JobScheduler script which returns the next start time. **Solaris: New libstdc++.so.6.0.8 and libgcc_s.so.1 Modules** As the JobScheduler now supports the Solaris gcc 4.1.1 library, the libstdc++.so.6.0.8 and libgcc_s.so.1 modules are required. A symbolic link must be created: `ln -s libstdc++.so.6.0.8 libstd++.so.6`
2.0.163.4781	2007-01-22	**Order.params Automatically Initialises Itself** `Order.params` automatically initialises itself when first accessed with a `Variable_set`. **Spooler.terminate() With Parameters for Cluster Operation** • `Spooler.terminate()` **Jobs Without <run_time> Take Account of <holidays>** • `<holidays>` **Holiday Administration in an External File: <holidays> with < include>** • `<holidays><include>`

Revision	Date	Note
2.0.163.4758	2007-01-19	**Cluster: Heartbeat Monitoring Delays Adjustable** • `<cluster>` **<file_order_source> Allows Environment Variables** • `<file_order_source directory="…">` **New Setting: mail_on_delay_after_error** • `factory.ini` (section `[spooler]`, entry `mail_on_delay_after_error = …`) • `factory.ini` (section `[job]`, entry `mail_on_delay_after_error = …`) **Options -?, -h and -V** • `-?` shows the possible options • `-h` also shows the possible options • `-V` shows the versions number. Should no other option have been set, then the JobScheduler stops itself immediately. **Java Memory Leak Stopped** A small, but continual memory leak has been sealed. A number of bytes remained in memory, in particular when working with a database (using JDBC). **Memory Leak for stdout/stderr files Stopped** Temporary files were not properly closed on Unix systems, after a task had written to stdout or stderr. This has been corrected.

Revision	Date	Note
2.0.162.4730	2007-01-12	**Cluster: Backup Schedulers and Distributed Orders** • (page 145) • `-exclusive` • `-backup` • `-backup-precedence` • `-install-service` • `-distributed-orders` • `<terminate restart="…">` • `<terminate timeout="…">` • `<terminate continue_exclusive_operation="…">` • `<terminate all_schedulers="…">` • `<job_chain distributed="yes">` **debug9 Logs SQL Commands** The JobScheduler now writes SQL commands in the main log file at `-log-level =debug9` . **Order.xml_payload Accepts ISO-8859-1** `Order.xml_payload` now accepts the ISO-8859-1 character set, when operating with and without a database. **setback() Functions Correctly** Correction of the faulty Version 2.0.160.4598 2006-11-22.

Revision	Date	Note
2.0.161.4619	2006-12-04	**<config ip_address="...">** The IP addresses for TCP and UDP can be specified. • -ip-address • <config ip_address="…"> **Continue Waiting Orders>** <modify_order at="now"> continues an order which was waiting because of Order.run_time or Order.setback() . **Order.suspend()** Orders can be suspended (paused). • Order.suspended • <modify_order suspended="…"> **<copy_params from="order">** <copy_params from="order"> now accepts orders without parameters.
2.0.160.4598	2006-11-22	**Order.setback()** Order.setback() terminates in this version of the JobScheduler (corrected in 2.0.162.4705 from 2007-01-05). **<show_order what="log">** This command now returns the orders executed log from the order history. • <show_order what="log"> **Order.state and Order.run_time** A change to the condition of an order (Order.state) no longer leads to an immediate execution of the order, when a later starting time is specified in its Order.run_time property. • Order.state • Order.run_time • <modify_order state="…"> • <modify_order><run_time>

Revision	Date	Note
2.0.160.4592	2006-10-06	**<show_state what="job_params">** The XML commando can now return the `<params>` of a `<job>`. • `<show_state what="job_params">`
2.0.160.4591	2006-10-03	**An Order in the Database Without Parameters Causes the JobScheduler to Terminate** This error was introduced to the JobScheduler with the file orders (version 2.0.158.4484 2006-09-13) and has now been corrected. **After the 26. September 2006 the JobScheduler Requires a New sos.mail.jar** The `sos.mail.jar` file should not be older than version 1.6.131.4555 (file date 2006-09-25 09:00).
2.0.160.4586	2006-10-30	**<modify_order><run_time>** The `<run_time>` of an order can now be reset for the next cycle. • `<modify_order><run_time>` • `Order.run_time` **<job replace="yes">** A job can now be replaced. • `<job replace="yes">` • `<add_jobs>`

Revision	Date	Note
2.0.159.4578	2006-10-16	**Variable Replacement has been Refined** Single `$` characters are left alone. This character will only be replaced when it is followed by a letter or underscore. This means that dollar characters can be used in Windows shared directories (`"\\HOST\C$"`) and in many regular expressions (`".txt$"`, `"[a-z$]"`, but not `"[$a-z]"`). • Chapter (page 106) **<modify_order setback="no" state="?">** • `<modify_order setback="no">` • `<modify_order state="…">`
2.0.158.4569	2006-10-11	**<show_order what='payload'/>** `<show_order what="payload">` returns the order parameters. **Log files are Deleted Earlier** The new JobScheduler deletes log files earlier, also where the corresponding object (usually an `Order`) is retained. Consider the following situation with a JavaScript job: when the same task gets two successive orders with the same ID, both orders would be held open because the garbage collector has not yet started. The two order objects would only be closed - i.e. the destructors called which delete the log files - after the task itself had ended. This led to an error, because both logs had the same order ID and therefore used the same file.

Revision	Date	Note
2.0.158.4559	2006-10-04	**Changes to the Documentation Files** • The `ersetzung_von_umgebungsvariablen.xml` file has been replaced by `variable_substitution.xml` (page 106). The page contains a directory of all attributes and .ini settings in which environment variables can be used. • The generated index can now be found in `register_data.xml`. The `register.xml` file now contains a text. **<process>-processes are Waited for When Ending the JobScheduler** The JobScheduler did not wait for `<process>` -Tasks on `<terminate>` . It now does this. **Variable_set.substitute** New methods for replacing $-Variables in a string. • `Variable_set.substitute()` **Subprocess.env** The environment variables of a subprocess which is to be started are held in a `Variable_set` . This allows the new `Variable_set.substitute()` method to be used. The subprocess inherits the environment variables from the current process when the `Task.create_subprocess()` is called. • `Subprocess.env`

Revision	Date	Note
2.0.158.4555	2006-09-26	**Environment Variables in < add_order><params> and < start_job><params>** Substitute Environment Variables (page 106) can now be called within `<add_order>` and `<start_job>` in `<param value="…">`.
2.0.158.4551	2006-09-24	**add_pid() for Process Groups** `Task.add_pid()` accepts negative Pids to describe process groups. On Unix systems the complete process group will now be stopped by `kill`. On Windows systems the negative Pid has no particular effect. **<commands> in the Basic Configuration Grouped Together** All `<commands>` in the configuration which are included in `<base>` configurations are executed.

Revision	Date	Note
2.0.158.4533	2006-09-20	**Widths of Table Columns Increased** The JobScheduler checks the width of columns in the `scheduler_orders.id` and `scheduler_order_history.order_id` and if necessary increases them to 255 characters. This allows more room for order identifiers in file orders. Errors which may occur are ignored. **File Orders with too Long Paths** The new database column widths allow the JobScheduler to process paths up to 255 characters long. Longer paths are ignored with a warning. The JobScheduler notes such paths, in order to avoid repeating such error messages. **Mail Transmission to a Directory** The new setting stops transmission and saves the message in a directory specified using `sos.ini` (section `[mail]`, entry `queue_dir= …`). The same applies for `queue_only` in the `sos.ini` file. • `sos.ini` (section `[mail]`, entry `queue_only= …`) • `factory.ini` (section `[spooler]`, entry `mail_queue_only= …`) **start_when_directory_changed Made More Robust** Directory monitoring on Windows systems is renewed after every call of `<start_when_directory_changed>` at the end of a task, without a signal being lost. When a regular expression is specified on Unix systems then `start_when_directory_changed` only starts the job when a file is added, and not when a file is deleted.

Revision	Date	Note
		• < start_when_directory_changed> • Job.start_when_directory_changed()
2.0.158.4505	2006-09-14	**Subprocess.own_process_group** `kill()` can be carried with a negative Pid in order to end the complete process group. • Subprocess.own_process_group • factory.ini (section [spooler], entry subprocess.own_process_group= …)

Revision	Date	Note
2.0.158.4484	2006-09-13	**Double Entries in spooler_task.changed_directories** `Task` could contain duplicate entries if the JobScheduler noted multiple changes in the same directory, befor e a task could be started. Each directory is now only included once. **<process> Returns the Command Line on Process Start** • `<process>` • `[info]` `SCHEDULER-987` Starting process: program and arguments **spooler_log.mail.to etc. Return Default Values** The following calls return default values from the .ini files: • `Mail.from` • `Mail.to` • `Mail.cc` • `Mail.bcc` • `Mail.subject` • `Mail.smtp` • `Mail.queue_dir` **Web_service_operation.peer_ip and .peer_hostname** Two new calls have been added: • `Web_service_operation.peer_ip` • `Web_service_operation.peer_hostname` **File Orders** Two new XML elements can now be specified in `<job_chain>`: • `<file_order_source>` • `<file_order_sink>` • `<show_job_chains>`

Revision	Date	Note
2.0.157.4442	2006-08-30	**<show_state what="job_commands" liefert alle <commands>** Previously the JobScheduler returned just the first `<commands>` element of a job.
2.0.157.4439	2006-08-25	**-send-cmd=... Output Without a Zero Byte** • `-send-cmd` now sends the JobScheduler's answer without the zero byte transferred via TCP. This means that a well-formed XML document is written to stdout.
2.0.157.4436	2006-08-25	**Subprocess.start(String[]) Functions With Java** • `Subprocess.start()` did not work in Java with a string array. This has now been corrected.

Revision	Date	Note
2.0.157.4424	2006-08-18	**<job stop_on_error="no">** • `<job stop_on_error="no">` allows the stopping of a job in the event of an error to be prevented. The JobScheduler then returns either `SCHEDULER-977` or `SCHEDULER-978` message when `-log-level=debug3`. **spooler_task.trigger_files** • `Task.trigger_files` • `<process>`: Hands over the `SCHEDULER_TASK_TRIGGER_FILES` environment variable to a process. **<environment>: Environment Variable Names Originally in Block Capitals** The JobScheduler wrote environment variables in lower case. Now they are written as specified in `<environment>`. This has no effect on Windows systems, where the case is not relevant. **Every Second Day is Omitted for Orders with <period repeat="...">** When the next start time calculated by the JobScheduler using `repeat=` was outside of the current period (i.e. after midnight), then it took as the period for the next start time the next but one period and not the next. This led, in the case of a simple `<run_time period="300">`, to a day being missed out around midnight. This has been corrected. The JobScheduler calculates the next period from the end of the current period.

Revision	Date	Note
2.0.156.4406	2006-08-09	**Signal from a Task which has been Broken Off Leads to a Negative Exit Code for non-<process>-Jobs** Only on Unix Systems: until now the signal from a task which has been broken off has only led to a negative signal code for `<process>` jobs. This now applies to all jobs. **<job ignore_signals="...">** A task ending in a signal (through `kill` or a crash), must no longer lead to the job being stopped. • `<job ignore_signals="SIGTERM SIGKILL ...">` **<commands on_exit_code="SIGTERM">** Only on Unix systems: signal names can now be executed. • `<commands on_exit_code="SIGTERM SIGKILL ...">` • `<commands on_exit_code="signal">`
2.0.155.4396	2006-07-27	**<show_job what="job_commands">** Commands which show the `<job>` element with the `what="job_commands"` attribute now show the job configuration `< commands>` element. • `<show_job what="job_commands">` • `<show_jobs what="job_commands">` • `<show_job_chains what="job_commands">` • `<show_state what="job_commands">`

Revision	Date	Note
2.0.154.4393	2006-07-27	**Database Calls from the scheduler_orders Table** The primary key of the `SCHEDULER_ORDERS` table has been extended to include `SPOOLER_ID`. `SPOOLER_ID` und `JOB_CHAIN` have been added to all "where" clauses for this table. The `RUN_TIME` Clob is now only set if `run_time` has been filled.
2.0.154.4390	2006-06-30	**Java Jobs without a Constructor** A Java job without a parameter free constructor leads to the process being broken off instead of the `java.lang.NoSuchMethodError` exception. **<config java_class_path="">** The `java_class_path` attribute was not evaluated when process classes are in use. • `<config java_class_path="…">`
2.0.154.4379	2006-06-20	**history_on_process now behaves as documented** A wrong entry is no longer ignored, but leads to an error. • `factory.ini` (section `[job]`, entry `history_on_process= …`)

Revision	Date	Note
2.0.154.4366	2006-06-19	**A Number of Improvements in the Internal Event Handling** The new `wait_until` attribute returns the time of the next planned action. After circa 20 Minutes idle time the JobScheduler sends out a notification of what it is waiting for and when this should occur. • `<state wait_until="…">` • `SCHEDULER-972` **<show_state> Error Returned on Ending a Subprocess which could Cause No E-mail** `<show_state>` and other commands which showed the state of task returned an error when a subprocess or the task process had ended itself and the JobScheduler had not registered this. The reason: the system call for reading the priority returned an error. In particular, when `<kill_task>` was used with subprocesses, this could lead to the e-mail at the end of a task not being sent and a `SCHEDULER-302` error being given out instead. • `<kill_task>` • `<subprocess priority="…">` • `Log.mail_on_error`

Revision	Date	Note
2.0.154.4352	2006-06-16	**<process param="">: Task Parameters are Replaced** The JobScheduler now replaces the task parameters in the `param=` attribute. These have priority over environment variables. This means that a different parameter row can be handed over for each task. • `<process param="…">` **New Property Order.at** • `Order.at` **Strg-C No Longer Stops Processes** Strg-C is now only effective on the JobScheduler process, and no longer on the task processes (with the exception of `<process>` and `<script>` on Windows systems). This allows the JobScheduler to properly end tasks after Strg-C. Note however that a second Strg-C still leads to an immediate end of all tasks and the JobScheduler.

Revision	Date	Note
2.0.154.4339	2006-06-06	**-env= corrected (Windows)** `-env=` had set variables for child processes, but not for its own. • `-env` **<add_order replace="yes">** The XML-Schema now allows replace="yes". • `<add_order replace="…">` **New Option -expand-classpath=** Expands a Java class path given as parameter, i.e. jokers are processed as in `sos.ini` (section `[java]`, entry `class_path= …`). • `-expand-classpath` **<commands on_exit_code="0">** • A new property `Task.exit_code` (int), initially 0, has been created. `Log.error()`, `Task.error()` or a task exception now set `Log.exit_code`=1. `exit_code` can also now be set by the job itself, but is overwritten under the conditions listed above. This means that every task possible in the JobScheduler will now return an `exit_code`. • The `exit_code` is now written in the `scheduler_history` database table in the new `exit_code` column, which is automatically created by the JobScheduler. • On Unix systems a process which has been broken off now returns the negative signal code number as the Exit-Code. • `<job>` has been extended with `<commands on_exit_code="…">`. A series of `"success"` (which is the same as `0`) or `"error"` exit codes can now be given for the `on_exit_code` attribute. The `"error"` setting is valid for all Exit-Codes with the exception

Revision	Date	Note
		of 0, which are not executed in other `<commands on_exit_code="…">`. This means that an exit code can no longer be allocated to more than one `<commands>`. • An error occurring when a command is being executed now stops the execution of subsequent commands and is logged for the task. The job is then stopped. • `<commands>` can now be empty. **Reasons for Starting a Task 'queue' and 'queue_at'** These reasons are now logged (with `SCHEDULER-930`). **Memory Leaks when Calling Spidermonkey Closed** Applies to `spidermonkey.dll` and `libspidermmonkey.so`. **<copy_params>** `<params>` receives the new `<copy_params from="…">` child element. This can only be used in under `<job><commands>` in `<start_job><params>` and `<add_order><params>` and copies the task or order parameters in its place. `<copy_params from="order">` is an error, when there is no order. **Orders with Start Time** Orders can now have a start time which is saved in the database. Such start times are thereby retained after the JobScheduler has been restarted. An order which has not been executed but whose start time has passed will be started immediately (as long as the job `<run_time>` allows this). This corresponds with the behavior of `<start_job at="…">`.

Revision	Date	Note
		• `<add_order at="…">` **Relative Times with 'now +HH:MM'** • `<start_job at="now+...">` • `<add_order at="now+...">` Possible values are: • `now` • `now +` *seconds* • `now +` *hh:mm* • `now +` *hh:mm:ss* • `yyyy-mm-dd HH:MM` • `yyyy-mm-dd HH:MM:SS` **language="perl" On Windows** On Windows systems the JobScheduler only gave the following instruction: `Win32::OLE->Option( Warn => 3 );` for `<script language="perlscript">`. `<script language="perl">` now has the same behavior. This means that an error occuring when calling a JobScheduler method is no longer ignored by Perl.
2.0.153.4313	2006-05-29	**Jokers in the Java Class Path** `class_path = c:\directory\sos.*.jar;...` • `sos.ini` (section `[java]`, entry `class_path= …`)

Revision	Date	Note
2.0.151.4305	2006-05-26	**Environment Variables With ${name}** Where previously environment variables could be called with `$` *name*, they can now also be called with `${` *name* `}`. This makes separation of a variable from any following text clearer. **Environment Variables can be used in <base file=""> and < include file="">** • `<base file="…">` • `<include file="…">` **Correction to spooler_task.params** The call could cause the JobScheduler to crash. • `Variable_set.names`

Revision	Date	Note
2.0.151.4304	2006-05-26	**<job min_tasks=...>** The JobScheduler can now ensure that a minimum number of tasks are kept running. • `<job min_tasks="…">` **Subprocess.wait_for_termination(int) now Effective on Unix Systems** The waiting time parameter was ignored on Unix systems. This has been corrected. • `Subprocess.wait_for_termination()` **Perl Interface on Unix Systems Extended by Object and Array Parameters** The Perl interface accepts the following calls on Unix systems: • `Spooler.add_job_chain()` • `Job_chain.add_order()` • `Order.params` • `Task.create_subprocess()` • `Subprocess.start()` Affects the `libsosperlscript.so` file. **New Order with Start Time Wakes Sleeping Tasks** An order with a start time now wakes a sleeping task (in the running_waiting_for_order state), so that the order will be immediately executed at its start time.

Revision	Date	Note
2.0.151.4254	2006-04-13	**Neue Option -env=NAME=WERT** • -env =NAME=VALUE • -sos.ini sets the SOS_INI environment variable. **Behavior Changed on <start_when_directory_changed> Error** Up till now directory monitoring was repeated at the start of every task. In the event of an error in the directory monitoring (for example, because the directory no longer existed) the job was stopped, although the task could be completed, and no further task could be started. Now the JobScheduler no longer stops the job directly. Directory monitoring is repeated at the end of the task (after spooler_exit()). An error then causes the job to be stopped - but in a way which can be caught by <delay_after_error> . • <start_when_directory_changed> **Web Service Parameters can now be Included in XSLT Style Sheets** Parameters containing apostrophes and inverted commas are rejected with a warning, as such values cannot be handed over to libxslt. • <web_service><params>
2.0.151.4253	2006-04-12	**New Property: Order.params** • Order.params **Integration of Web Service Parameters into XSLT Style Sheets** Parameters containing apostrophes and inverted commas are rejected with a warning, as such values cannot be handed over to libxslt. • <web_service><params>

Revision	Date	Note
2.0.150.4240	2006-04-06	**Schema Allows < delay_after_error delay="stop"> once more** • `<delay_after_error delay="stop">`
2.0.150.4233	2006-04-04	**command_line.xml Restructured** scheduler_client has been made obsolete. The options have been integrated in the JobScheduler. The different calls of the JobScheduler have now been documented, see, for example "Handing over a Job to Running JobScheduler".
2.0.150.4225	2006-03-31	**scheduler.xsd Restricted to ISO-8859-1** XML Commands can only contain ISO-8859-1 characters. **Process Priorities** • `<job priority="below_normal">` • `Task.priority` • `Task.priority_class` • `Subprocess.priority` • `Subprocess.priority_class`

Revision	Date	Note
2.0.149.4218	2006-03-25	**scheduler_client** The new `scheduler_client` Program can Forward Jobs and Orders to the JobScheduler. The start of a job, whose script is entered by way of `stdin` is written so: Forwarding an Order: **<job temporary="true">** A temporary job is now deleted when the start time is no longer held in `<run_time>` and the task queue is empty. **<script language="shell">** The new "shell" script language allows shell jobs to be specified in the XML configuration. (for the scheduler_client) • `<script language="shell">` **New Parameter for <run_time>: <at>** • `<at at="2006-03-24 12:00">`
2.0.148.4199	2006-03-17	**The JobScheduler Runs on Windows 2000 Once More** A work around for the defective SafeArrayGetVartype() routine in Windows 2000 has been developed.

Revision	Date	Note
2.0.148.4178	2006-03-10	**<web_service> With <params>** A Web Service accepts parameters which are read in a job. • `<web_service>` • `Web_service.params` **Messages Translated into English and Given Codes** The messages (with the exception of some debug messages) are now coded - for example " `SCHEDULER-900 JobScheduler is starting ...`". • List of Message Codes (page 234) **New Properties: Request|Response.content_type and .charset_name** • `Web_service_request.content_type` • `Web_service_request.charset_name` • `Web_service_response.content_type` • `Web_service_response.charset_name`
2.0.148.4155	2006-03-05	**<job_chain orders_recoverable="no">** • `<job_chain orders_recoverable="no">` • `Job_chain.orders_recoverable` **<script java_class="–"> can be overwritten** • `<script java_class="…">`

Revision	Date	Note
2.0.148.4126	2006-03-01	**New Windows C++-Compiler** After this revision (#4126) all Windows software is compiled with Microsoft Visual Studio 2005 C++. This should not lead to any changes. **XML Schema Introduced** The XML configuration and commands are validated against a built-in XML schema. The `-show-xml-schema` option shows this schema. A current version is available on the internet for test purposes: http://www.sos-berlin.com/repository/scheduler/current/scheduler.xsd. New Command Line Options: • `-show-xml-schema` • `-validate-xml` **New Property: order.xml_payload** `Order.xml_payload` can take on an XML document in addition to payload. A new property and a new XML element: • `Order.xml_payload` • `<xml_payload>` **<web_service job_chain=">** A web service can be implemented with a job chain. New XML elements, new attributes: • `<web_service job_chain="…">` • `<web_service timeout="…">` New classes and methods: • `Order.web_service_operation` • `Web_service_operation` • `Web_service_request` • `Web_service_response`

Revision	Date	Note
2.0.147.4100	2006-02-21	**jdbc Files Delete Temp Files on Windows Systems** Temporary files were not deleted when writing a LOB because FileInputStream.close() was not called. This was not so for Oracle. This has been corrected.
2.0.147.4099	2006-02-20	**HTTP-Server Corrected**
2.0.147.4091	2006-02-09	**Variable_set.names** `Variable_set.names` returns the variable names, separated by semicolons.
2.0.146.4073	2006-02-02	**factory.ini queue_dir=** queue_dir can be set in the factory.ini file once more. Affects the JobScheduler since version 2.0.139.3889 (2005-09-15). The best place for this setting remains the `sos.ini` (section `[mail]`, entry `queue_dir= …`) file.
2.0.146.4061	2006-01-30	**Web Services** New XML elements, new attributes: • `<web_services>` • `<web_service>` • `<add_order web_service="…">` • `<start_job web_service="…">` • `<order web_service="…">` • `<payload>` • `<task web_service="…">` • `<service_request>` • `<content>` • `<job_chain visible="no">` • `<job visible="no">` New classes and methods: • `Web_service` • `Order.xml` • `Order.web_service` • `Task.web_service`
2.0.145.4049	2006-01-15	**Job_chain.remove()** New commands `Job_chain.remove()` and `<remove_job_chain>`.

Revision	Date	Note
2.0.144.4038	2005-12-15	**Order.setback() and Job.set_max_order_setbacks() are now Available in Java** See `Order.setback()` and `Job.max_order_setbacks`.
2.0.144.4036	2005-12-08	**E-mail Settings in the factory.ini [job ...] Accepted Once More** When a task ends with an error or warning, then the JobScheduler overwrites the setting from `log_mail_subject`. This can be changed by the e-mail style sheet.
2.0.144.4026	2005-11-11	**Orders with <run_time> can now Wake a Task Waiting on an Order** Up to now the JobScheduler had ignored an order repeated through `run_time`, when a task was already waiting on the order (i.e. in the `running_waiting_for_order` state).

Revision	Date	Note
2.0.144.4015	2005-11-08	**SQL Table Names without ""; Large order.payload Now Possible** In the JobScheduler SQL commands table names are written in upper case without inverted commas. (Exception: queries coming from SQL files). The `Order.payload` is written as a CLOB by the hostware mechanism which allows it to be of any size. This is good for the `Variable_set`. **HPUX: Hostjava Integrated in the JobScheduler** Because of a problem with gcc 3.2 when loading the libhostjava.sl file (the static variables were not initialised), Hostjava is now integrated into HP-UX. system_information() already implements the disc space functions for HP-UX. **scheduler_variables Table Changed** The "value" column is now an integer. The new "textvalue" column is a Varchar(250) (is not used by the JobScheduler at the moment).
2.0.143.4005	2005-11-07	**<security ignore_unknown_hosts="yes">** `<security ignore_unknown_hosts="yes">` functions once more.
2.0.142.4003	2005-11-03	**<job java_options=>** See `<job java_options="…">` (page 42). In addition, `<config java_options="…">` functions as well, when a job runs in its own process.

Revision	Date	Note
2.0.140.3999	2005-10-31	**Job-<script> can be overwritten** See `<script>` (page 77). Attributes (`language=` etc.) cannot be changed. Only script code with `<include>` is changeable. The next task then uses the new code.
2.0.140.3995	2005-10-30	**Monitor for Job Control** See `<monitor>` (page 56). The current spidermonkey.dll or libspidermonkey.so must be used when monitoring in JavaScript. **New XML Element: <job_chains>** See `<job_chains>` (page 53). **New XML Element: <start_when_directory_changed>** See `<start_when_directory_changed>` (page 81). **New XML Element: <delay_after_error>** See `<delay_after_error>` (page 32). **New XML Element: <delay_order_after_setback>** See `<delay_order_after_setback>` (page 32).
2.0.140.3943	2005-09-25	**Order Repeating Because of <run_time> Creates its Own History Entry** After an order has reached the end state and its `Order.run_time` plans another termin, then: • the JobScheduler now closes the order log file, • writes an entry in the order history - see `factory.ini` (section `[spooler]`, entry `db_order_history_table= …`) (page 99), • and starts a new order log.

Revision	Date	Note
2.0.140.3939	2005-09-24	**Windows: Double Ctrl-C Ends All Processes** The first Ctrl-C ends the JobScheduler in the same way as `<modify_spooler cmd="terminate">` . The second then stops all processes. The third Ctrl-C lets the operating system stop the JobScheduler. **Termination With a Time Limit** The JobScheduler can now be stopped with a time limit. When the tasks have not stopped within the limit specified, then the JobScheduler stops the task processes. It then stops itself after waiting on the tasks at the most for a further 30 seconds. See `<modify_spooler cmd="terminate">` (page 195) and `Spooler.terminate()` .
2.0.139.3925	2005-09-21	**Incorrect <include> in the <description> of a Job Leads to a Warning in the Job Protokoll** (Until now errors were only written in the main log file.) See `<description>` (page 33).
2.0.139.3889	2005-09-15	**New Treatment of the Default Settings for Sending E-mails**
2.0.138.3866	2005-09-07	**spooler_log.debug1() to spooler_log.debug9() no Longer Suppressed in JobScheduler Script** **Database Tables Created With varchar Instead of char**

Revision	Date	Note
2.0.138.3851	2005-09-06	**Option -pid-file Taken Over as a Windows Service on Installation** See `-pid-file` **XSLT E-mail Contains <order> Element** The `<order>` element is handed over to the `<task>` element after a task has been ended.
2.0.137.3844	2005-08-24	**-sos.ini= Accepts Spaces in File Names**
2.0.137.3839	2005-08-21	**New Job.remove() Method, New XML Command <modify_job cmd="remove">** See `Job.remove()` and `<modify_job cmd="remove">` (page 193). **<remove_order> Carried Out Immediately in Database** **E-mails now have Correct Message Text**
2.0.136.3836	2005-08-12	**New Task.stdout_path and Task.stderr_path Methods** See `Task.stdout_path` and `Task.stderr_path`. **<remove_order> Carried Out Immediately in Database** **E-mails now have Correct Message Text**
2.0.136.3834	2005-08-12	**Error on SQL Insert for a New Task No Longer Leads to an E-mail Flood**
2.0.134.3832	2005-08-12	**Change in a Order <run_time> Only Effective at Beginning of Job Chain** A change to the <run_time> of an order is only effective when the order is at the start of a job chain and has not been carried out. Otherwise the effect of the change delayed until the order has reached the end state.

Revision	Date	Note
2.0.135.3824	2005-08-10	**<modify_order> Command Can Change <run_time>** `<run_time>` can now be set in `<modify_order>`. **New Job_chain.add_or_replace_order() Method** `Job_chain.add_or_replace_order()` functions in the same way as `Job_chain.add_order()`, when the job chain no order with the same name already active. Should an order with the same name already be in the job chain, then it will be replaced. More exactly: it will be removed from the job chain (as with `Order.remove_from_job_chain()`), and the new order then added. Should the order to be replaced be carried out by a task, then the JobScheduler will wait until this has been finished before the new order can be carried out. The order which has been replaced appears in the XML response as `<order replaced="yes">`. Note that this only applies within `<task>`, (and not `<order_queue>`), as the order is no longer in the order queue. The new, replacing order appears in the XML response as `<order replacement="yes">`. See also `<add_order replace="yes">` (page 13). **Change made to Order.remove_from_job_chain() Method** (`Order.remove_from_job_chain()` has the same effect as `<remove_order>`.) Should the order to be removed be being carried out by a task, then the property `Order.job_chain` will still

Revision	Date	Note
		return the job chain out of which the order will be removed from. It is only after the execution has been completed, that the property will return the property `null` (as long as the order has not been re-added to a job chain). This ensures that the `job_chain` property remains stable whilst a task is being carried out. After being removed - and not carried out - the order appears in the XML response as `<order removed="yes">`. Note that this only applies within `<task>`, (and not `<order_queue>`), as the order is no longer in the order queue. **sos.spooler.id() Java Method also Returns Integers as Strings** `.toString()` is now called instead of casting `(String)`, so that an order identifier which has been saved as an integer variant (or similar) is returned in Java as a string. (Up till now this caused a ClassCastException.) **let_run_terminate_and_restart Ends Tasks in the running_waiting_for_order State** See `<modify_spooler cmd="let_run_terminate_and_restart">` (page 195).
2.0.134.3818	2005-08-04	**The sos.spooler.Mail Class Functions on Unix/Java** The sos.spooler.Mail class could not be used on Unix in Java (Error code COM-80020009 DISP_E_EXCEPTION).
2.0.134.3817	2005-08-03	**<period single_start> Order Corrected** The next Single_start of an order with `<period>` will now be correctly carried out.

Revision	Date	Note
2.0.134.3814	2005-08-01	**Subprocess.termination_signal** `Subprocess.termination_signal` returns 0 on Unix systems or the number of the signal with which the subprocess ended. This method returns 0 on Windows systems.
2.0.134.3812	2005-07-30	**delay_after_error Creates a New Task** A task for which `Job.delay_after_error` is set, is now re-added by the JobScheduler, with an new identifier, to the task order queue. The task parameters are taken over once more. This does not apply to order controlled jobs. No new task will be started for these jobs, as such jobs are started by a waiting order. **New delayed_after_error_task Attribute in Answer XML Element <task>** See `<task delayed_after_error_task="…">` (page 230). **New delay_after_error= wnd in_period= Attributes in Answer XML Element <job>** See `<job delay_after_error="…">` (page 42) and `<job in_period="…">` (page 42).
2.0.133.3786	2005-07-04	**XSLT Style Sheets for E-mails** The `Mail.xslt_stylesheet_path` sets the XSLT style sheet for e-mails. **New Xslt_stylesheet Class** `Spooler.create_xslt_stylesheet()` returns a `Xslt_stylesheet`.
2.0.132.3743	2005-06-25	**<order_queue> returns the next_start_time attribute**

Revision	Date	Note
2.0.130.3742	2005-06-24	**Should <run_time> be missing, then this will be interpreted as if an empty <run_time/> had been given**
2.0.129.3696	2005-06-08	**Order job is stopped, should false be returned in spooler_init() oder spooler_open()** An order job, that returns `false` in `spooler_init()` or `spooler_open()`, will be stopped with an error message. This can be prevented using `Job.delay_after_error`. The order will then remain in the job order queue and will be processed after the next successful start of the job.
2.1.2.	2010-08-11	**JS-540: Added script attributes for Java API jobs** The script element do have one more attribute 'java_class_path' (see documentation). The current values for Java Classpath and Java VM Options will be logged in the task log.
2.1.2.	2010-07-29	**JS-550: on_exit_code 'error' will be correct evaluated** If the attribute on_exit_code is 'error', then commands with on_exit_code = 'error' will be executed only if the current exit code is not 0 (null). Until now the commands are executed every time.

Appendix C: SQL Instructions Used by the JobScheduler

The instructions the JobScheduler uses to create any missing database tables are listed below. Note that the instructions used depend on the database software being used.

C.1 SQL Instructions Used by the JobScheduler for DB2®

```
CREATE TABLE SCHEDULER_VARIABLES
(
    "NAME"     varchar(100) not null,
    "WERT"     integer,
    "TEXTWERT" varchar(250),
    primary key ( "NAME" )
);

CREATE TABLE SCHEDULER_TASKS
(
    "TASK_ID"           integer not null,
    "SPOOLER_ID"        varchar(100) not null,
    "CLUSTER_MEMBER_ID" varchar(100),
    "JOB_NAME"          varchar(255) not null,
    "ENQUEUE_TIME"      timestamp,
    "START_AT_TIME"     timestamp,
    "PARAMETERS"        clob,
    "TASK_XML"          clob,
    primary key( "TASK_ID" )
);

CREATE TABLE SCHEDULER_HISTORY
(
    "ID"                integer not null,
    "SPOOLER_ID"        varchar(100) not null,
    "CLUSTER_MEMBER_ID" varchar(100),
    "JOB_NAME"          varchar(255) not null,
    "START_TIME"        timestamp not null,
    "END_TIME"          timestamp,
    "CAUSE"             varchar(50),
    "STEPS"             integer,
    "EXIT_CODE"         integer,
    "ERROR"             numeric(1),
    "ERROR_CODE"        varchar(50),
    "ERROR_TEXT"        varchar(250),
    "PARAMETERS"        clob,
    "LOG"               blob,
    primary key( "ID" )
);

CREATE INDEX SCHEDULER_HIST_1 on SCHEDULER_HISTORY ( "START_TIME" );
CREATE INDEX SCHEDULER_HIST_2 on SCHEDULER_HISTORY ( "SPOOLER_ID" );
CREATE INDEX SCHEDULER_HIST_3 on SCHEDULER_HISTORY ( "JOB_NAME" );
CREATE INDEX SCHEDULER_HIST_4 on SCHEDULER_HISTORY ( "CLUSTER_MEMBER_ID" );

CREATE TABLE SCHEDULER_ORDERS
(
    "JOB_CHAIN"                    varchar(255) not null,
    "ID"                           varchar(255) not null,
    "SPOOLER_ID"                   varchar(100) not null,
    "DISTRIBUTED_NEXT_TIME"        timestamp,
    "OCCUPYING_CLUSTER_MEMBER_ID"  varchar(100),
    "PRIORITY"                     integer not null,
    "STATE"                        varchar(100),
    "STATE_TEXT"                   varchar(100),
    "TITLE"                        varchar(200),
    "CREATED_TIME"                 timestamp not null,
    "MOD_TIME"                     timestamp,
    "ORDERING"                     integer not null,
    "PAYLOAD"                      clob,
    "INITIAL_STATE"                varchar(100),
    "RUN_TIME"                     clob,
    "ORDER_XML"                    clob,
    primary key( "SPOOLER_ID", "JOB_CHAIN", "ID" )
);

CREATE TABLE SCHEDULER_ORDER_HISTORY
(
    "HISTORY_ID" integer not null,
    "JOB_CHAIN"  varchar(255) not null,
    "ORDER_ID"   varchar(255) not null,
    "SPOOLER_ID" varchar(100) not null,
    "TITLE"      varchar(200),
    "STATE"      varchar(100),
    "STATE_TEXT" varchar(100),
    "START_TIME" timestamp not null,
    "END_TIME"   timestamp not null,
    "LOG"        blob,
    primary key( "HISTORY_ID" )
);

CREATE INDEX SCHED_O_HIST_1 on SCHEDULER_ORDER_HISTORY ( "SPOOLER_ID" );
CREATE INDEX SCHED_O_HIST_2 on SCHEDULER_ORDER_HISTORY ( "JOB_CHAIN" );
CREATE INDEX SCHED_O_HIST_3 on SCHEDULER_ORDER_HISTORY ( "START_TIME" );

CREATE TABLE SCHEDULER_ORDER_STEP_HISTORY
(
  "HISTORY_ID"          numeric(9)     not null,
  "STEP"                numeric(9)     not null,
  "TASK_ID"             numeric(9)     not null,
  "STATE"               varchar(100)           ,
  "START_TIME"          timestamp      not null,
  "END_TIME"            timestamp              ,
  primary key ( "HISTORY_ID","STEP" )
);

CREATE TABLE SCHEDULER_CLUSTERS
(
    "MEMBER_ID"              varchar(100) not null,
    "SCHEDULER_ID"           varchar(100) not null,
    "PRECEDENCE"             integer,
    "LAST_HEART_BEAT"        integer,
    "NEXT_HEART_BEAT"        integer,
    "ACTIVE"                 numeric(1),
    "EXCLUSIVE"                 numeric(1),
    "DEAD"                      numeric(1),
    "COMMAND"                   varchar(250),
    "HTTP_URL"                  varchar(100),
    "DEACTIVATING_MEMBER_ID" varchar(100),
    "XML"                       clob,
    primary key( "MEMBER_ID" )
);
```

C.2 SQL Instructions Used by the JobScheduler for Microsoft SQL Server

```
CREATE TABLE SCHEDULER_VARIABLES
(
    "NAME"     varchar(100) not null,
    "WERT"     integer,
    "TEXTWERT" varchar(250),
    primary key ( "NAME" )
);

CREATE TABLE SCHEDULER_TASKS
(
    "TASK_ID"           integer not null,
    "SPOOLER_ID"        varchar(100) not null,
    "CLUSTER_MEMBER_ID" varchar(100),
    "JOB_NAME"          varchar(255) not null,
    "ENQUEUE_TIME"      datetime,
    "START_AT_TIME"     datetime,
    "PARAMETERS"        ntext,
    "TASK_XML"          ntext,
    primary key( "TASK_ID" )
);

CREATE TABLE SCHEDULER_HISTORY
(
    "ID"                integer not null,
    "SPOOLER_ID"        varchar(100) not null,
    "CLUSTER_MEMBER_ID" varchar(100),
    "JOB_NAME"          varchar(255) not null,
    "START_TIME"        datetime not null,
    "END_TIME"          datetime,
    "CAUSE"             varchar(50),
    "STEPS"             integer,
    "EXIT_CODE"         integer,
    "ERROR"             bit,
    "ERROR_CODE"        varchar(50),
    "ERROR_TEXT"        varchar(250),
    "PARAMETERS"        ntext,
    "LOG"               image,
    primary key( "ID" )
);

CREATE INDEX SCHEDULER_HISTORY_START_TIME on SCHEDULER_HISTORY ( "START_TIME" );
CREATE INDEX SCHEDULER_HISTORY_SPOOLER_ID on SCHEDULER_HISTORY ( "SPOOLER_ID" );
CREATE INDEX SCHEDULER_HISTORY_JOB_NAME   on SCHEDULER_HISTORY ( "JOB_NAME" );
CREATE INDEX SCHEDULER_H_CLUSTER_MEMBER   on SCHEDULER_HISTORY (
"CLUSTER_MEMBER_ID");

CREATE TABLE SCHEDULER_ORDERS
(
    "JOB_CHAIN"                    varchar(255) not null,
    "ID"                           varchar(255) not null,
    "SPOOLER_ID"                   varchar(100) not null,
    "DISTRIBUTED_NEXT_TIME"        datetime,
    "OCCUPYING_CLUSTER_MEMBER_ID"  varchar(100),
    "PRIORITY"                     integer not null,
    "STATE"                        varchar(100),
    "STATE_TEXT"                   varchar(100),
    "TITLE"                        varchar(200),
    "CREATED_TIME"                 datetime not null,
    "MOD_TIME"                     datetime,
    "ORDERING"                     integer not null,
    "PAYLOAD"                      ntext,
    "INITIAL_STATE"                varchar(100),
    "RUN_TIME"                     ntext,
    "ORDER_XML"                    ntext,
    primary key( "SPOOLER_ID", "JOB_CHAIN", "ID" );
);

CREATE TABLE SCHEDULER_ORDER_HISTORY
(
    "HISTORY_ID" integer not null,
    "JOB_CHAIN"  varchar(255) not null,
    "ORDER_ID"   varchar(255) not null,
    "SPOOLER_ID" varchar(100) not null,
    "TITLE"      varchar(200),
    "STATE"      varchar(100),
    "STATE_TEXT" varchar(100),
    "START_TIME" datetime not null,
    "END_TIME"   datetime not null,
    "LOG"        image,
    primary key( "HISTORY_ID" );
);

CREATE INDEX SCHEDULER_O_HISTORY_SPOOLER_ID on SCHEDULER_ORDER_HISTORY ( "SPOOLER_ID"
 );
CREATE INDEX SCHEDULER_O_HISTORY_JOB_CHAIN  on SCHEDULER_ORDER_HISTORY ( "JOB_CHAIN"
);
CREATE INDEX SCHEDULER_O_HISTORY_START_TIME on SCHEDULER_ORDER_HISTORY ( "START_TIME"
 );

CREATE TABLE SCHEDULER_ORDER_STEP_HISTORY
(
  "HISTORY_ID"          numeric(9)     not null,
  "STEP"                numeric(9)     not null,
  "TASK_ID"             numeric(9)     not null,
  "STATE"               varchar(100)           ,
  "START_TIME"          datetime       not null,
  "END_TIME"            datetime               ,
  primary key ( "HISTORY_ID","STEP" )
);
```

```
CREATE TABLE SCHEDULER_CLUSTERS
(
    "MEMBER_ID"                varchar(100) not null,
    "SCHEDULER_ID"             varchar(100) not null,
    "PRECEDENCE"               integer,
    "LAST_HEART_BEAT"          integer,
    "NEXT_HEART_BEAT"          integer,
    "ACTIVE"                   numeric(1),
    "EXCLUSIVE"                numeric(1),
    "DEAD"                     numeric(1),
    "COMMAND"                  varchar(250),
    "HTTP_URL"                 varchar(100),
    "DEACTIVATING_MEMBER_ID"   varchar(100),
    "XML"                      ntext,
    primary key( "MEMBER_ID" )
);
```

C.3 SQL Instructions Used by the JobScheduler for MySQL®

```
CREATE TABLE SCHEDULER_VARIABLES
(
    "NAME"       varchar(100) not null,
    "WERT"       integer,
    "TEXTWERT"   varchar(250),
    primary key ( "NAME" )
)
Type=InnoDB;

CREATE TABLE SCHEDULER_TASKS
(
    "TASK_ID"            integer not null,
    "SPOOLER_ID"         varchar(100) not null,
    "CLUSTER_MEMBER_ID"  varchar(100),
    "JOB_NAME"           varchar(255) not null,
    "ENQUEUE_TIME"       datetime,
    "START_AT_TIME"      datetime,
    "PARAMETERS"         longtext,
    "TASK_XML"           longtext,
    primary key( "TASK_ID" )
)
Type=InnoDB;

CREATE TABLE SCHEDULER_HISTORY
(
    "ID"                 integer not null,
    "SPOOLER_ID"         varchar(100) not null,
    "CLUSTER_MEMBER_ID"  varchar(100),
    "JOB_NAME"           varchar(255) not null,
    "START_TIME"         datetime not null,
    "END_TIME"           datetime,
    "CAUSE"              varchar(50),
    "STEPS"              integer,
    "EXIT_CODE"          integer,
    "ERROR"              bool,
    "ERROR_CODE"        varchar(50),
    "ERROR_TEXT"        varchar(250),
    "PARAMETERS"        longtext,
    "LOG"               longblob,
    primary key( "ID" )
)
Type=InnoDB;

CREATE INDEX SCHEDULER_HISTORY_START_TIME on SCHEDULER_HISTORY ("START_TIME"       ):
CREATE INDEX SCHEDULER_HISTORY_SPOOLER_ID on SCHEDULER_HISTORY ("SPOOLER_ID"       ):
CREATE INDEX SCHEDULER_HISTORY_JOB_NAME   on SCHEDULER_HISTORY ("JOB_NAME"         ):
CREATE INDEX SCHEDULER_H_CLUSTER_MEMBER   on SCHEDULER_HISTORY ("CLUSTER_MEMBER_ID");

CREATE TABLE SCHEDULER_ORDERS
(
    "JOB_CHAIN"                  varchar(255)  character set latin1  not null,
    "ID"                         varchar(255)  character set latin1  not null,
    "SPOOLER_ID"                 varchar(100)  character set latin1  not null,
    "DISTRIBUTED_NEXT_TIME"      datetime,
    "OCCUPYING_CLUSTER_MEMBER_ID" varchar(100),
    "PRIORITY"                   integer not null,
    "STATE"                      varchar(100),
    "STATE_TEXT"                 varchar(100),
    "TITLE"                      varchar(200),
    "CREATED_TIME"               datetime not null,
    "MOD_TIME"                   datetime,
    "ORDERING"                   integer not null,
    "PAYLOAD"                    longtext,
    "INITIAL_STATE"              varchar(100),
    "RUN_TIME"                   longtext,
    "ORDER_XML"                  longtext,
    primary key ( "SPOOLER_ID", "JOB_CHAIN", "ID" )
)
Type=InnoDB;

CREATE TABLE SCHEDULER_ORDER_HISTORY
(
    "HISTORY_ID" integer not null,
    "JOB_CHAIN"  varchar(255) not null,
    "ORDER_ID"   varchar(255) not null,
    "SPOOLER_ID" varchar(100) not null,
    "TITLE"      varchar(200),
    "STATE"      varchar(100),
    "STATE_TEXT" varchar(100),
    "START_TIME" datetime not null,
    "END_TIME"   datetime not null,
    "LOG"        longblob,
    primary key( "HISTORY_ID" )
)
Type=InnoDB;

CREATE INDEX SCHEDULER_O_HISTORY_SPOOLER_ID on SCHEDULER_ORDER_HISTORY ( "SPOOLER_ID"
 );
CREATE INDEX SCHEDULER_O_HISTORY_JOB_CHAIN  on SCHEDULER_ORDER_HISTORY ( "JOB_CHAIN"
);
CREATE INDEX SCHEDULER_O_HISTORY_START_TIME on SCHEDULER_ORDER_HISTORY ( "START_TIME"
 );
```

```
CREATE TABLE SCHEDULER_ORDER_STEP_HISTORY
(
  "HISTORY_ID"          integer       not null,
  "STEP"                integer       not null,
  "TASK_ID"             integer       not null,
  "STATE"               varchar(100)          ,
  "START_TIME"          datetime      not null,
  "END_TIME"            datetime              ,
  primary key ( "HISTORY_ID","STEP" )
);

CREATE TABLE SCHEDULER_CLUSTERS
(
    "MEMBER_ID"              varchar(100) not null,
    "SCHEDULER_ID"           varchar(100) not null,
    "PRECEDENCE"             integer,
    "LAST_HEART_BEAT"        integer,
    "NEXT_HEART_BEAT"        integer,
    "ACTIVE"                 bool,
    "EXCLUSIVE"              bool,
    "DEAD"                   bool,
    "COMMAND"                varchar(250),
    "HTTP_URL"               varchar(100),
    "DEACTIVATING_MEMBER_ID" varchar(100),
    "XML"                    longtext,
    primary key( "MEMBER_ID" )
)
Type=InnoDB;
```

C.4 SQL Instructions Used by the JobScheduler for Oracle®

```
CREATE TABLE SCHEDULER_VARIABLES
(
    "NAME"     varchar2(100) not null,
    "WERT"     integer,
    "TEXTWERT" varchar2(250),
    primary key ( "NAME" )
);

CREATE TABLE SCHEDULER_TASKS
(
    "TASK_ID"           integer not null,
    "SPOOLER_ID"        varchar2(100) not null,
    "CLUSTER_MEMBER_ID" varchar2(100),
    "JOB_NAME"          varchar2(255) not null,
    "ENQUEUE_TIME"      date,
    "START_AT_TIME"     date,
    "PARAMETERS"        clob,
    "TASK_XML"          clob,
    primary key( "TASK_ID" )
);

CREATE TABLE SCHEDULER_HISTORY
(
    "ID"                integer not null,
    "SPOOLER_ID"        varchar2(100) not null,
    "CLUSTER_MEMBER_ID" varchar2(100),
    "JOB_NAME"          varchar2(255) not null,
    "START_TIME"        date not null,
    "END_TIME"          date,
    "CAUSE"             varchar2(50),
    "STEPS"             integer,
    "EXIT_CODE"         integer,
    "ERROR"             numeric(1),
    "ERROR_CODE"        varchar2(50),
    "ERROR_TEXT"        varchar2(250),
    "PARAMETERS"        clob,
    "LOG"               blob,
    primary key( "ID" )
);

CREATE INDEX SCHEDULER_HISTORY_START_TIME on SCHEDULER_HISTORY ("START_TIME"       );
CREATE INDEX SCHEDULER_HISTORY_SPOOLER_ID on SCHEDULER_HISTORY ("SPOOLER_ID"       );
CREATE INDEX SCHEDULER_HISTORY_JOB_NAME   on SCHEDULER_HISTORY ("JOB_NAME"         );
CREATE INDEX SCHEDULER_H_CLUSTER_MEMBER   on SCHEDULER_HISTORY ("CLUSTER_MEMBER_ID");
```

```
CREATE TABLE SCHEDULER_ORDERS
(
    "JOB_CHAIN"                    varchar2(255) not null,
    "ID"                           varchar2(255) not null,
    "SPOOLER_ID"                   varchar2(100) not null,
    "DISTRIBUTED_NEXT_TIME"        date,
    "OCCUPYING_CLUSTER_MEMBER_ID"  varchar2(100),
    "PRIORITY"                     integer not null,
    "STATE"                        varchar2(100),
    "STATE_TEXT"                   varchar2(100),
    "TITLE"                        varchar2(200),
    "CREATED_TIME"                 date not null,
    "MOD_TIME"                     date,
    "ORDERING"                     integer not null,
    "PAYLOAD"                      clob,
    "INITIAL_STATE"                varchar2(100),
    "RUN_TIME"                     clob,
    "ORDER_XML"                    clob,
    primary key( "SPOOLER_ID", "JOB_CHAIN", "ID" )
);

CREATE TABLE SCHEDULER_ORDER_HISTORY
(
    "HISTORY_ID" integer not null,
    "JOB_CHAIN"  varchar2(255) not null,
    "ORDER_ID"   varchar2(255) not null,
    "SPOOLER_ID" varchar2(100) not null,
    "TITLE"      varchar2(200),
    "STATE"      varchar2(100),
    "STATE_TEXT" varchar2(100),
    "START_TIME" date not null,
    "END_TIME"   date not null,
    "LOG"        blob,
    primary key( "HISTORY_ID" )
);
```

```
CREATE INDEX SCHEDULER_O_HISTORY_SPOOLER_ID on SCHEDULER_ORDER_HISTORY ( "SPOOLER_ID"
 );
CREATE INDEX SCHEDULER_O_HISTORY_JOB_CHAIN  on SCHEDULER_ORDER_HISTORY ( "JOB_CHAIN"
);
CREATE INDEX SCHEDULER_O_HISTORY_START_TIME on SCHEDULER_ORDER_HISTORY ( "START_TIME"
 );

CREATE TABLE SCHEDULER_ORDER_STEP_HISTORY
(
  "HISTORY_ID"          number(9)     not null,
  "STEP"                number(9)     not null,
  "TASK_ID"             number(9)     not null,
  "STATE"               varchar2(100)         ,
  "START_TIME"          date          not null,
  "END_TIME"            date                  ,
  primary key ( "HISTORY_ID","STEP" )
);

CREATE TABLE SCHEDULER_CLUSTERS
(
    "MEMBER_ID"              varchar2(100) not null,
    "SCHEDULER_ID"           varchar2(100) not null,
    "PRECEDENCE"             integer,
    "LAST_HEART_BEAT"        integer,
    "NEXT_HEART_BEAT"        integer,
    "ACTIVE"                 numeric(1),
    "EXCLUSIVE"              numeric(1),
    "DEAD"                   numeric(1),
    "COMMAND"                varchar2(250),
    "HTTP_URL"               varchar2(100),
    "DEACTIVATING_MEMBER_ID" varchar2(100),
    "XML"                    clob,
    primary key( "MEMBER_ID" )
);
```

C.5 SQL Instructions Used by the JobScheduler for PostgreSQL

```
CREATE TABLE SCHEDULER_VARIABLES
(
    "NAME"     varchar(100) not null,
    "WERT"     integer,
    "TEXTWERT" varchar(250),
    primary key ( "NAME" )
);

CREATE TABLE SCHEDULER_TASKS
(
    "TASK_ID"           integer not null,
    "SPOOLER_ID"        varchar(100) not null,
    "CLUSTER_MEMBER_ID" varchar(100),
    "JOB_NAME"          varchar(255) not null,
    "ENQUEUE_TIME"      timestamp,
    "START_AT_TIME"     timestamp,
    "PARAMETERS"        text,
    "TASK_XML"          text,
    primary key( "TASK_ID" )
);

CREATE TABLE SCHEDULER_HISTORY
(
    "ID" integer not null,
    "SPOOLER_ID"        varchar(100) not null,
    "CLUSTER_MEMBER_ID" varchar(100),
    "JOB_NAME"          varchar(255) not null,
    "START_TIME"        timestamp not null,
    "END_TIME"          timestamp,
    "CAUSE"             varchar(50),
    "STEPS"             integer,
    "EXIT_CODE"         integer,
    "ERROR"             numeric(1),
    "ERROR_CODE"        varchar(50),
    "ERROR_TEXT"        varchar(250),
    "PARAMETERS"        text,
    "LOG"               bytea,
    primary key( "ID" )
)

CREATE INDEX SCHEDULER_HISTORY_START_TIME on SCHEDULER_HISTORY ("START_TIME"       );
CREATE INDEX SCHEDULER_HISTORY_SPOOLER_ID on SCHEDULER_HISTORY ("SPOOLER_ID"       );
CREATE INDEX SCHEDULER_HISTORY_JOB_NAME   on SCHEDULER_HISTORY ("JOB_NAME"         );
CREATE INDEX SCHEDULER_H_CLUSTER_MEMBER   on SCHEDULER_HISTORY ("CLUSTER_MEMBER_ID");

CREATE TABLE SCHEDULER_ORDERS
(
    "JOB_CHAIN"                    varchar(255) not null,
    "ID"                           varchar(255) not null,
    "SPOOLER_ID"                   varchar(100) not null,
    "DISTRIBUTED_NEXT_TIME"        timestamp,
    "OCCUPYING_CLUSTER_MEMBER_ID"  varchar(100),
    "PRIORITY"                     integer not null,
    "STATE"                        varchar(100),
    "STATE_TEXT"                   varchar(100),
    "TITLE"                        varchar(200),
    "CREATED_TIME"                 timestamp not null,
    "MOD_TIME"                     timestamp,
    "ORDERING"                     integer not null,
    "PAYLOAD"                      text,
    "INITIAL_STATE"                varchar(100),
    "RUN_TIME"                     text,
    "ORDER_XML"                    text,
    primary key( "SPOOLER_ID", "JOB_CHAIN", "ID" )
);

CREATE TABLE SCHEDULER_ORDER_HISTORY
(
    "HISTORY_ID" integer not null,
    "JOB_CHAIN"  varchar(255) not null,
    "ORDER_ID"   varchar(255) not null,
    "SPOOLER_ID" varchar(100) not null,
    "TITLE"      varchar(200),
    "STATE"      varchar(100),
    "STATE_TEXT" varchar(100),
    "START_TIME" timestamp not null,
    "END_TIME"   timestamp not null,
    "LOG"        bytea,
    primary key( "HISTORY_ID" )
);

CREATE INDEX SCHEDULER_O_HISTORY_SPOOLER_ID on SCHEDULER_ORDER_HISTORY ( "SPOOLER_ID"
 );
CREATE INDEX SCHEDULER_O_HISTORY_JOB_CHAIN  on SCHEDULER_ORDER_HISTORY ( "JOB_CHAIN"
);
CREATE INDEX SCHEDULER_O_HISTORY_START_TIME ON SCHEDULER_ORDER_HISTORY ( "START_TIME"
 );

CREATE TABLE SCHEDULER_ORDER_STEP_HISTORY
(
  "HISTORY_ID"           numeric(9)     not null,
  "STEP"                 numeric(9)     not null,
  "TASK_ID"              numeric(9)     not null,
  "STATE"                varchar(100)           ,
  "START_TIME"           timestamp      not null,
  "END_TIME"             timestamp              ,
  primary key ( "HISTORY_ID","STEP" )
);

CREATE TABLE SCHEDULER_CLUSTERS
(
    "MEMBER_ID"               varchar(100) not null,
    "SCHEDULER_ID"            varchar(100) not null,
    "PRECEDENCE"              integer,
    "LAST_HEART_BEAT"         integer,
    "NEXT_HEART_BEAT"         integer,
    "ACTIVE"                  numeric(1),
    "EXCLUSIVE"               numeric(1),
    "DEAD"                    numeric(1),
    "COMMAND"                 varchar(250),
    "HTTP_URL"                varchar(100),
    "DEACTIVATING_MEMBER_ID"  varchar(100),
    "XML"                     text,
    primary key( "MEMBER_ID" )
);
```

C.6 SQL Instructions Used by the JobScheduler for Sybase ASE

```
CREATE TABLE SCHEDULER_VARIABLES
(
    "NAME"     varchar(100) not null,
    "WERT"     integer,
    "TEXTWERT" varchar(250),
    primary key ( "NAME" )
);
```

```
CREATE TABLE SCHEDULER_TASKS
(
    "TASK_ID"            integer not null,
    "SPOOLER_ID"         varchar(100) not null,
    "CLUSTER_MEMBER_ID"  varchar(100),
    "JOB_NAME"           varchar(255) not null,
    "ENQUEUE_TIME"       datetime,
    "START_AT_TIME"      datetime,
    "PARAMETERS"         text,
    "TASK_XML"           text,
    primary key( "TASK_ID" )
);

CREATE TABLE SCHEDULER_HISTORY
(
    "ID"                 integer not null,
    "SPOOLER_ID"         varchar(100) not null,
    "CLUSTER_MEMBER_ID"  varchar(100),
    "JOB_NAME"           varchar(255) not null,
    "START_TIME"         datetime not null,
    "END_TIME"           datetime,
    "CAUSE"              varchar(50),
    "STEPS"              integer,
    "EXIT_CODE"          integer,
    "ERROR"              numeric(1),
    "ERROR_CODE"         varchar(50),
    "ERROR_TEXT"         varchar(250),
    "PARAMETERS"         text,
    "LOG"                image,
    primary key( "ID" )
);

CREATE INDEX SCHEDULER_HISTORY_START_TIME on SCHEDULER_HISTORY ( "START_TIME" );
CREATE INDEX SCHEDULER_HISTORY_SPOOLER_ID on SCHEDULER_HISTORY ( "SPOOLER_ID" );
CREATE INDEX SCHEDULER_HISTORY_JOB_NAME   on SCHEDULER_HISTORY ( "JOB_NAME" );
CREATE INDEX SCHEDULER_H_CLUSTER_MEMBER   on SCHEDULER_HISTORY (
"CLUSTER_MEMBER_ID");

CREATE TABLE SCHEDULER_ORDERS
(
    "JOB_CHAIN"                    varchar(250) not null,
    "ID"                           varchar(250) not null,
    "SPOOLER_ID"                   varchar(100) not null,
    "DISTRIBUTED_NEXT_TIME"        datetime,
    "OCCUPYING_CLUSTER_MEMBER_ID"  varchar(100),
    "PRIORITY"                     integer not null,
    "STATE"                        varchar(100),
    "STATE_TEXT"                   varchar(100),
    "TITLE"                        varchar(200),
    "CREATED_TIME"                 datetime not null,
    "MOD_TIME"                     datetime,
    "ORDERING"                     integer not null,
    "PAYLOAD"                      text,
    "INITIAL_STATE"                varchar(100),
    "RUN_TIME"                     text,
    "ORDER_XML"                    text,
    primary key( "SPOOLER_ID", "JOB_CHAIN", "ID" );
);

CREATE TABLE SCHEDULER_ORDER_HISTORY
(
    "HISTORY_ID" integer not null,
    "JOB_CHAIN"  varchar(255) not null,
    "ORDER_ID"   varchar(255) not null,
    "SPOOLER_ID" varchar(100) not null,
    "TITLE"      varchar(200),
    "STATE"      varchar(100),
    "STATE_TEXT" varchar(100),
    "START_TIME" datetime not null,
    "END_TIME"   datetime not null,
    "LOG"        image,
    primary key( "HISTORY_ID" );
);

CREATE INDEX SCHEDULER_O_HISTORY_SPOOLER_ID on SCHEDULER_ORDER_HISTORY ( "SPOOLER_ID"
 );
CREATE INDEX SCHEDULER_O_HISTORY_JOB_CHAIN  on SCHEDULER_ORDER_HISTORY ( "JOB_CHAIN"
);
CREATE INDEX SCHEDULER_O_HISTORY_START_TIME on SCHEDULER_ORDER_HISTORY ( "START_TIME"
 );

CREATE TABLE SCHEDULER_ORDER_STEP_HISTORY
(
  "HISTORY_ID"          numeric(9)     not null,
  "STEP"                numeric(9)     not null,
  "TASK_ID"             numeric(9)     not null,
  "STATE"               varchar(100)           ,
  "START_TIME"          datetime       not null,
  "END_TIME"            datetime               ,
  primary key ( "HISTORY_ID","STEP" )
);

CREATE TABLE SCHEDULER_CLUSTERS
(
    "MEMBER_ID"              varchar(100) not null,
    "SCHEDULER_ID"           varchar(100) not null,
    "PRECEDENCE"             integer,
    "LAST_HEART_BEAT"        integer,
    "NEXT_HEART_BEAT"        integer,
    "ACTIVE"                 numeric(1),
    "EXCLUSIVE"              numeric(1),
    "DEAD"                   numeric(1),
    "COMMAND"                varchar(250),
    "HTTP_URL"               varchar(100),
    "DEACTIVATING_MEMBER_ID" varchar(100),
    "XML"                    text,
    primary key( "MEMBER_ID" )
);
```

Appendix D: Scripts in JavaScript

Two versions of JavaScript are available:

<script language="JavaScript">

Spidermonkey from Mozilla, is delivered with the Scheduler.

<script language="JScript">

Microsoft JScript comes with Windows systems (the version depends on the operating system and Internet Explorer).

D.1 Java-Exceptions in Spidermonkey

Exception objects of differing classes are generated in the event of exceptions in Java constructors and in Java methods. The error message for every exception (including normal JavaScript exceptions) can be determined using these objects:

```
catch( x )
{
    var error_text = x + "";
}
```

The following example takes account of different exceptions:

```
function log_exception( x )
{
    var msg = x + "";

    if( x.getMessage           )  spooler_log.error( "Java-Exception: " + msg ),
x.printStackTrace( java.lang.System.out );
    else
    if( x.message != undefined )  spooler_log.error( "A JavaScript exception: " + msg
 );

    else
    if( x.length != undefined  )  spooler_log.error( "Apparently a Java constructor
exception: " + msg );

    else
                                  spooler_log.error( "Unknown Exception: " + msg );

}


try
{
    throw new Error( "A JavaScript error" );

}
catch( x )
```

```
{
    log_exception( x );                                    // x is an error

}

try
{
    var file = java.io.FileInputStream( "gibtsnicht" );    // Exception in a
constructor

}
catch( x )
{
    log_exception( x );                                    // x is a character array

}

try
{
    java.lang.Class.forName( "blah" );                     // Exception in a Java
method

}
catch( x )
{
    log_exception( x );                                    // x is a Java exception

}
```

Index