



# Managed User Jobs

# JOB SCHEDULER

[Documentation](#)

April 2006

## Contact Information

Software- und Organisations-Service GmbH  
Giesebrechtstr. 15  
10629 Berlin  
Germany

Telephone (030) 86 47 90-0  
Telefax (030) 8 61 33 35  
Mail [info@sos-berlin.com](mailto:info@sos-berlin.com)  
Web [www.sos-berlin.com](http://www.sos-berlin.com)

Last Updated: November 2005

# Table of Contents

<b>1 Introduction</b> .....	<b>4</b>
<b>2 Installation</b> .....	<b>5</b>
2.1 Job Scheduler .....	5
2.2 MySQL Procedures .....	5
2.3 MySQL UDF for UDP Datagrammes .....	5
2.4 Configuration of User Rights .....	6
<b>3 The SQL Interface</b> .....	<b>7</b>
3.1 JOB_SUBMIT .....	7
3.2 JOB_RUN .....	7
3.3 JOB_DELETE .....	8
3.4 JOB_SET_SCHEMA .....	8
3.5 JOB_SET_ACTION .....	8
3.6 JOB_SET_TITLE .....	9
3.7 JOB_SET_PRIORITY .....	9
3.8 JOB_SET_SUSPENDED .....	9
3.9 JOB_SET_NEXT_START .....	9
<b>4 The SQL Interface for Database Reports</b> .....	<b>11</b>
4.1 REPORT_JOB_SUBMIT .....	11
4.2 REPORT_JOB_SET_MAILTO .....	11
4.3 REPORT_JOB_SET_MAILCC .....	11
4.4 REPORT_JOB_SET_MAILBCC .....	12
4.5 REPORT_JOB_SET_ASBODY .....	12
4.6 REPORT_JOB_SET_BODY .....	12
4.7 REPORT_JOB_SET_SUBJECT .....	12
4.8 REPORT_JOB_SET_STYLESHEET .....	12
4.9 REPORT_JOB_SET_PATH .....	13
4.10 REPORT_JOB_SET_FILENAME .....	13
4.11 SEND_MAIL .....	13

# 1 Introduction

The `dbms_scheduler` or `dbms_job` packages are used to execute database statements at specific times under Oracle. Such components are not included in any of the MySQL distributions. The Managed User Jobs package has been written to fulfill this need, and offers an SQL interface for the definition of jobs similar to the Oracle `dbms_scheduler` interface. The Managed User Jobs package allows users to plan the execution of jobs within their allowed user rights using an external Job Scheduler.

In addition Managed User Jobs offers the possibility of informing registered recipients per E-mail about the success or failure of statement execution. Use of the Managed User Jobs requires that the following conditions are fulfilled:

- A MySQL 5 database (MySQL 4.1.1+ works within limits)
- A database user with super user rights
- Installation of the `udf_sos_send_udp` shared object (or dll on Windows) for the MySQL Server
- Installation of the Job Scheduler with database access for the user mentioned above
- The MySQL Maintenance Jobs Package
- The Scheduler must be configured so that the *JobSchedulerManagedUserJob* is started every minute.
- The database user requires further rights in order to be able to set up his own jobs. See Installation.

## 2 Installation

A MySQL database (at best Version 5+) is required before the Job Scheduler is installed. Installation and update instructions together with the installation files are to be found on the official MySQL web site. The server version must be installed.

### 2.1 Job Scheduler

The Job Scheduler may be installed on a different computer to the database.

Before installation of the Job Scheduler it is recommended that the necessary database user(s) and schema are created. This is described in detail in the MySQL documentation.

It is recommended that the Job Scheduler is installed using its own installation program. During installation of the Scheduler it is necessary that the "Database Support" and "MySQL Maintenance Jobs" packages are activated by clicking the appropriate check boxes. The Job Scheduler will create the necessary database tables for itself and the Managed User Jobs if required. This option may, however, be deactivated and the tables manually created later.

### 2.2 MySQL Procedures

MySQL procedures must be manually installed after the installation of the Job Scheduler. The script for this can be found in the `<scheduler_procedures_directory>/db/mysql/procedures` directory. The procedure for running this script is as follows:

- make a connection with the MySQL server using a MySQL client as Scheduler user;
- select the Job Scheduler database schema (e.g. `use scheduler`)
- change the delimiters to other characters: `delimiter //`
- read in the following procedure: `source <path>/scheduler_job_procedure.sql`
- reset the delimiter using: `delimiter ;`

### 2.3 MySQL UDF for UDP Datagrammes

A user defined function for sending UDP commands is provided in the directory `<scheduler_installationsverzeichnis>/lib/`. This function need not be installed unless the `JOB_RUN` procedure is to be used.

**Linux:** the path in which the `udf_sos_send_udp.so` file is to be found must be added to the `LD_LIBRARY_PATH` variable in the MySQL server start script, e.g.:

```
export LD_LIBRARY_PATH=/pfad/pfad:$LD_LIBRARY_PATH;
```

The MySQL server must now be restarted. After restarting, the presence of the function will be disclosed to the MySQL server with `create function sos_send_udp returns string soname 'udf_sos_send_udp.so'` ;

**Windows:** The `udf_sos_send_udp.dll` file must be copied into the the bin directory of the MySQL server. The MySQL server must then be restarted. After restarting, the MySQL server will disclose the presence of the function with `create function sos_send_udp returns string soname 'udf_sos_send_udp.dll'` ;

## 2.4 Configuration of User Rights

The Job Scheduler executes planned database statements. To do this it creates new users during run time. In order to be able to do this, it is necessary that the Job Scheduler possesses super user administrative rights:

```
grant all on *.* to <scheduler_user>@<scheduler_host>
identified by <scheduler_password> with grant option

grant all on mysql.* to <scheduler_user>@<scheduler_host>
identified by <scheduler_password> with grant option
```

Every normal database user who is to be allowed to set up his own jobs requires access rights for the tables used by these jobs, as well as the right to carry out the Scheduler users procedures:

```
grant execute on scheduler.* to <user_name>@<user_host>
```

This allows the user to set up his own jobs and only be able to modify his own jobs.

This last statement does not apply to MySQL from Version 4.1 up to (but not including) Version 5 as there are no procedures. Instead the user must be allowed access to the scheduler\_managed\_user\_jobs table:

```
grant all on scheduler.scheduler_managed_user_jobs to <user_name>@<user_host>
```

Naturally this sets the basic security concept described above out of action.

## 3 The SQL Interface

The job administration functions (similarly to Oracle) directly through a SQL interface. The procedures documented here are to be found in the Scheduler user's schema. In order to use these procedures it is necessary that the appropriate schema is selected (e.g. `use scheduler`) or that a qualified procedure call is made (e.g. `call scheduler.JOB_SUBMIT(...)`).

In order to be able to identify the result of a procedure call, all procedures have a common first parameter included in the output string. This parameter can then be later used to locate procedure output.

```
call scheduler.JOB_SET_SCHEMA(@result, 1, 'test');

SELECT @result;
```

Should it be necessary to call several API procedures when processing a job, then it is recommended that these are carried out in one transaction. Otherwise it is possible that a job is run before all parameters have been completely specified.

### 3.1 JOB\_SUBMIT

#### Parameters:

- OUT job\_result VARCHAR(250)
- OUT job\_id INT
- IN job\_action TEXT
- IN job\_start\_time DATETIME
- IN job\_next\_start TEXT
- IN job\_schema VARCHAR(250)

JOB\_SUBMIT creates a new job. This job is identified by the returned `job_id` which, in turn, is used by other procedures as an IN parameter. The SQL command to be carried out is to be found in `job_action`. It is possible to give a number of SQL commands together, which must, however, be separated by semicolons (see JOB\_SET\_ACTION). The starting time for the first start of a job is given using `job_start_time`. If the value of this parameter is null, then the job will be started as soon as possible (but after the first commit). The method by which the next start time for the job is calculated is given by `job_next_start`. This start time is calculated after the original job has run. Should this parameter be null, then the job will only run once. See also JOB\_SET\_NEXT\_START. The data bank schema to be used is specified using `job_schema`. This parameter may not have the value null and the user must possess the appropriate rights for the schema.

#### Example: Setting up a Job

```
call scheduler.JOB_SUBMIT(@result, @jobid, 'UPDATE persondata SET age=age+1;', null, 'ADDTIME(NOW(), "1:0:0")', 'test');
```

The '1:0:0' String (one hour) must be in double inverted commas, as it lies in another string.

### 3.2 JOB\_RUN

#### Parameters:

- OUT job\_result VARCHAR(250)
- IN job\_id INT
- IN job\_start\_time DATETIME

Either gives a job new start time immediately, or starts the job immediately should `job_start_time` be null. Thereby a command is given to the Job Scheduler per UDP. For this to work, it is necessary that the User Defined Function is installed. This command should not be used together with other commands within a transaction but singly - for example to start a job for test purposes.

**Example: Start Job 25 Immediately**

```
call scheduler.JOB_RUN(@result, 25, null);
```

### 3.3 JOB\_DELETE

**Parameters:**

- OUT job\_result VARCHAR(250)
- IN job\_id INT

Deletes the Specified Job.

**Example: Delete Job 20**

```
call scheduler.JOB_DELETE(@result, 20);
```

### 3.4 JOB\_SET\_SCHEMA

**Parameters:**

- OUT job\_result VARCHAR(250)
- IN job\_id INT
- IN job\_schema VARCHAR(250)

Sets the database schema in which a job is to be run.

**Example: Set Schema for Job 19 to 'test'.**

```
call scheduler.JOB_SET_SCHEMA(@result, 19, 'test');
```

### 3.5 JOB\_SET\_ACTION

**Parameters:**

- OUT job\_result VARCHAR(250)
- IN job\_id INT
- IN job\_action TEXT

Sets the commands to be carried out for a job. Any number of commands separated by semi-colons can be specified.

**Example: Multiple Commands for a Job**

```
call scheduler.JOB_SET_ACTION(@result, 19, 'UPDATE persondata SET age=age+1; UPDATE persondata SET weight=weight-1;');
```

## 3.6 JOB\_SET\_TITLE

### Parameters:

- OUT job\_result VARCHAR(250)
- IN job\_id INT
- IN job\_title VARCHAR(250)

Specify the Job Title.

#### **Example: Specify Title**

call scheduler.JOB\_SET\_TITLE(@result, 15, 'Testjob');

## 3.7 JOB\_SET\_PRIORITY

### Parameters:

- OUT job\_result VARCHAR(250)
- IN job\_id INT
- IN job\_priority INT

Specifies the priority of a job. Should insufficient tasks be available to carry out all jobs together, then jobs with a higher priority will be carried out first. The default priority value is 1.

#### **Example: Set the Priority of Job 13 to 5**

call scheduler.JOB\_SET\_PRIORITY(@result, 13, 5);

## 3.8 JOB\_SET\_SUSPENDED

### Parameters:

- OUT job\_result VARCHAR(250)
- IN job\_id INT
- IN job\_suspended BOOL

Activates (job\_suspended = false) or deactivates (job\_suspended = true) a Job.

#### **Example: Deactivate Job 16**

call scheduler.JOB\_SET\_SUSPENDED(@result, 16, true);

## 3.9 JOB\_SET\_NEXT\_START

### Parameters:

- OUT job\_result VARCHAR(250)
- IN job\_id INT
- IN job\_next\_start TEXT

Sets the rules by which the next start time for a job is calculated. This must be an SQL expression which returns an ISO date. The expression is always evaluated after a job has been completed.

#### **Example: Start Job 10 Every Day at the Same Time (every 24h) After the End of the First Run**

call scheduler.JOB\_SET\_NEXT\_START(@result, 10, 'ADDDATE(NOW(), 1)');

**Example: Start Job 10 Once Every Hour**

```
call scheduler.JOB_SET_NEXT_START(@result, 10, 'ADDDATE(NOW(), "01:00:00");
```

**Example: Start Job 10 every 5 Minutes**

```
call scheduler.JOB_SET_NEXT_START(@result, 10, 'ADDTIME(NOW(), "0:5:0");
```

**Example: Start Job 10 Every Hour, On the Hour**

```
call scheduler.JOB_SET_NEXT_START(@result, 10, 'ADDDATE(CURRENT_DATE(),  
INTERVAL EXTRACT(HOUR FROM CURRENT_TIME()+1 HOUR));
```

**Example: Start Job 10 to the Ultimo of Every Month at 18:00**

```
call scheduler.JOB_SET_NEXT_START(@result, 10, 'adddate(last_day(current_date()), interval 18 hour);
```

**Example: Start Job 10 every Day at 5:00**

```
call scheduler.JOB_SET_NEXT_START(@result, 10, 'IF(HOUR(NOW())>=5,  
ADDDATE(CURRENT_DATE(), INTERVAL 24+5 HOUR),  
ADDDATE(CURRENT_DATE(), INTERVAL 5 HOUR));
```

## 4 The SQL Interface for Database Reports

After the Managed Jobs packet has been installed, the following additional database report jobs can be created. These jobs function similarly to normal database jobs. After the selection statements are completed, a report containing the result of the last select statement is sent by e-mail. This process is described in more detail in the Managed Jobs documentation. The SQL interface described above is valid for the report jobs, together with the following functions:

### 4.1 REPORT\_JOB\_SUBMIT

#### Parameters:

- OUT job\_result VARCHAR(250)
- OUT job\_id INT
- IN job\_action TEXT
- IN job\_start\_time DATETIME
- IN job\_next\_start TEXT
- IN job\_schema VARCHAR(250)
- IN report\_recipient VARCHAR(250)

REPORT\_JOB\_SUBMIT creates a new report job similar to JOB\_SUBMIT. The `report_recipient` parameter is, however, new and specifies the e-mail address of the report recipient.

#### Example: Creation of a Report Job

```
call scheduler.REPORT_JOB_SUBMIT(@result, @jobid, 'SELECT * FROM persondata;', null, 'ADDTIME(NOW(), "1:0:0")', 'test', 'info@sos-berlin.com');
```

### 4.2 REPORT\_JOB\_SET\_MAILTO

#### Parameters:

- OUT job\_result VARCHAR(250)
- IN job\_id INT
- IN report\_recipient VARCHAR(250)

Specifies the e-mail address of the report recipient.

#### Example: Set the report address von job 30 to 'info@sos-berlin.com'.

```
call scheduler.REPORT_JOB_SET_MAILTO(@result, 30, 'info@sos-berlin.com');
```

### 4.3 REPORT\_JOB\_SET\_MAILCC

#### Parameters:

- OUT job\_result VARCHAR(250)
- IN job\_id INT
- IN report\_recipient VARCHAR(250)

Specifies the e-mail address of the report cc recipient.

## 4.4 REPORT\_JOB\_SET\_MAILBCC

### Parameters:

- OUT job\_result VARCHAR(250)
- IN job\_id INT
- IN report\_recipient VARCHAR(250)

Specifies the e-mail address of the report bcc recipient.

## 4.5 REPORT\_JOB\_SET\_ASBODY

### Parameters:

- OUT job\_result VARCHAR(250)
- IN job\_id INT
- IN asbody BOOL

If `asbody` is true(1), then the report will be sent as the e-mail body and not as an attachment.

## 4.6 REPORT\_JOB\_SET\_BODY

### Parameters:

- OUT job\_result VARCHAR(250)
- IN job\_id INT
- IN body TEXT

The `body` parameter specifies the report e-mail body, when `asbody` is not set to true.

## 4.7 REPORT\_JOB\_SET\_SUBJECT

### Parameters:

- OUT job\_result VARCHAR(250)
- IN job\_id INT
- IN subject VARCHAR(250)

The `subject` parameter specifies the report e-mail subject.

## 4.8 REPORT\_JOB\_SET\_STYLESHEET

### Parameters:

- OUT job\_result VARCHAR(250)
- IN job\_id INT
- IN stylesheet VARCHAR(250)

The `stylesheet` parameter specifies the stylesheet to be used in the transformation of the report.

## 4.9 REPORT\_JOB\_SET\_PATH

### Parameters:

- OUT job\_result VARCHAR(250)
- IN job\_id INT
- IN path VARCHAR(250)

If a path is specified here, then the report is not only sent by e-mail but also saved in the specified directory.

## 4.10 REPORT\_JOB\_SET\_FILENAME

### Parameters:

- OUT job\_result VARCHAR(250)
- IN job\_id INT
- IN filename VARCHAR(250)

The `filename` parameter specifies the file name mask for the report file.

## 4.11 SEND\_MAIL

### Parameters:

- OUT job\_result VARCHAR(250)
- IN mailto VARCHAR(250)
- IN subject VARCHAR(250)
- IN body TEXT

This function sends an e-mail with the `subject` subject to the `mailto` recipient. The e-mail text is specified with `body`. HTML can also be used here.