

# How to Set Up pgAgent for Postgres Plus<sup>®</sup>

## **A Postgres Evaluation Quick Tutorial From EnterpriseDB**

**February 19, 2010**

EnterpriseDB Corporation, 235 Littleton Road, Westford, MA 01866, USA  
**T** +1 978 589 5700 **F** +1 978 589 5701 **E** [info@enterprisedb.com](mailto:info@enterprisedb.com) **www**.[enterprisedb.com](http://enterprisedb.com)

# Introduction

This [EnterpriseDB](#) Quick Tutorial helps you get started with the [Postgres Plus Standard Server](#) and [Postgres Plus Advanced Server](#) database products. It is assumed that you have already downloaded and installed Postgres Plus Standard Server or Postgres Plus Advanced Server on your desktop or laptop computer.

This Quick Tutorial is designed to help you expedite your Technical Evaluation of Postgres Plus Standard Server or Postgres Plus Advanced Server. In this Quick Tutorial you will learn how to set up pgAgent for a Postgres Plus database.

For more informational assets on conducting your evaluation of Postgres Plus, visit the self-service web site, [Postgres Plus Open Source Adoption](#).

In this Quick Tutorial you will learn the following:

- The variety of features of pgAgent
- Verify a working pgAgent installation
- Create a pgAgent job
- Create one or more job steps for a job
- Create one or more job schedules for a job

## Feature Description

pgAgent is a pre-bundled enterprise module installed by default with Postgres Plus Standard Server and Postgres Plus Advanced Server.

pgAgent is a job scheduler that allows authorized users to schedule jobs consisting of one or more SQL or shell/batch steps. Common uses include scheduled tasks such as archiving old records from a table, vacuuming certain tables, or performing application-oriented operations such as batch posting daily transactions to accounts.

Using pgAdmin, you create a *job* to define what you want done and when you want it done. The specific actions that you want accomplished are defined in a series of one or more *steps*. When you want the actions in the steps to occur is defined in one more *schedules*.

The following are some of the features illustrating the power and flexibility of pgAgent:

- Jobs can be as simple as a single step, or can include a series of multiple steps each of which can be independently activated.
- Steps can consist of SQL statements acting upon a local or remote Postgres Plus

database.

- Steps can consist of shell script commands (for Linux systems) or batch commands (for Microsoft Windows® systems).
- Setting up a schedule is a simple procedure. You select the times a job is to run, the day of the week it is to run, the days of the month it is to run, or months of the year it is to run. You can set up a job to run at every given date/time interval by omitting a selection in the category. For example, omitting a selection in the Months category sets the schedule so the job runs every month. In addition, you can specify exceptions to the schedule you established.
- You can set up multiple schedules for a single job that can be independently activated.
- A history is maintained that lets you view the outcome of past job runs.
- Statistics are maintained that show you the last time a job was run, if it is currently running, and the next time it is scheduled to run.

Typically a single instance of pgAgent is installed and used on the database server itself, though in some circumstances it is beneficial to install one or more instances on machines other than the database server.

The pgAdmin graphical user interface (Postgres Studio on Advanced Server) is used to create and manage pgAgent jobs.

The [pgAgent](#) page is an index to the complete pgAgent documentation.

Additional information about pgAgent and the pgAgent project can be found on the [Postgres Community Projects](#) page of the [EnterpriseDB](#) web site.

## Tutorial Steps

**Step 1:** Verify pgAgent is installed on the database server. You should see a subdirectory named `pgAgent` under the Postgres Plus home directory. (In Advanced Server there is a subdirectory with the path `dbserver/share/pgagent` under the Postgres Plus Advanced Server home directory.)

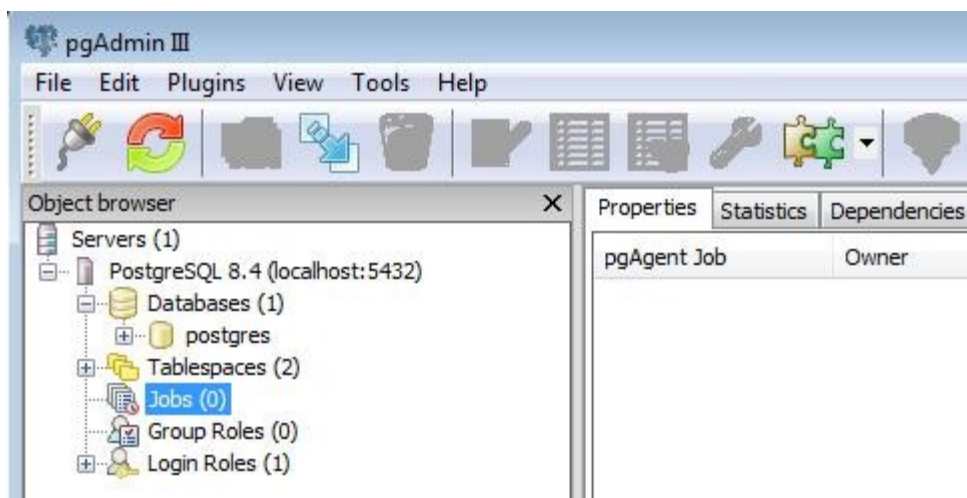
**Note:** When installing Standard Server, if you de-selected the pgAgent component, the subdirectory `pgAgent` and its contents are not installed. If you did not install pgAgent, you can use StackBuilder Plus to add pgAgent to your Standard Server configuration.

The following table lists the names and directory locations of certain pgAgent files that may be useful as you progress through this tutorial. The directory locations are relative to the Postgres Plus home directory.

File Name	Location (Standard Server)	Location (Advanced Server)	Description
pgagent	pgAgent/bin	dbserver/bin	Program executable file
pgagent.sql	pgAgent/share	dbserver/share/pgagent	SQL script to create the pgagent schema and its objects
pgagent.log	/var/log	Not presently created by default (give location and file name when starting pgagent)	Log file

**Note:** For Windows systems use the Event Viewer to check the Windows logs for pgAgent errors. Open Control Panel, Administrative Tools, Event Viewer, (then open Windows Logs if you are using Windows Vista), then open Application.

If pgAgent is installed on your database server, there is also a Jobs node in the pgAdmin Object Browser window:



**Note:** If you do not see the Jobs node, disconnect pgAdmin from the database server, then click the secondary mouse button on the database server node. Select Properties and make sure the Maintenance DB field shows the database with the pgagent schema (see the next step for identifying the pgagent schema). For default Postgres Plus installations the Maintenance DB field should contain postgres (edb for Oracle-compatible Advanced Server installations).

**Step 2:** Verify pgAgent is running.

Once pgAgent is installed, it is set up to automatically start whenever you start your computer. (For Advanced Server, pgAgent must be manually started.) pgAgent runs as a daemon on Linux and as a service on Microsoft Windows systems. The daemon or

service is named `pgagent`.

### For Linux only:

Verify the pgAgent daemon is running by using the following command:

```
ps aux | grep pgagent
```

This is shown by the following:

```
$ ps aux | grep pgagent
postgres 5399 0.0 0.1 6800 1732 pts/0 S 13:00 0:00
/opt/PostgresPlus/8.4SS/pgAgent/bin/pgagent -s /var/log/pgagent.log
hostaddr=127.0.0.1 port=5432 dbname=postgres user=postgres
user 5890 0.0 0.0 3064 728 pts/0 S+ 13:34 0:00 grep pgagent
```

If you need to start pgAgent, issue the following command as operating system user `postgres` (For Advanced Server use operating system user `enterprisedb`).

```
pgagent_home/pgagent -s pgagentlog connection_string
```

The directory containing the `pgagent` program is represented by `pgagent_home`. The full directory path to the pgAgent log file is represented by `pgagentlog`.

The connection string contains space-delimited, `libpq parameter=value` pairs. These parameters include the following:

- **dbname.** Name of the database in which the `pgagent` schema was created. (The `pgagent` schema is created for you in the pgAdmin *maintenance database* when pgAgent is installed. The maintenance database is the database to which pgAdmin is configured to open upon its initial connection.) Specify database `postgres` (`edb` for Oracle-compatible Advanced Server installations).
- **hostaddr.** Numeric IP address of the database server hosting the database specified by the `dbname` parameter.
- **port.** Port number on which the database server is listening for connections.
- **user.** Database user name used to create jobs. This database user must have `SELECT`, `INSERT`, `UPDATE`, and `DELETE` privileges on all tables in the `pgagent` schema. For Standard Server, specify `postgres`. For Advanced Server, specify `enterprisedb`.

**Note:** The `pgagent` schema is hidden from pgAdmin. Use the `psql` utility program to list the `pgagent` schema or see its contents.

```
$ psql -d postgres -U postgres
Password for user postgres:
psql (8.4.1)
Type "help" for help.

postgres=# \dn
```

```

List of schemas
Name          | Owner
-----+-----
information_schema | postgres
pg_catalog      | postgres
pg_toast        | postgres
pg_toast_temp_1 | postgres
pg_toast_temp_2 | postgres
pgagent         | postgres
public          | postgres
(7 rows)

```

**Note:** Do not specify the password in *connection\_string* as the password is then observable by other users. Instead, the password should be given in the Postgres Plus *password file* named `.pgpass`. The syntax of an entry in the password file is as follows:

```
hostname:port:database:username:password
```

An example password file entry is shown by the following:

```
localhost:5432:postgres:postgres:password
```

If one does not already exist, create the `.pgpass` file under the `postgres` user's home directory (`enterprisedb` user's home directory for Advanced Server). Be sure to set the file permissions so it is accessible only to the `postgres` user. This can be done using the `chmod` command as shown in the following example:

```
chmod 0600 ~/.pgpass
```

The following example shows how pgAgent is started:

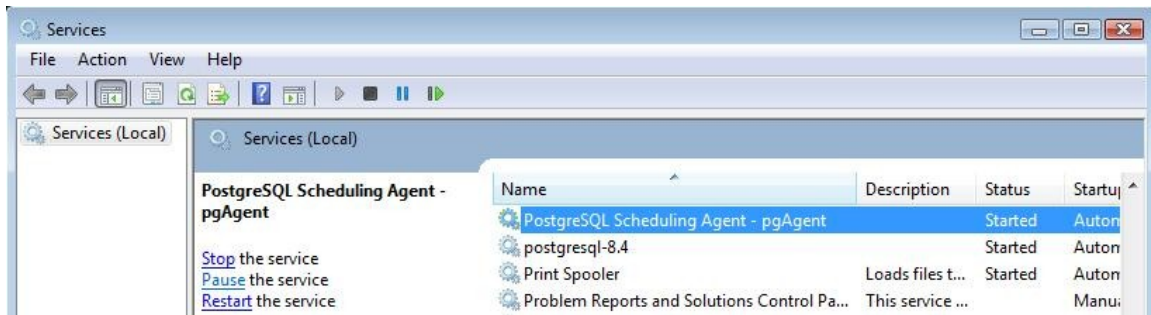
```

$ su postgres
Password:
$ /opt/PostgresPlus/8.4SS/pgAgent/bin/pgagent -s /var/log/pgagent.log
hostaddr=127.0.0.1 port=5432 dbname=postgres user=postgres

```

### For Windows only:

To verify the pgAgent service is running, open Control Panel, Administrative Tools, and then Services. The pgAgent service should be one of the started services.



If the pgAgent service is not running, click the Start link.

**Note:** As in the preceding description for Linux, a Postgres Plus password file must contain an entry for the maintenance database. In Windows systems the password file is %APPDATA%\postgresql\pgpass.conf where %APPDATA% is the postgres (enterprisedb for Advanced Server) Window user's application subdirectory. (The setting of %APPDATA% can be obtained by giving the command `echo %APPDATA%` in a Command Prompt window.)

### Step 3: Create a job.

In the pgAdmin Object Browser window, click the secondary mouse button on the Jobs node and select New Job. The New Job dialog box appears.

The dialog box contains the following fields:

- **Name.** Enter a name you want to assign to the job.
- **Enabled.** Leave the box checked if you want the job to run according to its schedules, otherwise de-select the check box.
- **Job Class.** Select a category from the drop-down list that describes the purpose of the job.
- **Host Agent.** Enter the hostname of a specific server running pgAgent if this job should run only on that server. If this field is left blank, any server may run the job (see the following note). This field is not normally needed for SQL-only jobs, but jobs with batch or shell scripts may need to be run on a specific server.
- **Comment.** Optionally, enter a comment describing the job.

**Note:** You can have pgAgent running on multiple machines. The pgAgent instances can be connected to a single maintenance database containing the `pgagent` schema. In this manner, jobs running on remote servers are defined in one central database. You use the Host Agent field to restrict which server a particular job can run on.

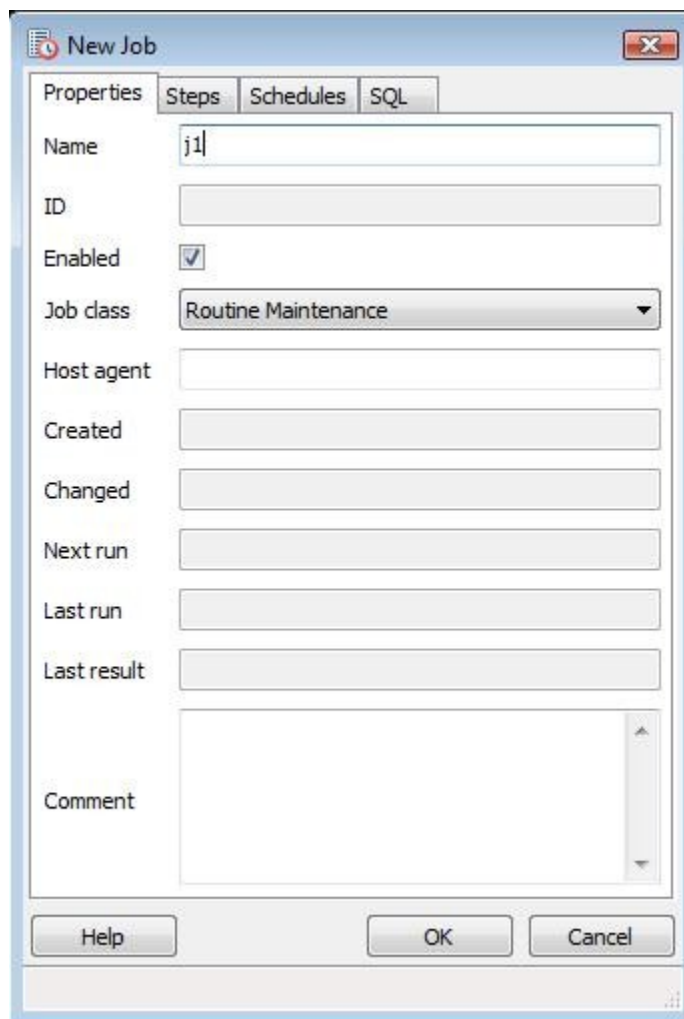
If you want to add such a restriction to a job, you need to determine the value you should enter in the Host Agent field. Make sure pgAgent is running on the target server and

connected to the maintenance database with the `pgagent` schema where you are creating the job.

Issue the following query to obtain the hostname that you should enter in the Host Agent field:

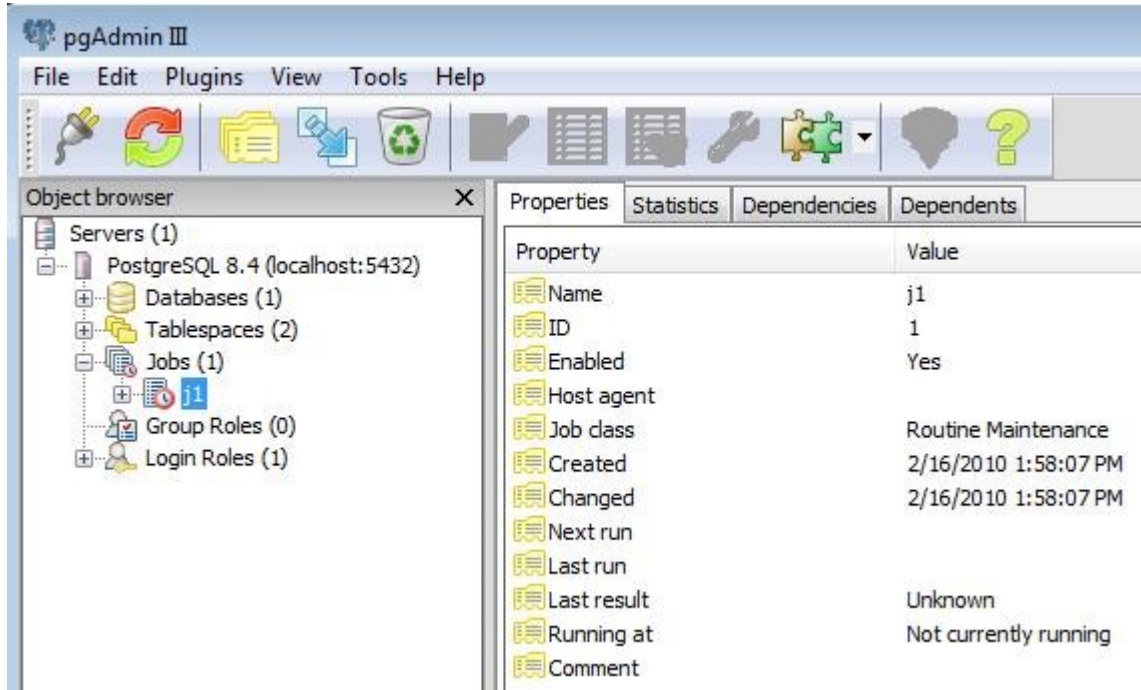
```
SELECT jagstation FROM pgagent.pga_jobagent;
```

The value returned by the query should be entered in the Host Agent field. This job will then only run when a pgAgent instance is started on that specified server with a connection string connecting it to the maintenance database in which you defined the job.



Click the OK button. A job node for the job you just created is added under the Jobs node.





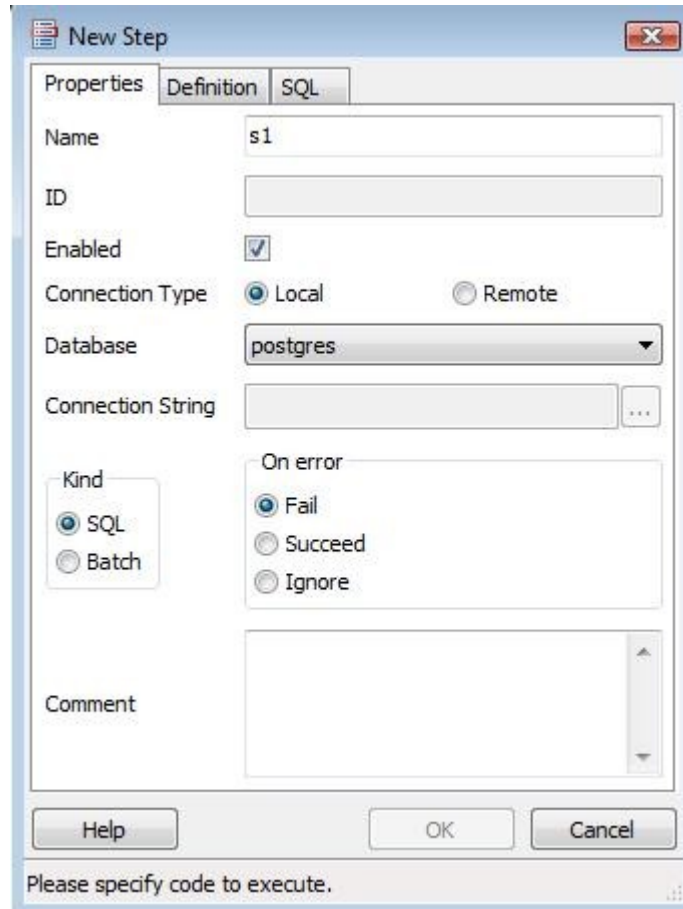
**Step 4:** Create one or more steps for the job.

Expand the job node you just created and click the secondary mouse button on the Steps node. Select New Step. The New Step dialog box appears.

The dialog box contains the following fields:

- **Name.** Enter a name you want to assign to the step. If you are going to define multiple steps, keep in mind that job steps are run in database-sort order according to step name.
- **Enabled.** Leave the box checked if you want the step to run whenever the job runs, otherwise de-select the check box.
- **Kind.** Select SQL if the step is to consist of SQL statements. Select Batch if the step is to consist of batch or shell commands.
- **Connection Type.** (Applies only to SQL steps.) Select whether the SQL step is to run on a local or remote database.
- **Database.** (Applies only to SQL steps on a local database.) Select the database on which the SQL step is to run from the drop-down list.
- **Connection String.** (Applies only to SQL steps on a remote database.) Browse and select the remote database on which the SQL step is to run from the drop-down list, or enter the connection string to be used to connect to the remote database.

- **On Error.** Select the action to be taken if the job step fails.
- **Comment.** Optionally, enter a comment describing the step.

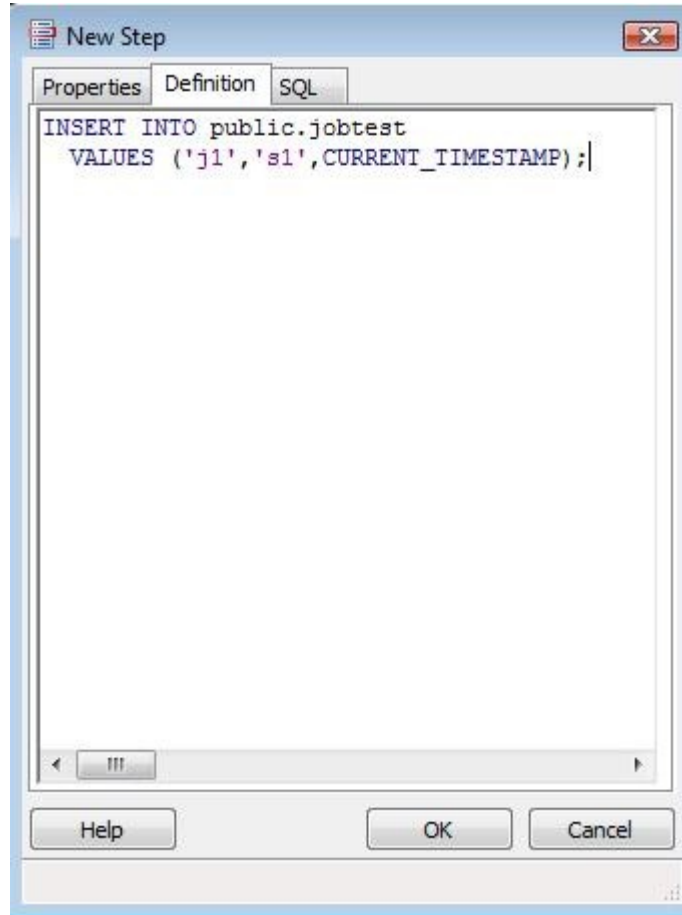


Click the Definition tab. Enter one or more SQL statements if this is a SQL step. Enter one or more batch/shell commands if this is a Batch step.

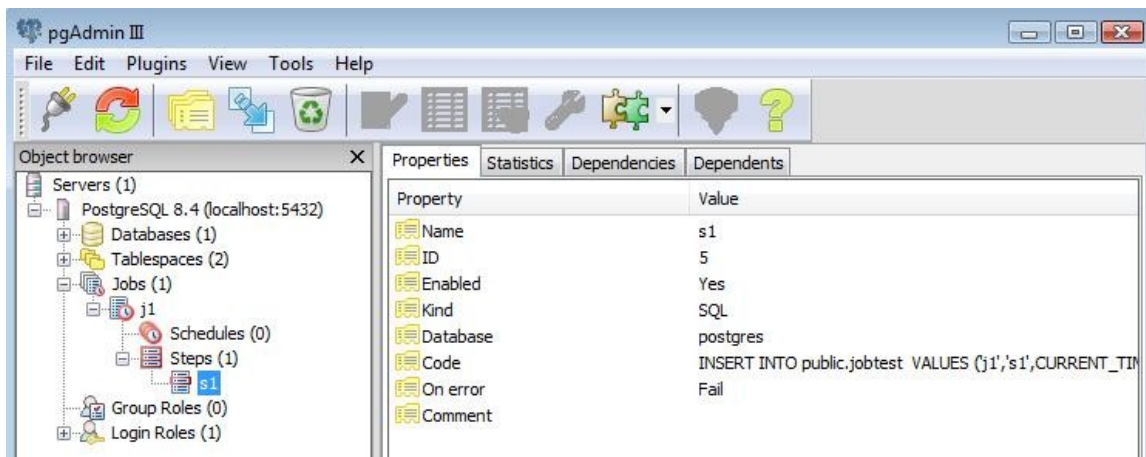
For a SQL step, create an entry in the password file allowing a connection to the database given in the Database field or the Connection String field. For local databases, the connecting user is postgres (enterprisedb for Advanced Server). For remote databases, the connecting user is given in the Connection String field.

**Note:** For a Batch step running on Linux systems, make sure you specify an appropriate shell script interpreter on the first line such as `#!/bin/bash`.

Click the OK button after you have added the SQL statements or batch/shell commands.



A step node is added under the job node.

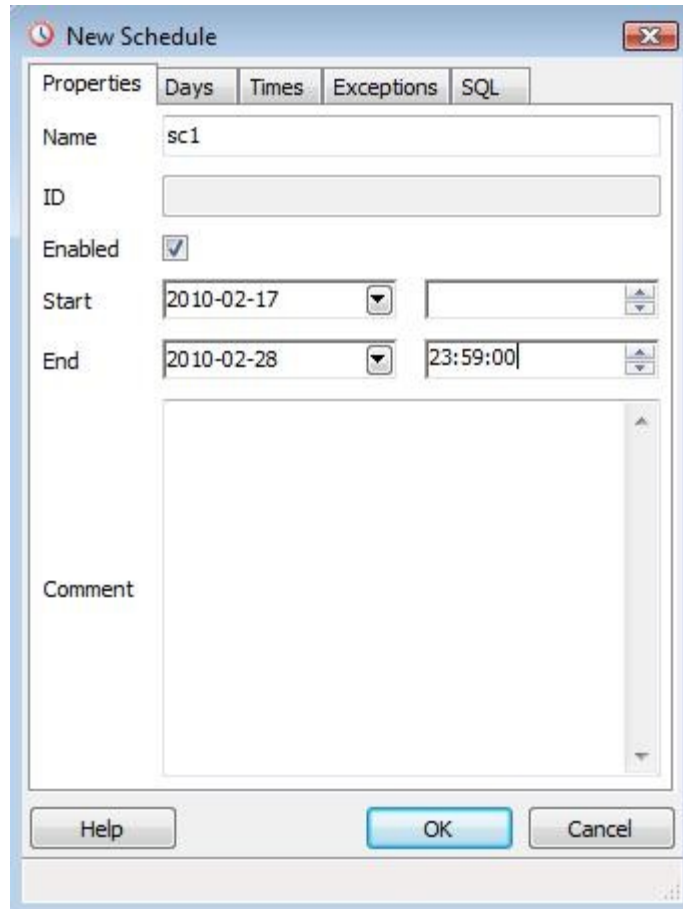


### **Step 5:** Create one or more schedules.

Under the job node you created, click the secondary mouse button on the Schedules node. Select New Schedule. The New Schedule dialog box appears.

The dialog box contains the following fields:

- **Name.** Enter a name you want to assign to the schedule.
- **Enabled.** Leave the box checked if you want the job to run according to this schedule, otherwise de-select the check box. A job may have multiple schedules in which case the job runs in accordance with all enabled schedules.
- **Start.** Select the date on, or after which the job is to start running according to this schedule. Optionally, enter a time for the selected date.
- **End.** Select the date after which the job will no longer run according to this schedule. If omitted, the job will run indefinitely on this schedule unless the schedule or the job itself is marked as disabled. Optionally, enter a time for the selected date.
- **Comment.** Optionally, enter a comment describing the schedule.



The screenshot shows a 'New Schedule' dialog box with the following fields and values:

- Name: sc1
- ID: (empty)
- Enabled:
- Start: 2010-02-17
- End: 2010-02-28 23:59:00
- Comment: (empty text area)

Buttons at the bottom: Help, OK, Cancel.

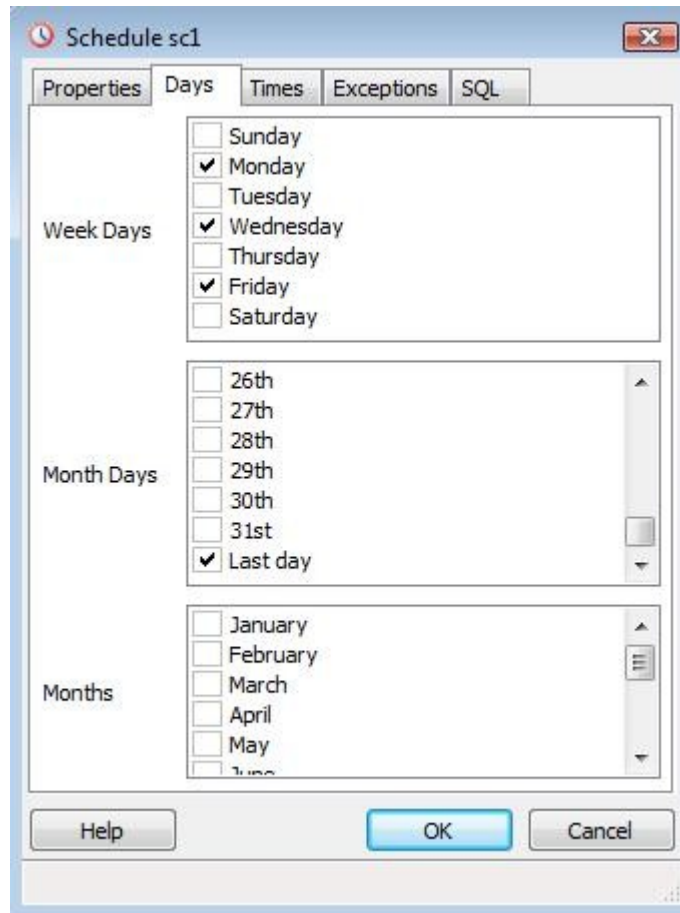
Click the Days tab.

**Note:** If you click the OK button and save the schedule at this point, the job is scheduled to run at approximately one to two minute intervals throughout every day and every month between your job start and end dates.

You can restrict the days and months when the job runs using the following fields:

- **Week Days.** Check the boxes for the days of the week you want the job to run. If you make no selection, the job runs every day unless you make one or more selections in the Month Days field.
- **Month Days.** Check the boxes for the days of the month you want the job to run. (Note there is a check box at the bottom of the selection list if you want to run the job on the last day of each selected month.) If you make no selection, the job runs every day unless you make one or more selections in the Week Days field.
- **Months.** Check the boxes for the months you want the job to run. If you make no selection, the job runs every month on the days you selected.

**Note:** The job runs on the days selected for Week Days and also on the days selected for Month Days. If no selections are made for Week Days and Month Days, the job runs every day of the month.



In this example the job is scheduled to run every Monday, Wednesday, Friday, and also on the last day of every month.

Click the Times tab.

**Note:** If you click the OK button and save the schedule at this point, the job is scheduled to run at approximately one to two minute intervals throughout the days you selected between your job start and end dates.

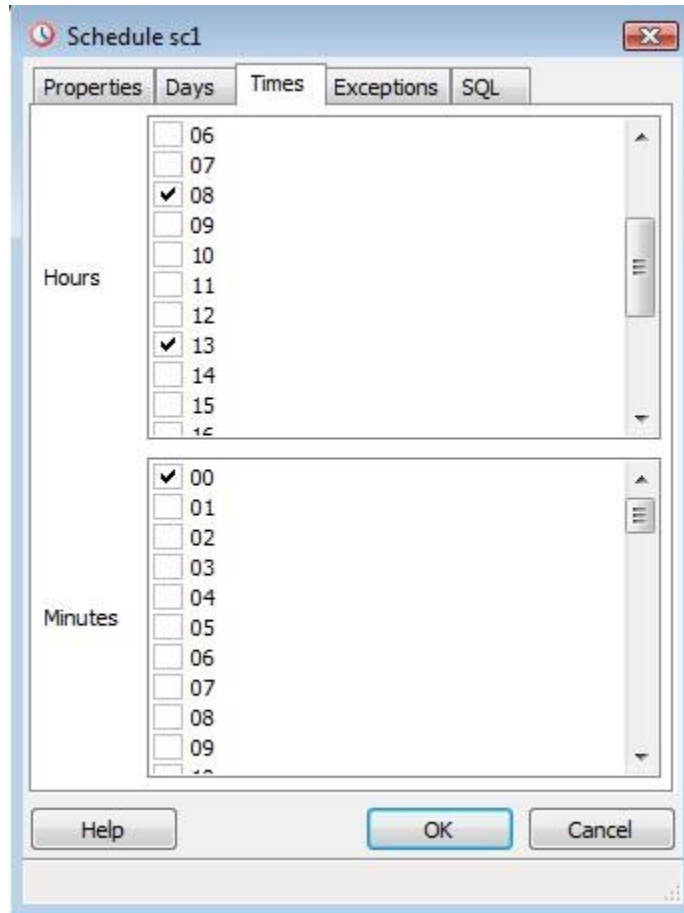
You can restrict the times during the day when the job runs using the following fields:

- **Hours.** Check the boxes for the hours of the day you want the job to run. If you make no selection, the job runs every hour at the minutes past the hour you select

in the Minutes field.

- **Minutes.** Check the boxes for the minutes past the hour you want the job to run. If you make no selection, the job runs at approximately one to two minute intervals throughout the hours you select in the Hours field.

**Note:** If you make no selections for Hours and Minutes, the job runs at approximately one to two minute intervals throughout the entire day.



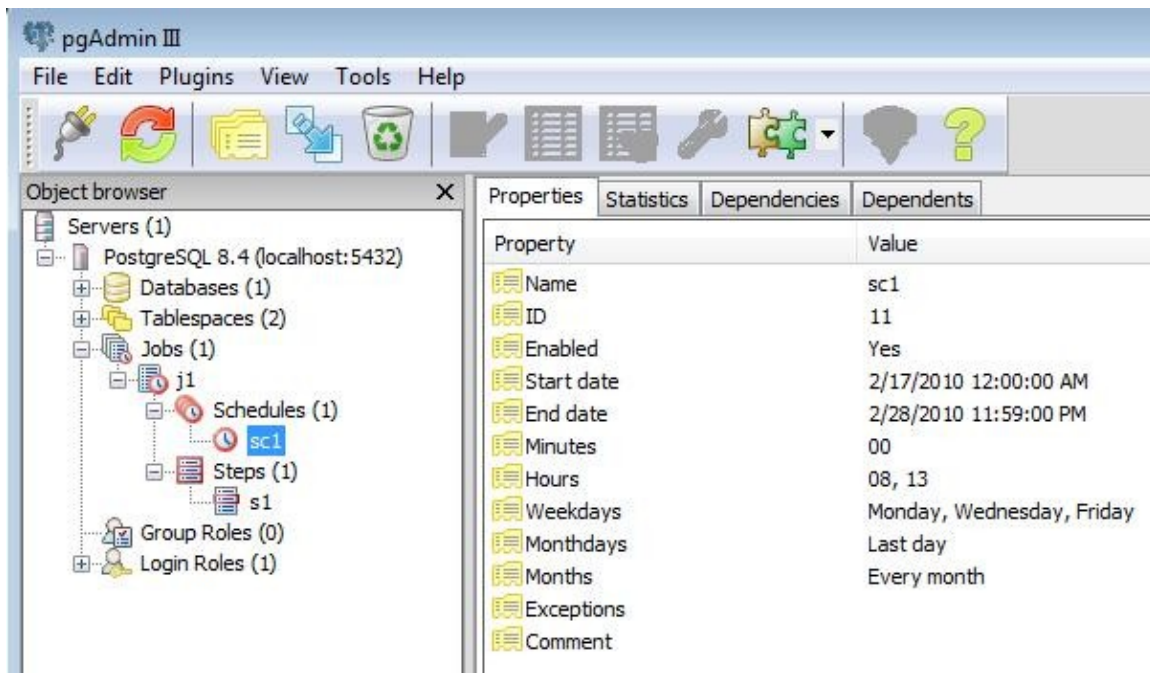
In this example the job is scheduled to run at 8:00 AM and 1:00 PM.

Click the OK button to save the schedule.

**Note:** You can add exceptions to the schedule by clicking the Exceptions tab and then adding the dates and times you do not want the job to run.

You can verify the schedule you created by examining the Properties tab when the

schedule node is highlighted.



Your job is now complete and will run according to your schedule whenever pgAgent is running.

## Conclusion

In this Quick Tutorial you learned how to set up pgAgent on a Postgres Plus database.

You should now be able to proceed confidently with a Technical Evaluation of Postgres Plus knowing that you can schedule simple to complex jobs on your Postgres Plus database.

The following resources should help you move on with this step:

- [Postgres Plus Technical Evaluation Guide](#)
- [Introduction to Postgres Administration Training](#)
- [Postgres Plus Getting Started resources](#)
- [Postgres Plus Quick Tutorials](#)
- [Postgres Plus User Forums](#)
- [Postgres Plus Documentation](#)
- [Postgres Plus Webinars](#)