Using PostgreSQL on Applicable Sun Cobalt™ Servers

1 What is PostgreSQL

PostgreSQL is a sophisticated Object-Relational Database Management System (DBMS). PostgreSQL is fully SQL-compliant. You can use PostgreSQL for any application that requires a database system, for example, ecommerce or for the back-end information storage.

Sun Cobalt™ servers use PostgreSQLto administer the interface to store information about users, virtual sites, and Active Monitor statistics.

For more information on PostgreSQL, go to `http://www.postgresql.org/` for documentation and the mailing-list archives. All the documentation for PostgreSQL is quite extensive and is present in HTML format in `/usr/doc/postgresql-7.0.2/`

Table of Contents

What is PostgreSQL **1**
Audience **1**
Applicable Products **1**
Accessing PostgreSQL **2**
Protecting your Data **2**
Password Protection **3**
Accessing the Database from a Network **4**

1.1 Audience

The audience for this technical note are developers who want to make use of the capabilities of PostgreSQL for their Sun Cobalt server applications.

1.2 Applicable Products

PostgreSQL is included on every product from the Sun Cobalt RaQ 3 server and later. PostgreSQL is used only by the administration interface on the Sun Cobalt RaQ 3, RaQ 4, and RaQ XTR server appliances (English and Japanese versions). The Sun Cobalt Qube 3 appliance does not use PostgreSQL for administration; it was replaced by the Sausalito architecture.

Important

Do **not** access the `Cobalt` database with PostgreSQL; you risk breaking the administration interface. You can also use the meta-verify script from `ftp://ftp.cobaltnet.com/pub/users/duncan/raq3/meta-verify.pl` if you need to restore the `Cobalt` database.

---

2 Accessing PostgreSQL

You can find PosgreSQL in these directories:

On the Sun Cobalt™ RaQ 3: `/home/pgsql`

On the Sun Cobalt™ RaQ 4 and XTR: `/var/lib/pgsql`

On the Sun Cobalt™t Qube 3: `/usr/lib/psql`

Important

Do **not** turn off PostgreSQL. It should be left running at all times for the interface to work.

---

3 Protecting your Data

Before resetting the database, you can check a few places and make sure the setup is appropriate. First, look in the PostgreSQL host access control file located at `/var/lib/pgsql/data/pg_hba.conf`

At a minimum, it should contain this uncommented line:
`local cobalt crypt`

This allows local programs to connect to the PostgreSQL back end via a UNIX local domain socket in `/tmp/.s.PGSQL.5432`

The next step is to make sure this file exists and is owned by the `postgres` user. If the above checks don't turn up anything unusual, then try resetting the database.

Note

This will **not** work on a Sun Cobalt RaQ 3 server.

On a Sun Cobalt™ RaQ 4 server, this is fairly simple. Follow these directions:

1) Stop the running PostgreSQL daemon. Type:

   ```
   /etc/rc.d/init.d/postgresql stop
   ```

2) Move the existing data directory out of the way. Type:

   ```
   mv /var/lib/pgsql/data /var/lib/pgsql/data.save
   ```

3) Restart the PostgreSQL daemon. Type:

   ```
   /etc/rc.d/init.d/postgresql start
   ```

The Sun Cobalt server should display output indicating that the database system is being initialized. Once this completes, it automatically adds the necessary `cobalt` database and tables from the `/etc/cobalt/postgres.db` file.

4) The next step is to reconstruct the tables from your system configuration. Type:

```
/usr/local/sbin/meta-verify -f
```

This command should also display some encouraging output about repairing invalid virtual sites and users, and hopefully marking each one *saved*. That should be all you need to do, but you can check that `swatch` is working by running:

```
/usr/local/sbin/swatch -d
```

Look for any warnings or errors. The Sun Cobalt server attempts to connect to the database first. and does not continue if it does not succeed.

If all is well with `swatch` and the Admin GUI, delete the saved data directory in `/var/lib/pgsql`

4 Password Protection

The PostgreSQL database admin password in pg_shadow is unrelated to the system password. It is generated when the database is initialized, and all this work is done by a script in `/usr/local/sbin/setup-postgres`, which in turn is called by the `initscript` if the directory `/var/lib/pgsql/data/` is missing.

Access to the database is controlled by the `pg_hba.conf` file. By default, it contains the line `local all trust`, which allows any local user to connect to any database without authentication. Also by default. the `postgres superuser` account has no password defined. This combination is particularly bad as it would allow anyone with shell access to administer the entire PostgreSQL setup.

Important

Sun Cobalt recommends that during initialization comment this line, which prevents the `postgres` user from logging in at all. Sun Cobalt recommends adding the `local cobalt crypt` line, which allows only local connections with password authentication to the `cobalt` database.

However, before this change is made, the `cobalt` database is created, and the SQL commands in `/etc/cobalt/postgres.db` are executed by the `postgres superuser.` This creates the admin account and the necessary tables used by the GUI and swatch. A password is generated using first 16 bytes from `md5sum` of random seed, and saved in `/etc/cobalt/.meta.id` for use by the `perl` module `Cobalt::Meta.pm` (for the GUI) and Active Monitor (`swatch`).

This file is mode `600` and owned by `root`, since both the admin web server and the `cron` daemon run as root. Fortunately, this whole process is only done the first time you power on the system, or if you choose to reset it by removing the data directory.

Sun Cobalt™ attempts to prevent users from making changes to PostgreSQL by disabling it to protect the administration system. The easy way to enable PostgreSQL is to add a line in

`pg_hba.conf` that reads `local template1 trust` (the `template1` database is created by default for the superuser). Then, you can connect. Type:

```
psql -U postgres template1
```

Perform any administrative tasks. Sun Cobalt recommends that you first assign the `postgres` account a password. Type:

```
ALTER USER postgres WITH PASSWORD 'password';
```

Change the newly added `pg_hba.conf` line to `local template1 crypt` to ensure that the database cannot be modified by local users. Alternately, you can change the password assigned on the `admin` account, in order to make `psql` access easier from the shell, rather than typing the 16 random chars. You can either use the `ALTER USER` command above and put the same value in.`meta.id`, or use the utility function provided that will take care of synchronizing them. Type:

```
perl -MCobalt::Meta -e 'Cobalt::Meta::setpw("password")
```

5 Accessing the Database from a Network

After setting up an appropriate `host ... line` in `pg_hba.conf,` you must start the daemon with TCP/IP connections enabled. There are many good examples in the file.

To do this, edit the `initscript` in `/etc/rc.d/init.d/postgresql` and change the line:

```
su postgres -c "nohup /usr/bin/postmaster >> /var/log/postgresql 2>&1 &"
```

to start postmaster with the `-i` option:

```
su postgres -c "nohup /usr/bin/postmaster -i >> /var/log/postgresql 2>&1
&"
```

If you run into any problems. check out the log file in `/var/log/postgresql` as it generally contains any errors. To save more debug information, edit the file:

`/var/lib/pgsql/data/pg_options` and increase the `verbose` or `query` values.