SMDR (Phone Switch Call Accounting)

Jason Healy, Director of Networks and Systems

Last Updated Mar 18, 2008

Contents

Chapter 1

SMDR (Phone Switch Call Accounting)

Last updated 2008/03/18

1.1 Introduction

Suffield Academy has its own PBX phone switch, and allows users to make direct local and long-distance calls from the phones on campus. We used to have a 3rd-party vendor who tracked and billed our long distance, but we stopped using them after we cancelled student long-distance accounts (the students were primarily using their cell phones).

In order to keep track of the basic call activity on our lines, we wrote a small script to parse the SMDR (**S**tation **M**essaging **D**etail **R**ecord) data that is output directly by our phone switch. It reads the SMDR data, parses it, and logs it to a SQL database for later analysis. Using simple SQL queries, we can run basic reports for total and per-number usage.

Our phone switch is a NEC NEAX 2400. The SMDR format is specific to this model of switch, but the general discussion of how it works should be roughly the same across other equipment.

1.2 Overview

1.2.1 Basic Operation

To log SMDR data from a phone switch, the following broad steps must be taken:

1. A SQL database (we use PostgreSQL) must be created to store the data.
2. One or more machines with serial ports must have our logging script installed on them (the script is written in Perl, and we use Debian Linux as our OS of choice).
3. The phone switch must be programmed to emit SMDR data on one or more of its serial ports (refer to the manual for the phone switch on how to enable this logging).

Once set up, the basic flow of information works like this:

1. The phone switch detects the termination of a call.
2. The phone switch emits Call Data Record (CDR) on all of its configured serial ports.
3. The machines running the logging script are waiting for serial input. They receive this input, parse it into individual fields, and insert the data into the SQL database.

Once in the database, administrators can run reports on the data using regular SQL queries.

1.2.2 Requirements

Our phone switch logs SMDR data over a slow (9600 baud) serial link. We do not experience a heavy call volume, so the amount of data is very small. Additionally, the phone switch automatically buffers the data on the serial ports, so in the unlikely event of the processing system being unable to keep up with the data flow, it will be buffered until the system can catch up.

Because the amount and rate of data is so small, almost any machine will do for processing the data. We run 3 Cobalt RaQ servers (200MHz MIPS processors) as our processing machines, though almost any machine will do (our previous processors were 486-class machines).

The script is designed to work with multiple processors in simultaneous operation. You can configure the phone switch to emit SMDR data on multiple serial ports, and then connect the ports to different machines for redundancy. The script will properly detect records that have already been inserted by another processor, and not duplicate the records. To help determine if the processors are running, a separate table tracks the records inserted by each processor so that you can ensure that all the machines are working properly.

The processing script is written in Perl, so you must have Perl installed on the processing machine. Additionally several common modules are required by the script:

- Sys::Hostname (built into recent versions of Perl)
- Time::Local (built into recent versions of Perl)
- Data::Dump (built into recent versions of Perl)
- POSIX ('setsid', should be available on POSIX-compliant systems)
- DBI (for inserting data to a SQL database)
- DBD::Pg (for communicating with a PostgreSQL server)
- Device::SerialPort (for listening to the serial port)

All of the modules, if not already built in to your system, are freely available on CPAN. On the "Etch" flavor of Debian Linux, all modules are already installed, or available directly via APT.

1.3 Running the Code

This section gives a (very) brief overview on setting up the processing systems, along with an even briefer overview of the code structure. Note that the script itself is well-commented, and should be easily changed by someone with a basic understanding of Perl.

1.3.1 Setting Up the Database

We have included a SQL file which defines the basic tables. It's written for the PostgreSQL database engine, though it could be made to work with any SQL-compliant database with very simple modifications. You can download the SQL here:

nec_smdr.sql

One table tracks all of the call data, and the other tracks the processing machines that inserted each record. To prevent duplicate records in a multi-processor environment, a UNIQUE constraint is placed on the call table. In the event that a processor attempts to insert a duplicate record, it will detect the duplicate on insertion and instead merely log that it too would have inserted a particular record. This information is stored in the "loggers" table, where the ID of each call record is listed with all the processors that processed the record.

1.3.2 Setting the Script to Run

First, download the actual script:

nec_smdr daemon Perl script

We suggest placing the script in `/usr/local/sbin`, though it can go anywhere. If you do change its location, be sure to update the init script below.

Download and install the package dependencies for the script.

Modify the script to include the correct serial port and database connection information.

Next, create the file `/etc/nec_smdr_postgres_password`, and put the database password inside it.

Next, download the init script, which starts and stops the daemonized Perl script:

nec_smdr init script

This is a Debian-based init script, and should work without modification on that platform. If you don't have Debian, modify it as necessary.

Link the init script into your `rc.d` directories as appropriate. Under Debian, the `update-rc.d` utility can help you with this.

To test the script, you can run it directly with the `-D` flag to run in debugging mode, *e.g.*:

```
/usr/local/sbin/nec_smdr -D
```

Instead of daemonizing, the script will run in the foreground and provide extra information as call records are processed.

If your serial port is hooked up, you should see debugging information about the calls as they are processed. Confirm that the calls are being logged to the database.

If everything is working correctly in debug mode, quit and issue:

```
/etc/init.d/nec_smdr start
```

To launch the daemonized version. The program is now running in normal mode and processing calls. You're all done!

==== Some Hints About the Script ===

The processing script has three major phases in it:

1. Wait for data on the serial line, and accumulate it in a buffer.
2. When a full record has been accumulated, parse it into individual fields.
3. Form the fields into a SQL record and insert it in the database (detecting duplicates if necessary)

If you need to modify the script to understand a different record format, you'll need to modify the record parsing section, and possibly the SQL section (if the number or names of the fields change).

The script has a very well-commented section on the call record format that it understands. Additionally, the script will log any lines it receives that it does not understand, so you can quickly debug the format of the lines.

If you're having trouble with the script, be sure to run it in debug mode (with the `-D` flag). This mode prevents the script from disassociating from the terminal, and provides much more verbose logging of its activity. It's very helpful for seeing what's going on during processing.