# Administrator's Guide
for SaberNet DCS 2.0

Title:	SaberNet DCS Administrator's Guide
Author:	Seth Remington <sremington@saberlogic.com>
Author:	Matthew Ranostay <mranostay@saberlogic.com>
Date:	2013-02-19
Revision:	1.8.2.8
Description:	Documentation for properly administrating an enterprise SaberNet DCS deployment.

Contents

Pyro Name Server and Event Server

PYRO consists of two parts, a server side, and client side. The server side is made up of an 'Event Server' and a 'Name Server'. The client side is of an application(i.e like 'sndcs_gtk'), which PYRO refers to these as 'Subscribers'. The 'Name Server' listens on a subnet/network segment for an 'Event Server' to register to it. It is import to note that ONLY ONE 'Name Server' can run on subnet at any given time, so chose wisely which host runs it. One or more Event Servers can be run on the same network segment, usually one per namespace(NOTE: See below for reference). Applications then can subscribe to the namespace(s) via the 'Name Server' using network broadcasts or a direct connection. When an client sends a event to the namespace, the 'Event Server' pushes it to all the clients. The events can be any object, except sockets and file descriptors.

For more in-depth discussion on PYRO, visit http://pyro.sourceforge.net/manual/PyroManual.html

SaberNet DCS requires an instance of the Pyro name server and event server to be running somewhere on the network. For a basic installation it is easiest to allow the SaberNet server (sndcsd) to automatically start an instance of the name and event servers, which it will do by default. In certain circumstances it may be desirable to run a separate instance of the name and event server. For instance, if you are running other Pyro based programs you would probably want to be running a separate shared instance of the name and event server. Another good reason to run a separate instance is that otherwise, if the sndcsd daemon is restarted all of the clients will need to be restarted as well because the heartbeat would be lost if the name/event server was restarted. For more information on this see Starting the Pyro Name and Event Server Separately From the DCS Server

To automatically start the name/event server answer 'Y' to the following prompt during the configuration:

```
Do you want to automatically start the pyro Name Server? [Y/n]:
```

If you answer 'N' you will be prompted to specify the host that the name/event server is running on:

```
What host is the Pyro Name Server running on (empty string means use broadc
```

Enter the host that the name server is running on. Alternatively you can leave it blank and just hit <Enter>. This will cause Pyro to broadcast for the name server. This has certain advantages but also add a bit of overhead as well.

Starting the DCS Server Daemon

The DCS server is named sndcsd.

Quick Start

Linux Run the command:

sndcsd

Win32 If you installed from source the 'sndcsd' server should have been installed in the "Scripts" directory in your Python installation. For example if your Python directory is at C:Python25 then the sndcsd script will be at C:Python25Scriptssndcsd. You can start it with the following command:

python C:Python25Scriptssndcsd

Alternativly you can install the DCS server to run as a Windows service with the following command:

python C:Python25Scriptssndcsd_service.py install

And remove it with:

python C:Python25Scriptssndcsd_service.py remove

Less Quick Start

Linux For a production installation you will want to start the DCS Server with an init script. There is currently an init script for Debian based distros in the scripts directory.

1. Copy rc.d-sndcsd-debian into your /etc/init.d directory
2. Rename it to something more appropriate like sndcsd
3. update-rc.d sndcsd defaults
4. Copy rc.d-sndcsxmlrpc-debian into your /etc/init.d directory
5. Rename it to something more appropriate like sndcsxmlrpc
6. update-rc.d sndcsxmlrpc defaults

Starting the DCS Web Server

SaberNet DCS Web has a built in web server thanks to Webware. It can also use other web servers like Apache when a more robust solution is needed. (See the documentation at the Webware project for instructiona on how to do this.) (TODO: Provide instructions here.) For testing or simple installations though, the built in web server should work fine.

Quick Start

To start the built in web server and DCS Web use the following command:

Linux

```
/usr/share/sndcs/web/AppServer
```

It must be started by a user with write privlidges in the directory where AppServer resides.

Win32 If you installed from source the DCS Web server should have been installed in your Python installation. For example if your Python directory is at C:Python25 then the DCS Web server script will be at C:Python25sharesndcswebLaunch.py. You can start it with the following command:

```
python C:Python25sharesndcswebLaunch.py
```

Alternativly you can install the DCS server to run as a Windows service with the following command:

```
python C:Python25Scriptssndcs2httpd_service.py install
```

And remove it with:

```
python C:Python25Scriptssndcs2httpd_service.py remove
```

If you need to configure the host or port that the web server runs on you can change it in the <Python Directory>/share/sndcs/web/Configs/AppServer.config file.

Less Quick Start

Linux For a production installation you will want to start the DCS Web Server with an init script.

1. Change into the /etc/init.d directory (i.e. cd /etc/init.d)
2. Create a symbolic link in your /etc/init.d directory to the /usr/share/sndcs/web/sndcs2httpd file. (i.e. ln -s /usr/share/sndcs/web/sndcs2httpd sndcs2httpd)
3. update-rc.d sndcs2httpd defaults

The default user account for DCS Web is: user: administrator passwd: admin

Logging

The logging configuration is in ’/etc/sndcs/logging.conf’ for POSIX systems, or for Windows systems its sys.prefix + ’\etc\sndcs\logging.conf’ *(NOTE: Usually C:\Python24 unless changed when installed)*. To enable/disable a logging function change the ’keys’ parameter under the ’formatters’ section. Also to change the debuging level for a function, first find its section and then change the ’level’ parameter. The possible choices in order of the most verbose output are DEBUG **(NOTE: Use this if you plan to send in a bug report)**, INFO, WARNING, ERROR, and CRITICAL.

Namespace

PYRO allows multiple namespaces, so you could have multiple instances of DCS pointing at different databases *(i.e More than one company shares a network)*. By default DCS uses "sndcs" as it default namespace. If you wish to change the namespace please do the following:

1. Open the ’sndcsd.conf’ file located in ’/etc/sndcs’ for POSIX systems, or for Windows in sys.prefix + ’\etc\sndcs\’.

2. Edit the ’namespace’ parameter to what you want, as long as it’s not conflicting with another namespace.

3. Edit all the clients ’sndcs2.conf’ files *(NOTE: Located in the same directory as above)* ’namespace’ parameter to point to one defined above.

Starting the Pyro Name and Event Server Separately From the DCS Server

In certain circumstances it may be desirable to start the Pyro "Name Server" and "Event Server" manually instead of letting the DCS Server do it for you. For instance, if you are running other Pyro based programs you would probably want to be running a separate shared instance of the Name and Event server. Another good reason to run a separate instance is that otherwise, if the sndcsd daemon is restarted all of the clients will need to be restarted as well. The reason for this is that restarting the DCS Server will restart the Pyro Name and Event servers, cutting off all communication between the clients and the server. Even if the server is restarted the client doesn’t currently rebind to the server automatically and will subsequently need to be restarted. Setting this up is not difficult, it just requires a little extra administrative effort and baby sitting.

The first step is to configure the DCS Server to *not* start the Pyro Name and Event servers automatically. Set the start_name_server configuration option to "False" in the sndcsd.conf config file.

```
[pyro]
start_name_server = False
```

Now we need to set up our system to start the Pyro Name and Event servers manually. This part will be dependant on what platform you are running the server on. The most common situations are described below. For much more detailed information please see the Pyro documentation on the Name Server and the Event Server.

Linux

Windows

TODO

Plugins

For information about plugins please refer to the Plugin Guide

Barcode Format

SaberNet DCS attempts to be as efficient as possible when in barcode mode. Because of this the barcodes themselves need to contain more than just the basic data. There should also be control characters suronding the data. An example will make it clear:

```
$305|s$   <----- barcode data
| |  |||
| |  ||--> Main selection signifier indicating the end of the barcode
| |  |---> The 's' is the clock in/out and selection signifier
| |  ----> The '|' is the separator signifier... it separated the domain spe
| ------> This is the employee's number (305)
--------> Main selection signifier indicating the start of the barcode
```

The following table defines all of the available signifiers. The signifiers will be configurable in a future version of SaberNet DCS but for now they are hardcoded.

Main Signifiers (these begin and end the barcode)	
Code	**Descroption**
$	Selection
*	Action
@	Multi-function (see Multi Function Barcode)

Separator Signifier (separates data from commands)	
Code	**Description**
|	Separator Signifier

Selection Signifiers	
Code	**Description**
s	Clock in/out and selection
l	Lunch in/out
b	Break in/out
r	Resume

... continued on next page

Selection Signifiers	
Code	**Description**
o	Clock out

Action Signifiers	
Code	**Description**
i	Indirect activity
p	Production activity
s	Setup Activity

Multi Function Barcode

The special multi-function barcode signifier (@) allows you to combine a select and an action into a single barcode swipe. If the employee is not currently clocked in it will also clock them in as well as select them. It currently only supports indirect activity actions. Here's an example:

```
@305|100|i@   <----- barcode data
| |  | |  |||
| |  | |  ||--> Multi-function signifier indicating the end of the barcode
| |  | |  |---> The 'i' is the indirect activity action signifier
| |  | | ----> The second '|' separates the action data from the signifier
| |  | ------> This is the action data (i.e. indirect activity 100)
| | ----> The first '|' separates the selection data from the action data
| ------> This is the employee's number (305)
--------> Multi-function signifier indicating the start of the barcode
```

Scanning the above barcode would clock employee 305 onto indirect activity 100 (additionally clocking the employee in as well if they were not already).

Commands

SaberNet DCS has several keyboard commands. All commands start with the forward slash '/' followed by the command. NOTE: There is no separator or ending character. The available commands are:

Command	Description
quit	End the application
exit	End the application
about	Show the about box
version	Show the about box
refresh	Reconnect to the server and refresh all data
reload	Reconnect to the server and refresh all data
reconnect	Reconnect to the server and refresh all data

Client GUI Interaction Commands

In order to allow some basic interaction with the SaberNet DCS client using barcode scans there are client interaction commands that mimic clicking on buttons with a mouse. As an example, instead of clicking on the "Start Gang" button on the selected employee screen you could instead scan a barcode that would perform the same function. All client GUI interaction commands start with the caret character '^' followed by the command. NOTE: There is no separator or ending character. The available commands are:

Command	**Description**
single_job	Switch to the "Single Job" screen
job_gang	Switch to the "Job Ganging" screen
setup_job	Switch to the "Setup" screen
setup_gang	Switch to the "Setup Ganging" screen
indirect	Switch to the "Indirect" screen
end_activity	Switch to the "End Activity" screen
information	Switch to the "Information" screen
break	Simulate pressing the "Break In/Out" button on the selected employee screen
lunch	Simulate pressing the "Lunch In/Out" button on the selected employee screen
clock_out	Simulate pressing the "Clock Out" button on the selected employee screen
cancel	Simulate pressing the "Cancel" button on whatever screen is current (if applicable)
ok	Simulate pressing the "OK" button on whatever screen is current (if applicable)
escape	Simulate pressing the 'Escape' key on the keyboard to reset the screen

Tips

- Since all timestamps are based on the system time of the box that the DCS Daemon (sndcsd) is running on it is generally a good idea to run ntp on that box as well to keep the time synced.

Hosted by SourceForge