Plugin Guide
for SaberNet DCS 2.0

Title:	SaberNet DCS Plugin Guide
Author:	Seth Remington <sremington@saberlogic.com>
Date:	2006-08-02
Revision:	1.2
Description:	Documentation for installing and creating plugins for SaberNet DCS

Contents

Events, Subscribers, and Publishers

SaberNet DCS uses events to communicate between the server, clients, and plugins. It does so by means of the Pyro Event Server (for more detailed information please see the Pyro documentation. TODO: provide link) A Publisher is the process that generates the events and the Subscriber is the process that consumes the events. The Publishers do not know or care who is consuming the information and the Subscribers do not know or care who generated the information.

Each event contains pieces of data related to that particular event. For example: the clock in event contains the employee's serial number, ID, name, the timestamp of when they clocked in, and the name of the terminal they clocked in on. The data is sent in the form of a Python dictionary.

Channels and Namespaces

Event Types

The following list is the currently available event types and the data that is sent. Remember that all event channel names are preceeded with the namespace (i.e. if the namespace is "sndcs" the clock in event would be sndcs_clock_in).

Event Name	Data Key	Data Value
clock_in	serialNum	Employee's serialNum
	empId	Employee's employee ID
	properName	Employee's proper name (i.e. lastName, firstName)
	startStamp	Clock in date/time
	startTerminal	The name of the terminal the employee clocked in on
clock_out	serialNum	Employee's serialNum
	empId	Employee's employee ID
	properName	Employee's proper name (i.e. lastName, firstName)
	endStamp	Clock out date/time
	endTerminal	The name of the terminal the employee clocked out on
break_in	serialNum	Employee's serialNum
	empId	Employee's employee ID
	properName	Employee's proper name (i.e. lastName, firstName)
	startStamp	Break in date/time
	startTerminal	The name of the terminal the employee used
	activity	A dictionary of data about the activity being started
		serialNum: Will always be the serialNum of the BREAK indirect activity
		indirect: Will always be the code of the BREAK indirect activty
		description: Will always be the description of the BREAK indirect activty
break_out	serialNum	Employee's serialNum
	empId	Employee's employee ID
	properName	Employee's proper name (i.e. lastName, firstName)
	startStamp	Break out date/time
	startTerminal	The name of the terminal the employee used
	activity	A dictionary of data about the activity being started
		serialNum: Will always be the serialNum of the BREAK indirect activity
		indirect: Will always be the code of the BREAK indirect activty
		description: Will always be the description of the BREAK indirect activty
lunch_in	serialNum	Employee's serialNum
	empId	Employee's employee ID
	properName	Employee's proper name (i.e. lastName, firstName)
	startStamp	Lunch in date/time
	startTerminal	The name of the terminal the employee used
	activity	A dictionary of data about the activity being started
		serialNum: Will always be the serialNum of the LUNCH indirect activity
		indirect: Will always be the code of the LUNCH indirect activty
		description: Will always be the description of the LUNCH indirect activty
lunch_out	serialNum	Employee's serialNum
	empId	Employee's employee ID

Event Name	Data Key	Data Value	
	properName	Employee's proper name (i.e. lastName, firstName)	
	startStamp	Lunch out date/time	
	startTerminal	The name of the terminal the employee used	
	activity	A dictionary of data about the activity being started	
		serialNum	Will always be the serialNum of the LUNCH indirect activity
		indirect	Will always be the code of the LUNCH indirect activty
		description	Will always be the description of the LUNCH indirect activty
indirect_start	serialNum	Employee's serialNum	
	empId	Employee's employee ID	
	properName	Employee's proper name (i.e. lastName, firstName)	
	startStamp	Date/time the indirect activity is started	
	startTerminal	The name of the terminal the employee used	
	activity	A dictionary of data about the activity being started	
		serialNum	The serial number of the indirect activity being started
		indirect	The code of the indirect activity being started
		description	The description of the indirect activity being started

This maintainers of this document will try to keep the above data as up to date as possible but the final reference should always be the source code.

An Example

To clear up any confusion let's follow a real life example:

- An employee clocks in with the client.
- The client communicates with the server and executes the "employee clock in routine".
- The server sends out a "clock_in" event which contains the employee's serial number, ID, name, the timestamp of when they clocked in, and the name of the terminal they clocked in on.
- Any process that is subscribed to the "clock_in" channel receives the event and decides what to do with it. For example: the sndcs_gtk client would update it's list of active employees, showing the clocked-in employee as idle. But a custom plugin might compare the employee's clock-in time with the time of their shift start and if they are late it might pipe a strongly worded message through festival (Project Page: http://www.cstr.ed.ac.uk/projects/festival) and pump it over the shop floor loudspeakers. Thus strongly curtailing late arrivals ;)

Heartbeat

The *'sndcsd'* program sends out a **Heatbeat** event to the clients every 5 seconds. If a client doesn't receive four Heartbeat's in a row it throws an exception and exits, this is to prevent *'zombie'* terminals.

Creating Plugins

Web Plugins

Coming soon...

Server Plugins

Coming soon...

Client Plugins

Coming soon...