PHP ESL

Overview

Assuming PHP support for ESL was installed it is relatively simple to include the ESL Library and perform interaction with FreeSWITCH. It can be done by connecting to a running FreeSWITCH as well as having FreeSWITCH connecting to a socket application as needed. This is the same potential exposed to other languages due to the standardize SWIG bindings.

Below are some working examples with comments to help educate how to use the ESL via PHP.

Examples

single_command.php

included with FreeSWITCH tree

Example

```
#!/usr/bin/php

require_once('ESL.php');

if ($argc > 1) {
    array_shift($argv);
    $command = sprintf('%s', implode(' ', $argv));
    printf("Command to run is: %s\n", $command);
    $sock = new ESLconnection('localhost', '8021', 'ClueCon');
    $res = $sock->api($command);
    printf("%s\n", $res->getBody());
} else {
    printf("ERROR: You Need To Pass A Command\nUsage:\n\t%s <command>", $argv[0]);
}
```

The example above is designed to run from the local console using the default credentials for the event socket and run whatever command is based as an argument.

Example

```
$ ./single_command.php status
```

```
Command to run is: status

UP 0 years, 0 days, 2 hours, 37 minutes, 44 seconds, 733 milliseconds, 519 microseconds
0 session(s) since startup
0 session(s) 0/30
1000 session(s) max
```

Example of making an inbound connection from script to FS and execute a FSAPI command show channels.

Example

```
#!/usr/bin/php

require_once('ESL.php');
$command = "show channels";
$sock = new ESLconnection('localhost', '8021', 'ClueCon');
$res = $sock->api($command);
printf("%s\n", $res->getBody());
```

Example of making an inbound connection and listening for events and printing them serialized.

Example.

```
#!/usr/bin/php

<?php

require_once('ESL.php');

set_time_limit(0); // Remove the PHP time limit of 30 seconds for completion due to loop watching events

// Connect to FreeSWITCH
$sock = new ESLconnection('localhost', '8021', 'ClueCon');
// We want all Events (probably will want to change this depending on your needs)
$sock->sendRecv("event plain ALL");


// Grab Events until process is killed
while($sock->connected()){
$event = $sock->recvEvent();
print_r($event->serialize());
}

?>
```

Above example connects to the event socket and just prints the events as they are received until the process is stopped.

Example of an outbound socket connection where the call is answered, a variable is set then perhaps play one of the pre-installed files and hangup.

ivrd

fs_ivrd comes with freeswitch. It being a small daemon just invokes the script defined in a variable and passes data from it via STDIN/OUT

Since this is an outbound socket connections it needs to be defined in the dialplan.

Example.

```
<extension name="outbound-socket">
<condition field="destination_number" expression="^55(522)$">
<action application="set" data="ivr_path=/usr/local/freeswitch/scripts/ivrd-demo.php"/>
<action application="socket" data="127.0.0.1:8084 async full"/>
</condition>
</extension>
```

The above dialplan sample would invoke ivr-demo.php when 55522 is called as long as fs_ivrd is running. To start fs_ivrd:

/usr/local/freeswitch/bin/fs_ivrd -h 127.0.0.1 -p 8004

It takes 2 arguments -h for hostname and -p for port.

Example.

```
#!/usr/bin/php -q

<?php

// set a couple of things so we dont kill the system
ob_implicit_flush(true);
set_time_limit(30);

// Open stdin so we can read the data in
$in = fopen("php://stdin", "r");

// Connect
echo "connect\n\n";

// Answer
echo "sendmsg\n";
echo "call-command: execute\n";
echo "execute-app-name: answer\n\n";

// Play a prompt
echo "sendmsg\n";
echo "call-command: execute\n";
echo "execute-app-name: playback\n";
echo "execute-app-arg: /usr/local/freeswitch/sounds/en/us/callie/ivr/8000/ivr-welcome_to_freeswitch.wav\n\n";

// Wait
sleep(5);

// Hangup
echo "sendmsg\n";
echo "call-command: hangup\n\n";

fclose($in);

?>
```

ivrd will call this script for each call. All itdoes is answer the channel tell FreeSWITCH to play the “welcome to freeswitch” prompt. Since the script is now controlling all call flow I needed to add a wait or it would send the hangup immediately before the prompt was played.