Using PERL to Send XML scripts to an iLO Management Processor

Example 1: Building your own CPQLOCFG with PERL:

```
#!/usr/bin/perl
###########################################################################
##
## Simplified perl version of CPQLOCFG
## Copyright 2003 Hewlett Packard Development Company, L.P.
##
## To use this program, you must have Net::SSLeay and IO::Socket::SSL
## installed.  You may obtain these modules from http://www.cpan.org/
##
## You may use and modify this program to suit your needs.
##
###########################################################################

use IO::Socket::SSL;
use Getopt::Long;


sub usage
{
        print "Usage:\n";
        print "    locfg [-s server] [-l logfile] [-f inputfile]\n";
        exit 0;
}

###########################################################################
##
## Process options
##
###########################################################################

my $host, $logfile, $file, $verbose, $help;
$verbose = 0;
$r = GetOptions("server|s=s" => \$host,
               "logfile|l=s" => \$logfile,
               "input|f=s" => \$file,
               "verbose" => \$verbose,
               "help|?" => \$help
               );

if ($help || !$host || !$file) {
        usage();
}

if ($logfile) {
        # If a logfile is specified, open it and select it as the default
        # filehandle
        open(L, ">$logfile") || die "Can't open $logfile\n";
        select(L);
}

# Set the default SSL port number if no port is specified
$host .= ":443" unless ($host =~ m/:/);

# Open the SSL connection and the input file
my $client = new IO::Socket::SSL->new(PeerAddr => $host);
open(F, "<$file") || die "Can't open $file\n";

# Send the XML header and begin processing the file
print $client '<?xml version="1.0"?>' . "\r\n";
while($ln=<F>) {
        # Chomp of any EOL characters
        $ln =~ s/\r|\n//g;

        # Special case: UPDATE_RIB_FIRMWARE violates XML.  Send the full
        # UPDATE firmware tag followed by the binary firmware image
        if ($ln =~ m/UPDATE_RIB_FIRMWARE/i) {
                if ($ln =~ m/IMAGE_LOCATION=\"(.*)\"/i) {
                        $firmware = $1;
                        open(G, "<$firmware") || die "Can't open $firmware\n";
                        $len = (stat(G))[7];
                        print $client "\r\n<UPDATE_RIB_FIRMWARE
IMAGE_LOCATION=\"$firmware\" IMAGE_LENGTH=\"$len\"/>\r\n";
                        print "\r\n<UPDATE_RIB_FIRMWARE IMAGE_LOCATION=\"$firmware\"
IMAGE_LENGTH=\"$len\"/>\r\n" if ($verbose);
                        $x = read(G, $buf, $len);
                        print "Read $x bytes from $firmware\n" if ($verbose);
                        $x = $client->write($buf, $x);
                        print "Wrote $x bytes\n" if ($verbose);
                        close(G);
                        next;
                }
        }
        # Send the script to the iLO board
        print $ln . "\n" if ($verbose);
        print $client $ln . "\r\n" ;
}
close(F);

print "----\n" if ($verbose);

# Ok, now read the responses back from iLO
while($ln=<$client>) {
        last if (length($ln) == 0);

        # This isn't really required, but it makes the output look nicer
        $ln =~ s/<\/RIBCL>/<\/RIBCL>\n/g;
        print $ln;
}

# All done
exit 0;
```