

***KickOff* – Remote Exec Utility for Windows**

By Ruigang Yang, Nov 2001
ryang@cs.unc.edu

Introduction: *KickOff* is a window utility that allows executing applications remotely under the MS Windows Environment. It has two parts, a client and a server. The client sends commands to the server through a TCP/IP connection, and the server executes the commands on the machine that the server resides. The server could be installed as a Window service so it will be automatically started as the machine boots. *KickOff* is NOT a remote desktop application. It does not redirect input/output, it simply allows users remotely starting programs. For remote desktop access, please refer to the VNC program or the built-in remote desktop option available from Windows 2000 or Windows XP server edition.

Requirement:

- Operation System:
 - Server: Windows NT 4.0, Windows 2000, or Windows XP
 - Client: the MS Windows family. (A Linux port is also possible)

Installation: It is best to install the *KickOff* server as a window service. Using the following command to install it as a service (Administrator privilege is required)

```
kickoff portNumber -install
```

This will install the server, listening on `portNumber`, as a service. By default, the service will start automatically after reboot. You could also start/stop the service manually in the Window control panel, using the standard `services` applet. To remove the service, simply type

```
kickoff portNumber -remove
```

Client Usage: (*KickOff* Client is abbreviated as “kfc”)

Syntax: `kfc serverName portNumber [opCode] commandString`

To start a program, type the `commandString` with *full* path name. If there are spaces in the path, double quote the `commandString`. Here is an example to start the Mspaint program to open an image on a remote machine `foo`, assuming that the *KickOff* server is running on `foo`, listening to port 9999.

```
kfc foo 9999 "c:\WINNT\System32\mspaint c:\winnt\zaptec.bmp"
```

Kfc could also be used to terminate a program originally started with kfc. To kill a program, use the optional opcode “kill”. For example, if we want to stop the mspaint program we started in the above example, we could issue the following command:

```
kfc foo 9999 kill "c:\WINNT\System32\mspaint
c:\winnt\zaptec.bmp"
```

Furthermore, if we want to terminate all the programs, we can just use the kill opcode without any command string, as in the following example:

```
kfc foo 9999 kill
```

Tips:

- *Event Logging*: For security consideration, **KickOff** server logs every command through Windows’ event logging mechanism. The coming connection’s IP address, as well as the command string, is recorded in the Windows application event log, which can be viewed using the standard Event Viewer tool (under Administrative tools). Note that there are three different kinds of logs; system log, security log, and application log in the Event Viewer. You must select the application log to view *KickOff* events.
- *To run multiple instances of a same program*: The **KickOff** server maintains a list of active processes using the command string as the key. If two command strings are exactly the same, the information about the new process will overwrite the information about the previous one. When this happen, the previous process cannot be shutdown through the client. To circumvent this limitation, we could append white spaces to the end of the command string. So a new list entry will be created for each process.
- *Network access*: If the **KickOff** server runs as a service, it only has local account access privileges by default. To gain access to network neighborhood, you can run the service under a user account. This can be changed in Control Panel → Administrative Tools → Services. Double click the **KickOff** service to open the service setting dialog, click on the “Log On” Table, in which you can make the necessary adjustment.
- *Error Handling*: **KickOff** server only tries to run the commands it receives. It will only report errors if the command cannot be executed, usually means a bad path. It will not report the errors, if any, returned by the child process, nor does it redirect the standard input or output.