



Tip of the Week :

- using user debug functions with attachfec

[Does not apply to java servers
(sorry!)]

- Two ways to register command-line interface routines:
 - RegisterUserCommand(cmd,fcn, iparam, fparam, access)
 - Cmd is the

Does not apply to java servers (sorry!)

```
int RegisterUserCommand ( char *      cmd,  
                        int(*) (int, int, int, int) fcn,  
                        int *      iparam,  
                        float *    fparam,  
                        int         access  
                        )
```

Registers a user-defined command or variable which can be accessed via the TINE command line interpreter.

As a console application, a running tine server (or client) offers a variety of services at the command line (just type 'help'). For instance, you can turn debugging on or off, get the current server statistics, examine the connection tables, etc. This interface is available via the 'remote' accessor even if the server is running in the background. It is often frequently desirable to examine or alter server-specific variables or call a server specific routine. This can be partially achieved by making use of this registration function.

Parameters:

cmd is the command string to be parsed by the command interpreter.

fcn is an optional user defined function taking four integer arguments.

iparam is a pointer to an optional global integer variable.

fparam is a pointer to an optional global float variable.

access is a TINE access code (either CA_READ or CA_READ|CA_WRITE). To allow 'set' commands at the command line, you should include the CA_WRITE flag.

Note: Only one of *fcn*, *iparam*, or *fparam*, should be non-NULL when making this call. If the user types "get <cmd>" without arguments at the command line, then either *iparam* or *fparam* is displayed on the console, depending on which parameter was non-NULL at registration time. Likewise "set <cmd> = <val>" can set this variable to a value entered at the console. If a function is registered, then "get <cmd>(arg1,arg2,arg3,arg4)" will call the function registered with the arguments supplied. "set <cmd>(arg1,arg2,arg3,arg4) = <val>" will set all arguments equal to <val> and then call the function. If fewer arguments than 4 are supplied then the function will be called with '0' for the missing arguments.

A more reasonable command line function registration call is underway. However the routine so presented is generally sufficient for checking hardware readouts etc. and checking and setting the contents of global variables.

Returns:

0 if successful, otherwise a TINE completion code

```
int RegisterUserFunction ( char *      name,  
                        int(*) (int argc, char *argv[]) fcn  
                        )
```

Registers a user-defined function which can be accessed via the TINE command line interpreter.

As a console application, a running tine server (or client) offers a variety of services at the command line (just type 'help'). For instance, you can turn debugging on or off, get the current server statistics, examine the connection tables, etc. This interface is available via the 'remote' accessor even if the server is running in the background. It is often frequently desirable to examine or alter server-specific variables or call a server specific routine. This can be partially achieved by making use of this registration function.

Example 1: RegisterUserCommand()

Initialization routine from the central archiver:

```
void mexinit(void)
{
    DTYPE dout;
    char *data=NULL;
    int i, j, k, cc = 0;

    RegisterDeviceName(MEX_EQMNAME, "keyword", 0);

    remove("problems.log");
    RegisterUserCommand("ArchiveDebug", NULL, &archdebug, NULL, CA_READ+CA_WRITE);
    RegisterUserCommand("FilterTable", dumpFilterTable, NULL, NULL, CA_READ);
    RegisterUserCommand("ArchiveRecord", dumpArchiveRecord, NULL, NULL, CA_READ);

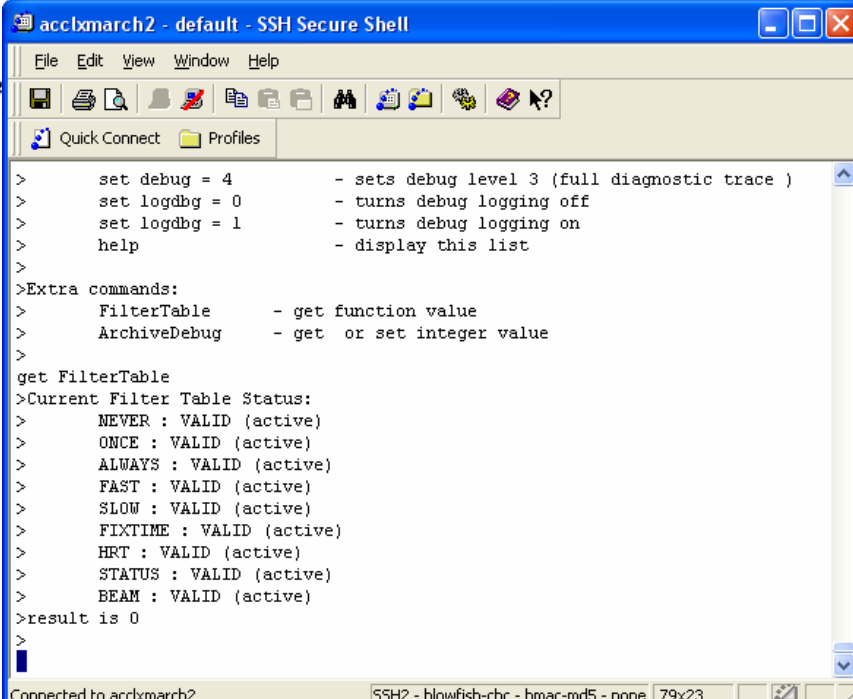
    daqInit(srvinfodbfile[0], srvinfodbfiles);
    nredirfiles = getRedirInfo();
    redirInit(redirfile, redirsrvc, nredirfiles);

    getNetGlobalList(GLOBALS_FILENAME);
    if (ngloballist > 0)
    { /* found a registered globals file */
        feclog("found %d globals from list %s", ngloballist, GLOBALS_FILENAME);
        for (i=0; i<nrecords; i++) for (j=0; j<daqTbl[i].nparam; j++)
            for (k=0; k<ngloballist; k++)
```

Example 1: continued

```
int dumpFilterTable(int a0,int a1,int a2,int a3)
{
    int i;
    a0 = a0; a1 = a1; a2 = a2; a3 = a3;
    ttyoutput("Current Filter Table Status:");
    for (i=0; i<nDaqFilterTblEntries; i++)
    {
        ttyoutput("\t%s : %s (%s)",
            daqFilterTbl[i].tag,
            daqFilterTbl[i].valid ? "VALID"
            daqFilterTbl[i].active ? "active"
        )
    }
    return 0;
}
```

Prints output onto stdout as well as any debug pipes!



```
acclxmarch2 - default - SSH Secure Shell
File Edit View Window Help
Quick Connect Profiles
> set debug = 4 - sets debug level 3 (full diagnostic trace )
> set logdbg = 0 - turns debug logging off
> set logdbg = 1 - turns debug logging on
> help - display this list
>
>Extra commands:
> FilterTable - get function value
> ArchiveDebug - get or set integer value
>
get FilterTable
>Current Filter Table Status:
> NEVER : VALID (active)
> ONCE : VALID (active)
> ALWAYS : VALID (active)
> FAST : VALID (active)
> SLOW : VALID (active)
> FIXTIME : VALID (active)
> HRT : VALID (active)
> STATUS : VALID (active)
> BEAM : VALID (active)
>result is 0
>
Connected to acclxmarch2 SSH2 - blowfish-cbc - hmac-md5 - none 79x23
```

Example 1: continued ...

e.g. the CDI
Hardware Server

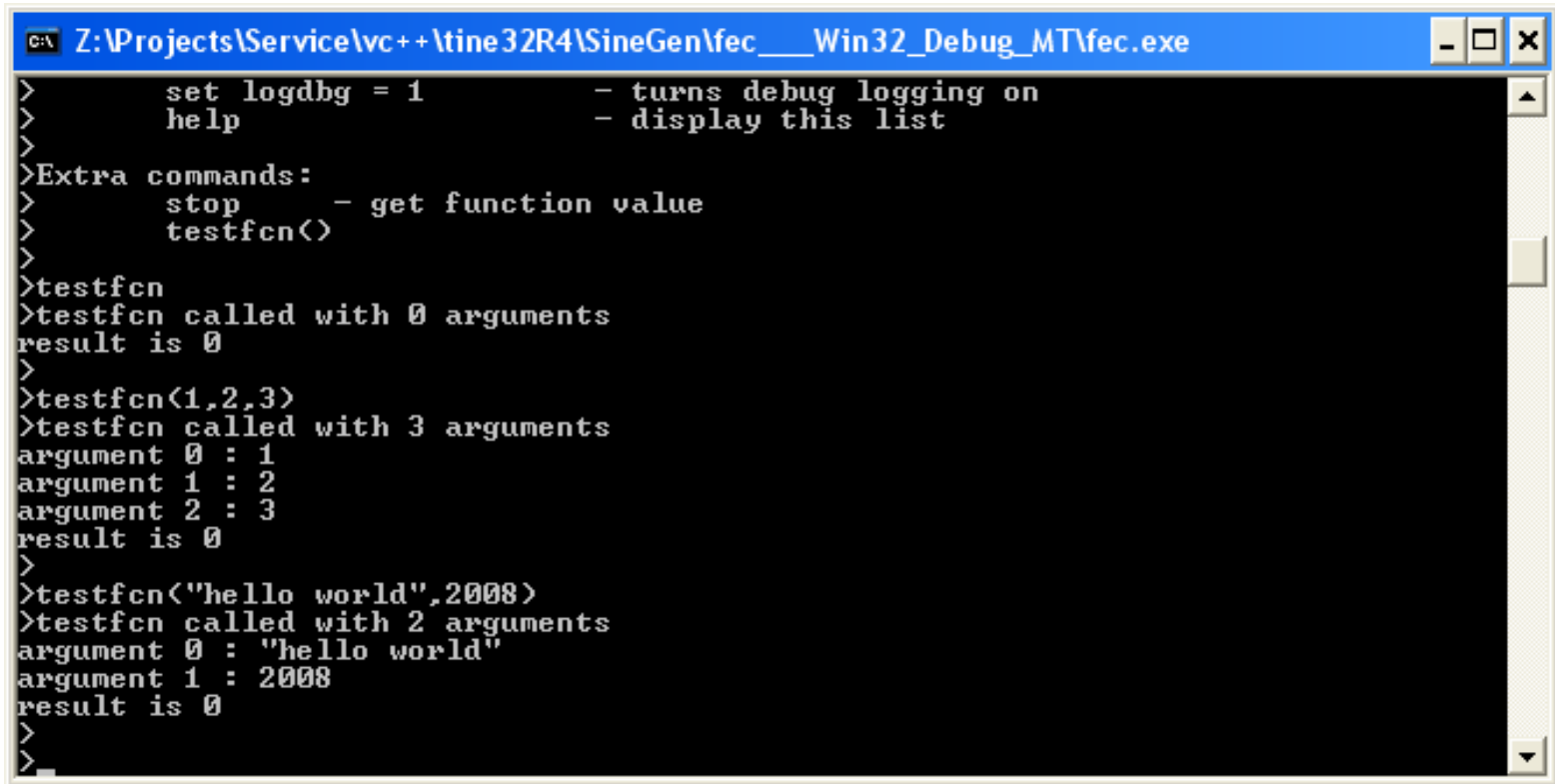
```
C:\WINDOWS\system32\cmd.exe - cdihdwsrv
> set logdbg = 1          - turns debug logging on
> help                   - display this list
>
>Extra commands:
>  cdddevices            - get function value
>  cdlisteners          - get function value
>  cdidebug             - get or set integer value
>
>get cdddevices
>bus line 0 : hashed (780 devices) @04.12.08 19:11:02.000 CET
>device 115 (hash 3, line 0 srv dev 316): M10AddPia.livSta
>device 104 (hash 3, line 0 srv dev 305): M2AdcPia.trgMod
>device 105 (hash 4, line 0 srv dev 306): M4AdcPia.trgMod
>device 106 (hash 5, line 0 srv dev 307): M10AdcPia.trgMod
>device 689 (hash 6, line 0 srv dev 1990): M16AddPia.addVal
>device 107 (hash 6, line 0 srv dev 308): M12AdcPia.trgMod
>device 108 (hash 7, line 0 srv dev 309): M16AdcPia.trgMod
>device 109 (hash 14, line 0 srv dev 310): M18AdcPia.trgMod
>device 110 (hash 15, line 0 srv dev 311): M21AdcPia.trgMod
>device 111 (hash 16, line 0 srv dev 312): M24AdcPia.trgMod
>device 112 (hash 17, line 0 srv dev 313): M26AdcPia.trgMod
>device 113 (hash 18, line 0 srv dev 314): M2AddPia.livSta
>device 114 (hash 19, line 0 srv dev 315): M4AddPia.livSta
>device 115 (hash 20, line 0 srv dev 316): M10AddPia.livSta
>device 116 (hash 21, line 0 srv dev 317): M12AddPia.livSta
>device 117 (hash 22, line 0 srv dev 318): M16AddPia.livSta
>device 71 (hash 23, line 0 srv dev 172): TstPlsPia.status
>device 118 (hash 23, line 0 srv dev 319): M18AddPia.livSta
>device 119 (hash 30, line 0 srv dev 320): M21AddPia.livSta
>device 120 (hash 31, line 0 srv dev 321): M24AddPia.livSta
>device 271 (hash 31, line 0 srv dev 772): M10AddPia.p13p14
>device 121 (hash 32, line 0 srv dev 322): M26AddPia.livSta
>device 122 (hash 33, line 0 srv dev 323): TrigPia.startRam
>device 117 (hash 35, line 0 srv dev 318): M16AddPia.livSta
>device 641 (hash 43, line 0 srv dev 1842): M26AddPia.sRamCt
>device 633 (hash 50, line 0 srv dev 1834): M2AddPia.sRamCt
>device 634 (hash 51, line 0 srv dev 1835): M4AddPia.sRamCt
>device 635 (hash 52, line 0 srv dev 1836): M10AddPia.sRamCt
>device 636 (hash 53, line 0 srv dev 1837): M12AddPia.sRamCt
>device 637 (hash 54, line 0 srv dev 1838): M16AddPia.sRamCt
>device 533 (hash 54, line 0 srv dev 1534): M16AddPia.staDat
>device 638 (hash 55, line 0 srv dev 1839): M18AddPia.sRamCt
>device 639 (hash 62, line 0 srv dev 1840): M21AddPia.sRamCt
>device 640 (hash 63, line 0 srv dev 1841): M24AddPia.sRamCt
>device 272 (hash 63, line 0 srv dev 773): M12AddPia.p13p14
```

Example 2: RegisterUserFunction()

- Less 'restrictive', but you will have to 'design more of it yourself'
- Prototype:
 - `int RegisterUserFunction(char *name, int (*fcn)(int argc, char *argv[]))`
- e.g.
 - `RegisterUserFunction("testfcn", testfcn);`

```
{
int testfcn(int argc, char *argv[])
{
    int i;
    printf("testfcn called with %d arguments\n", argc);
    for (i=0; i<argc; i++)
    {
        printf("argument %d : %s\n", i, argv[i]);
    }
    return 0;
}
```

Example 2: continued

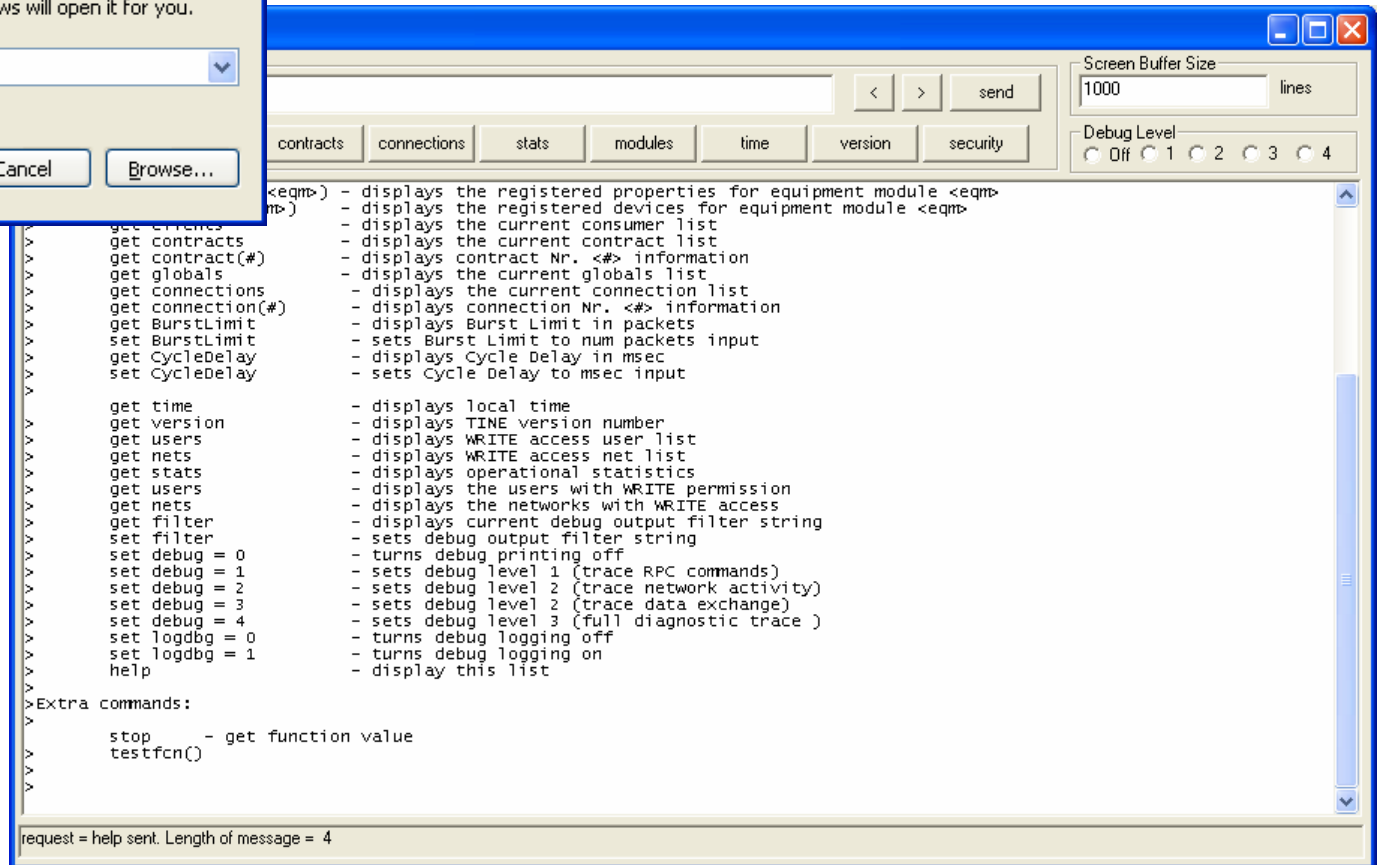
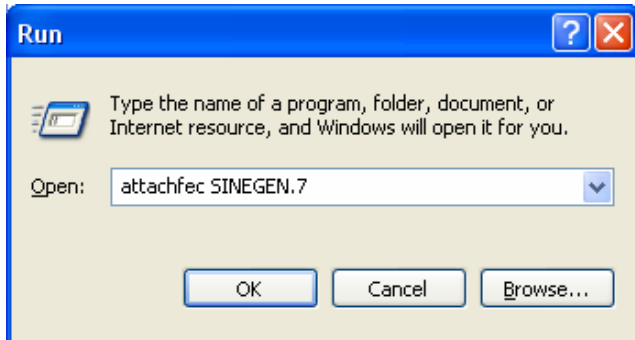


```
C:\ Z:\Projects\Service\vc++\tine32R4\SineGen\fec___Win32_Debug_MT\fec.exe
> set logdbg = 1          - turns debug logging on
> help                   - display this list
>
>Extra commands:
> stop                  - get function value
> testfcn()
>
>testfcn
>testfcn called with 0 arguments
result is 0
>
>testfcn(1,2,3)
>testfcn called with 3 arguments
argument 0 : 1
argument 1 : 2
argument 2 : 3
result is 0
>
>testfcn("hello world",2008)
>testfcn called with 2 arguments
argument 0 : "hello world"
argument 1 : 2008
result is 0
>
>
```


[Getting there with Attachfec ...]

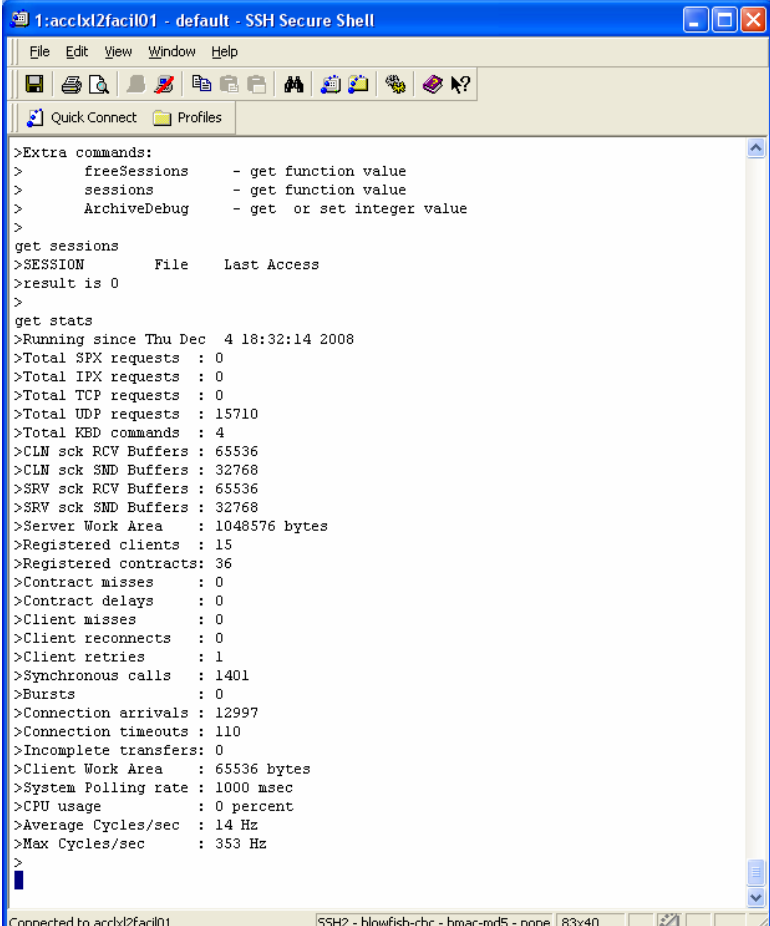
- If
 - server is a console program and is
 - running in the foreground and
 - you are sitting at the console
- Then you don't need attachfec!
- Otherwise,
 - login to the machine running the server and type
 - “attachfec <fecname>”

Windows Example:



UNIX Example:

- `ssh -l fecadmin acclxl2facil01`
- `attachfec L2CASFE`



```
1:acclxl2facil01 - default - SSH Secure Shell
File Edit View Window Help
Quick Connect Profiles
>Extra commands:
>   freeSessions - get function value
>   sessions     - get function value
>   ArchiveDebug - get or set integer value
>
get sessions
>SESSION      File      Last Access
>result is 0
>
get stats
>Running since Thu Dec  4 18:32:14 2008
>Total SPX requests : 0
>Total IPX requests : 0
>Total TCP requests : 0
>Total UDP requests : 15710
>Total KBD commands : 4
>CLN sck RCV Buffers : 65536
>CLN sck SMD Buffers : 32768
>SRV sck RCV Buffers : 65536
>SRV sck SMD Buffers : 32768
>Server Work Area   : 1048576 bytes
>Registered clients : 15
>Registered contracts: 36
>Contract misses    : 0
>Contract delays    : 0
>Client misses      : 0
>Client reconnects : 0
>Client retries     : 1
>Synchronous calls : 1401
>Bursts             : 0
>Connection arrivals : 12997
>Connection timeouts: 110
>Incomplete transfers: 0
>Client Work Area   : 65536 bytes
>System Polling rate : 1000 msec
>CPU usage           : 0 percent
>Average Cycles/sec : 14 Hz
>Max Cycles/sec     : 353 Hz
>
Connected to acclxl2facil01  SSH2 - blowfish-cbc - hmac-md5 - none 83x40
```