EInspect Quick Start

Contents

Overview

I have reviewed two Native Inspect manuals provided by Tandem. The full Native Inspect manual includes a quickstart section while the shorter manual, *Migrating From Inspect to Native Inspect,* is designed entirely as a quickstart for experienced Inspect users.

Both manuals exhibit the same basic two flaws. They do not adequately explain how to assign the source file to the object and they do not explain how to start Native Inspect for looking at saved files. As you can readily imagine, without understanding how to perform these two functions the rest of the manuals was completely useless. I think that this is the source of frustration that several have expressed about the product, eInspect. Note the strange capitalization in the spelling.

If you want to get started ASAP then skip to the next page titled *Quick Start*. Otherwise, follow along for some preliminaries.

GNU is a successor operting system to Unix. It is defined as "*GNU is not Unix"*. Notice that the definition requires its own definition recursively. This project was started about 1983 as a cross platform OS more powerful than Unix. See http://en.wikipedia.org/wiki/GNU.
A debugger was written as part of that ongoing project now called the Gnu DeBugger or GBD. It has seen wide acceptance.

Native Inspect was adapted from this existing open source debugger for the Itanium processor. It was ported to the Tandem not written by HP. This was done, as admitted by one of their reps, to save the cost of development. But, it is of value for us to know the product because that skill can be used on other systems. Native Inspecet is a GBD.

Native Inspect is a command line debugger that functions very much like Inspect. Visual Inspect is another tool that allows using the mouse to set breakpoints or to display variables. Software must be installed on both the Tandem and the PC to run Visual Inspect. Native Inspect allows multiple commands to be executed from a stored edit file, that is, a script, but does require the tedium of typing while Visual Inspect provides point and shoot efficiency without the ability to script repeated commands.

An additional tool, TCL, provides additional scripting and conditional execution that envelops eInspect. At first, according to my unnamed rep, Tandem was not going to publish a Native Inspect manual thus forcing users to use Visual Inspect. As of this date, June 10, 2011, this paper does not describe Visual Inspect but does provide some information about TCL.

Quick Start

Starting

Start a program for debugging in the usual way

- *Set Debug On* in the Pathway serverclass
- *RunD object* from Tacl
- *Debug* <*cpu,pin*> from Tacl
- Scobol is debugged by Inspect just as before

The operating system will choose the debugger that is appropriate for the object, Inspect, eInspect or Debug.

Debugging a save file

- Type *eInspect* from Tacl then issue the new command *Snapshot ZZsannnn*. This is what you would use if the save file is from an code-800 object.
- Type *Inspect* from Tacl then issue the former command *Add Program ZZsannnn*. This is what you would use for code-700 or code-100 objects. That is, there is no change to debugging save files from these kinds of objects.

Assigning the Source File

When doing an assignment of new source you must include the **Node Name** of the original source and that name is **Case Sensitive.**

Commands are not case sensitive but their arguments may be. The source assignment command is

Map source-name = *new-source-name* !!! *source-name* is case sensitive

```
Example:   MAP \KFSZ.$RMSB.RMSTEMP.A0095302 = $sourc1.dbgwhsrc.swhovu02
```

You can use a trick to aid this assignment. Do a LIST command. This will cause a display of the expected file name and, probably, a warning that it can't be found because RMS uses temporary files for compilation. Using Windows, copy the filename into the scratchpad then paste it into the MAP command. This avoids missing the node name or altering the case.

Sample eInspect session with highlights in blue:

Server started with Debug ON

```
TNS/E Native Inspect gdb Debugger [T1237 - 23-Jan-2008 18:36]
Copyright 1998 Free Software Foundation, Inc.
Copyright 2003-2008 Hewlett-Packard Development Company, L.P.

Native Inspect (based on GDB) is covered by the GNU General Public License.
Type "show copying" for conditions for changing and/or distributing copies.
Type "show warranty" for warranty/support information.

Working directory \KFSZ.$TEST04.TE4.
Symbols read in for program loadfile \KFSZ.$TEST08.TE4OBJ.SWHOVU02.
Added process (1,1184).
Switching process (1,1184) to eInspect from DMON
Breakpoint 1 at 0x70844840:0: file \KFSZ.$RMSB.RMSTEMP.A0095689, line 19.
 (eInspect 1,1184):list
\KFSZ.$RMSB.RMSTEMP.A0095302: No such file or directory.
(eInspect 1,1184):map \KFSZ.$RMSB.RMSTEMP.A0095302 = $sourc1.dbgwhsrc.swhovu02
(eInspect 1,1184):list
warning: Timestamp mismatch for $sourc1.dbgwhsrc.swhovu02
Source modification time at present: 2005-03-09 14:34:48
Source modification time at compilation: 2008-04-08 13:58:46
     1               ?SAVE ALL
     2               ?HEADING "SERVER SWHOVU02"
     3               ?Symbols; Inspect
     4               ?Diagnose-85; DiagnoseAll; Errors 99
     5               ?Subset EXTENDED, DEB1, SEG2
     6               ?FMap
     7               ?CONSULT =D45_NMCSALNK_SSA50301
     8               ?CONSULT =D45_NMCSALNK_SSA60001
     9               ?CONSULT =D45_NMCSALNK_SSA61001
    10               ?CONSULT =D45_NMCWHLNK_SWHGL801
```

You will find that the expected source file is an RMS temporary file and that a single program may require the mapping of more than one temporary file name to more than one new source names or to the same source name twice. This is not new to eInspect.

Server enters debug after process has started

This presents a subtle problem because you will enter debug somewhere inside of a system call, most likely READ. That means that the preceeding tips about LIST and MAP won't display as advertised. You have to set your scope first using the BT and FRAME commands. Process enters eInspect which displays:

Working directory \KFS3.$SYSTEM.DBA.
Symbols read in for program loadfile \KFS3.$PROD01.POBJ.SAR07101.
Added process (3,1114).
Switching process (3,1114) to eInspect from DMON
Process (3,1114) forced into DEBUG.
(eInspect 3,1114):**list**
Can't find a default source file ← happens because we're inside of the READ routine
(eInspect 3,1114):BT ← This is the old TRACE
#0 0xfffffffff905d270 in $n_EnterPriv ()
#1 0xfffffffff238b7c0 in CRE_RECEIVE_READ_ ()

#2 0xfffffffff2ee5d00 in COBLIB_READSEQ_ ()
#3 0x7001ed20:0 in SAR07101 () at **\KFSZ.$RMSB.RMSTEMP.SAR07101:638 ← so we are here**
(eInspect 3,1114):**FRAME 3 ← establishes scope to the application level**
#3 0x7001ed20:0 in SAR07101 () at \KFSZ.$RMSB.RMSTEMP.SAR07101:638
\KFSZ.$RMSB.RMSTEMP.SAR07101:638: No such file or directory
Current language: auto; currently COBOL
 (eInspect 3,1114):LIST
\KFSZ.$RMSB.RMSTEMP.SAR07101: No such file or directory. ← copy to clipboard
  15> MAP \KFSZ.$RMSB.RMSTEMP.SAR07101=\KFSZ.$SOURC1.ECBARSRC.SAR07101
(eInspect 3,1114):LIST
warning: Timestamp mismatch for \KFSZ.$SOURC1.ECBARSRC.SAR07101
Source modification time at present: 2008-06-10 11:34:37
Source modification time at compilation: 2008-06-11 07:15:43
  635   C000-MAIN-BODY.
  636   **
****
  637
* 638   READ MSG-IN-FD
  639   AT END
  640   MOVE CON-TRUE TO EXIT-PROGRAM
  641   EXIT PARAGRAPH.
  642
(eInspect 3,1114):

Basic Commands

Both the quickstart manual and the full Native Inspect manual provide a conversion list of Inspect to Native Inspect commands. The following list is not as extensive but should be enough to provide basic debugging.

Both FC and fc are valid but the argument is **CASE SENSITIVE**.
FC map will not find the same command as *FC MAP*.

Inspect Command	Native Inspect Command	Description
Add Program – for zzsa file	Snapshot	Debug a save file
Alias xx = "D hv-district"	Define xx P hv-district end	xx will perform P hv-district Must be end not End
Assign old-source,new-source	Map old-source = new-source Map \kfsz.$rmsb.rmstemp.Ann = $sourc1.dbgwhsrc.swhxxxxx	Include the node name \kfsz Old source is case sensitive
B paragraph or line number	B paragraph or line number not #line number	B C000-Main-Body B 1825 – don't do B #1825
B #subprogram or Tal #proc	B subprogram-name without the #	Wrong: B #Cwhg4401 – don't use the #
B paragraph IF condition	B paragraph → yields # Condition # xxxx EQUAL yy	This can be tricky – see notes below. Don't use "="
D variable-name	Print variable-name Print /x variable-name For Hex output P Length variable	Print A or P A Print /x A for hex (eInspect 3,1009):set A = "1234" (eInspect 3,1009):P /x A $8 = {0x31, 0x32, 0x33, 0x34} Gives length of variable
Exit	Exit	
FC or fc	FC or fc	But argument is case sensitive
Files	Fopen	List files opened by program
Inspect- Add Program Zzsa	eInspect: Snapshot Zzsa	
Modify variable = new value	P variable = new value OR SET variable = new value	
OBEY commands in a file	Source file name	Execute commands from a file
R	C, Continue	
R at line number or pgraph	Jump line-number	
Scope	Frame	Selects program from the call stack – see bt
Source	List	List source
Source Search	Not supported	
Step	N, Next	
Step IN	Step	
Stop	Kill	Stops eInspect and should

		stop the program. Now works even when not at a breakpoint
Trace	BT	Displays the call stack

Notes On The Commands and More Advanced Uses

Conditional Breakpoints *can be set – commands shown in red*

(eInspect 0,716):B J600-POST ← Break command
"7" becomes the breakpoint number below
Breakpoint 7 at 0x70889190:0: file \KFSZ.$SOURC1.ECBWHSRC.SWHOVU02, line 2376.
(eInspect 0,716):CONDITION 7 REPLY-CODE EQUAL 99 – puts a condition on breakpoint 7

GOTCHA

The symbol "=" cannot be used instead of EQUAL. What is worse is that using "=" is accepted as a valid command which serves to assign the condition not to test it. JD discovered this.
The manual lists EQUAL TO but that does not work.
There is no "IF" in the setting of the condition.

Define replaces Alias

```
 (eInspect 2,534):Define XX          ← You type this
Type commands for definition of "xx". ← eInspect responds
End with a line saying just "end".    ← eInspect responds
>p hv-district                       ← What you want XX to do.  You type this.
>end                                 ← Stop the list of commands -  lower case!
(eInspect 2,534):XX                  ← Now XX does "print hv-district"
$3 = 1933
```

Fopen replaces Files
Shows files open at a breakepoint. This command became available about release 13+ and notice of that is courtesy T. Begonja.

```
(eInspect 2,534):Fopen
FileNum  LastErr  Name
1        0        \KFSZ.$TEST09.TE4DBS.MFLOCNBT
2        0        \KFSZ.$TEST09.TE4DBS.WHSBPHBT
 (eInspect 2,534):
```

Jump line number replaces **Restart At** line number

Pslice macro to display a range of elements of an array

Pslice is a TCL macro which you have to load – see below for further description of TCL

Load the macro called co_pslice:

(eInspect 1,888):tcl source \$test01.jparchiv.pslice ← don't skip the extra "\"

Example

```
05 ROUTE-OV-REC  OCCURS 20000 TIMES.
      10 OV-SEQUENCE-NBR            PIC  9(6).
```

After loading the macro you can now type:

co_pslice route-ov-rec.ov-sequence-nbr 10 15 ← subscript is From To
Old Inspect would be: Display ov-sequence-nbr of route-ov-rec (10:5)

```
route-ov-rec.ov-sequence-nbr(10) = 225026
route-ov-rec.ov-sequence-nbr(11) = 225040
route-ov-rec.ov-sequence-nbr(12) = 225030
route-ov-rec.ov-sequence-nbr(13) = 225054
route-ov-rec.ov-sequence-nbr(14) = 225055
route-ov-rec.ov-sequence-nbr(15) = 225033
```

P Length gives the length of a variable which used to be shown as part of the Display command. This is very useful for determining the length of a complicated structure that will be redefined or of RPL-OUT to resolve transaction reply size discrepancies.

P Length variable ← Note the space between the P and the L
(eInspect 1,1263):p length msg-in
$3 = 2000

Trace – provided at the request of P. Langley
Set a breakpoint in the usual way giving a breakpoint number N
Then enter:
Commands N
Enter what you would like eInspect to do when it stops at breakpoint N
Enter the next command
Enter the next command
end ← to stop the command list

Example program

```
100-MAINLINE.
   if i > 3
```

```
        stop run.
    add 1 to i
    go to 200-continue.
200-continue.
    go to 100-mainline.
```

```
(eInspect 2,780):b 100-mainline
Breakpoint 2 at 0x700014d0:0: file \KFSZ.$SOURC1.ECBWHSRC.SWHCSK01, line 48.
Current language:  auto; currently COBOL
(eInspect 2,780):commands 2
Type commands for when breakpoint 2 is hit, one per line.
End with a line saying just "end".
>print i
>continue
>end
(eInspect 2,780):c
Continuing.
```

You now have a trace:

```
Breakpoint 2, COBSKEL () at \KFSZ.$SOURC1.ECBWHSRC.SWHCSK01:48.010
*   48.010       if i > 3
$1 = 0000000

Breakpoint 2, COBSKEL () at \KFSZ.$SOURC1.ECBWHSRC.SWHCSK01:48.010
*   48.010       if i > 3
$2 = 0000001

Breakpoint 2, COBSKEL () at \KFSZ.$SOURC1.ECBWHSRC.SWHCSK01:48.010
*   48.010       if i > 3
$3 = 0000002

Breakpoint 2, COBSKEL () at \KFSZ.$SOURC1.ECBWHSRC.SWHCSK01:48.010
*   48.010       if i > 3
$4 = 0000003

Breakpoint 2, COBSKEL () at \KFSZ.$SOURC1.ECBWHSRC.SWHCSK01:48.010
*   48.010       if i > 3
$5 = 0000004
Process (2,780) exited normally.
Removed process (2,780).
```

TCL Intro

The **GNU** is an open source operating system designed to extend the power of UNIX. It stands for "GNU is Not Unix". It is a project still in development.

GBD stands for the GNU DeBugger.
While the GNU is not in production, its debugger has become a standard interface to many platforms. Thus, although unnecessarily frustrating to us as trained Inspect users, this new debugger is useful to know for other platforms.

eInspect is a GBD.

TCL is an open-source scripting language that is implemented in eInspect. TCL stands for Tool Command Language. The native Inspect Manual contains a very brief intro to this language together with links to self-instruction exercises. TCL runs in the background of our GBD, eInspect.

When giving a command to eInspect the GBD receives it and executes it.
If you want TCL to execute the command you must preface your command with TCL.
Some eInspect commands are already written to pass through to TCL such as JUMP.

Thus

:source filename – loads eInspect commands from a file (like Obey)
:TCL source filename – loads TCL commands from the file

The reason that I'm introducing TCL, although superficially, is that I am providing one macro that is very useful. It displays a range of data in a Cobol array, a function that was part of the old Inspect.

To look at a range of an array in Cobol load this macro from within eInspect: If your server is started in eInspect for $Te04 you will be pointed to $Test04.Te4 where the macro has been placed.

:TCL source pslice ← lower case with "source"! Upper or lower case with TCL

```
(eInspect 2,1081):fc
   3> TCL source \$test04.te4.pslice  ← the extra "\" is required
   3..tcl
   3> tcl source \$test04.te4.pslice
   3..
(eInspect 2,1081):fc
   4> tcl source \$test04.te4.pslice
   4..   SOURCE
   4> tcl SOURCE \$test04.te4.pslice
```

4..
invalid command name "SOURCE"

Example:

```
        01  WS-DISTRICT-ABBREV.
              05 WS-DA-ENTRY OCCURS 50 TIMES.
                 10 WS-DA-DESCRIPTION     PIC X(15).
                 10 WS-DA-DISTRICT        PIC 9(04).
                 10 WS-DA-ABBREV          PIC X(03).
```

In old Inspect you could type
Display WS-DA-ENTRY (3:5) to get 5 occurrences starting with the 3rd..

In eInspect, with the TCL macro loaded. It has been loaded to the GBD. You can now type

:Co_Pslice WS-DA-ENTRY 3 5

ws-da-entry(3) =
WS-DA-DESCRIPTION = "MILWAUKEE "
WS-DA-DISTRICT = 0248
WS-DA-ABBREV = "MIL"
ws-da-entry(4) =
WS-DA-DESCRIPTION = "INDIANAPOLIS "
WS-DA-DISTRICT = 1054
WS-DA-ABBREV = "IND"
ws-da-entry(5) =
WS-DA-DESCRIPTION = "CINCINNATTI "
WS-DA-DISTRICT = 1103
WS-DA-ABBREV = "CIN"

Gotcha
In TCL the "$" sign is a control character which refers to a display occurrence of a variable. The symbol "\" is a control character that causes the special nature of the next character to be escaped.
This means that trying to source in a file like \Kfsz.$Test04.Te4.Pslice becomes problematic. If you're already in the corrrect vol/subvol as mentioned above then just load the macro as

:TCL source pslice

But if you're not in that subvol you'll have to do

:TCL source \\Kfsz.\$Test04.Te4.Pslice ← extra "\" to escape the special function that follows