Native Inspect
Quick Reference Card

Part #: **544998–004** Feb 2012 - (C)2012 Hewlett-Packard L.P

Starting Debugging Sessions

- Compile the program files with symbols and at optimization level 1 (or 0).
- If in a PC-based development environment, transfer the files to a TNS/E host.
- Launch Native Inspect and gain debugging control.
- To customize your sessions, modify the **EINSCSTM** custom file in your logon default `volume.subvolume.`

Manual Startup Options

If you start Native Inspect manually, you can enter the following command options:

- **nocstm** - Ignore the custom file.
- **version** - Display the debugger version.
- **help** - Display the online help.

Starting processes under debugger control in TACL or OSS:

- **TACL> rund *filename***
- **OSS> run -debug *filename***

Starting the debugger on a running process:

- **TACL> debug [processname], term [*term_name*]**
- **OSS> debug [*cpu_id*] [*process_id*]**

Invoking the debugger from a running TNS/E native process:

- A call to `PROCESS_DEBUG` or `DEBUG.`
- Encountering a previously-set breakpoint.
- User switches debuggers to Native Inspect.

Use the **attach [*pin*] | [$*process-name*]** command from within Native Inspect. (The process must run in the same CPU.)

Traps and signals can cause the debugger to start automatically.

Stopping Native Inspect

- To manually stop Native Inspect, enter: **exit** or **quit**.
- Native Inspect automatically stops when its target process stops.

Snapshot Files

Snapshot files save a process state for later analysis

- **save filename [compression] [!]** – Provides control over the compression that is applied to the file.

  compression can be one of *bzip2, gzip, or none*, and this applies when debugging a 64-bit process.
- Set the compiler option **SAVEABEND** to **ON** - Creates snapshot files automatically when a process abends.
- **snapshot [pathname]** - Examine a snapshot

Switching Debuggers

- **switch** - Switch to Inspect or Visual Inspect.
- Switch to System Debugger - Switch to Native Inspect from Visual Inspect.
- **SELECT DEBUGGER DEBUG** - Switch to Native Inspect from Inspect.

Utility Commands

- **amap [*address*]** - Displays attributes associated with an address.
- **base {input | output} {HEX | OCT | DEC}** - Sets the base for numeric input and output.
- **cd [*pathname*]** - Changes the current working directory.
- **{comment | #} [*text*]** - Precedes a comment line.
- **eq [*expression*]** - Evaluates an expression and displays the result in several bases.
- **fc [*command-number* | *command-string*]** - Redisplays a command for editing and execution.
- **{files | ls} [*pattern*]** - Displays the files in the current working directory.
- **fopen [file-num] [-d]** - Displays information about the files opened by the current program.
- **info *attribute*** - Displays information about the debugging target.
- **log [*pathname* | -d]** - Turns session logging on or off.
- **pwd** - Prints the current working directory.
- **{quit | exit}** - Ends a session.
- **set *attribute value*** (Environment) - Establishes environment settings.
- **show [*attribute* | {history | print | check} [*sub-attr*]]** - Displays environment settings.
- **source [*pathname*]** - Reads debugger commands from a file.
- **which *symbol*** - Prints file, function, and line information of the specified (text or data) symbol.

Source File Commands

- **dir [*directory*]** - Specifies the search path used to locate source files when they can not be found at their compile-time location.
- **list [*start-linespec*][,*end-linespec*] [+|-]** - Lists source code.
- **{map-source-name | map} [[*source-name*]=*alias-name*] | [*source-prefix*=*alias-prefix*]** - Defines mapping rules between the compile-time name and location of a file and its current name and location. (Useful when a file is renamed).

Controlling Sessions

- **attach [*pin*] | [$*process-name*]** - Associates Native Inspect with a process that is executing in the same CPU.
- **detach [*pin*] | [$*process-name*]** - Disassociates Native Inspect from the current process.
- **{next | nexti} [*count*]** - Advances program execution by one or more statements, stepping over function calls.
- **priv {on | off}** - Controls the privilege level of the working session (super user only).
- **switch** - Transfers the current process to another debugger.
- **vector [*pin*] | [$*process-name*]** - Changes the current process.
- **wait** - Suspends prompting pending the next debug or Break key event.

Controlling Processes

- **continue [*ignore-count*]** - Resumes execution of the current process.
- **finish** - Executes the current process until execution returns from the currently selected frame.
- **hold** - Suspends the current process so that you can perform debugging operations.
- **jump *linespec*** - Resumes execution at the specified location.
- **kill** - Terminates the current process.
- **{step | stepi} [*count*]** - Advances program execution by one source statement or a specified number of statements, stepping into any called functions.
- **until [*linespec*]** - Continues execution of the current process until a specified location is reached or until the current stack frame returns.

Controlling Execution

- **mh *signal-name* {SIG_IGN | SIG_ABORT | SIG_DFL | SIG_DBG | *native-address*}** - Modify signal handlers for a specified signal (modify handler).
- **ih [*signal-name*]** - Display information about signal handlers (info handler).

Controlling Breakpoints

- **{break | tbreak} [*linespec*] [*flags*] [-e *llce* | if *cond-exp*]** - Sets an instruction breakpoint (temporary, for **tbreak**).
- **catch *event*** - Sets a logical breakpoint on a specified event.
- **commands *breakpoint-number*** - Specifies commands that execute at a specified breakpoint.

Controlling Breakpoints (Cont.)

- `condition breakpoint-number [conditional-expression]` - Specifies a conditional expression for evaluation at a specific breakpoint.
- `delete [breakpoints] [breakpoint-number ...]` - Deletes instruction breakpoints
- `disable [breakpoints] [breakpoint-number ... ]` - Disables specified breakpoints
- `dmab [-g]` - Deletes a memory access breakpoint (MAB).
- `enable [once | delete] [breakpoint-number ...]` - Enables breakpoints that have been disabled.
- `ignore breakpoint-number ignore-count` - Sets the number of breakpoint hits to ignore.
- `info breakpoints` - Lists information about all user-defined breakpoints (including per-process breakpoints, global breakpoints, and catch events.).
- `mab [{*native address | variable} [size][flags] [-e linespec]]` - Sets a memory access breakpoint (MAB).

Displaying and Modifying Process Information

- `a native-address [count]` - Displays memory in ASCII format.
- `d native-address [count] [:format]` - Displays memory in a specified format.
- `delete display [num]` - Removes an expression from the automatic display list.
- `disable display [num]` - Disables automatic display items.
- `{disassemble | da} [[start-address][end-address] | function-name]` - Displays a range of memory as instructions.
- `display [[/format] expression]` - Adds an expression to the list that is automatically displayed when the process is suspended.
- `enable display [num]` - Enables automatic display items.
- `env` - Displays process environment information.
- `fn value [start-addr [end-addr]] [type]` - Searches for a value (finds a number) in the virtual address space of the current process.
- `i {{native-address [count]} | function-name}` - Displays memory as instructions.
- `output [/format] expressions` - Displays the value of a specified expression (does not save the result in *value history*).

Displaying and Modifying Process Information (Cont.)

- `modify native-address value {8 | 16 | 32 | 64}` (or `info` with `registers`) - Changes the content of memory.
- `print` - `[/format] expressions` Evaluates and display the value of a specified expression (saves the result in *value history*).
- `reg` - Displays registers.
- `set [variable] var-name {expression | value}` (Variable) - Evaluates an expression and assigns its value to a variable.
- `x [/format] address` - Examines memory at a specified address.

Displaying and Selecting Stack Information

- `{bt | tn} [count]` - Prints a backtrace of all the stack frames.
- `{down | down-silently} count` - Selects the stack frame that is called by the selected stack frame.
- `{frame | select-frame} [number]` - Selects a specified stack frame.
- `info attribute` (with the `frame` option) - Displays information about frames and registers.
- `tj native-address` - Traces the stack from a TNS/E native jump buffer contained at the specified address.
- `tu native-address` - Traces the stack from a `ucontext` buffer contained at the specified address.
- `{up | up-silently} count` - Selects the stack frame that called the currently selected stack frame.

Using Object and Symbol Files

- `{add-symbol-file | symbol-file} filename` - Adds symbol file information.
- `ptype [data-type | variable-name]` - Prints information about a specified data type.
- `{symbol | symbol-file} [-g] [-readnow] pathname` - Opens a native code file and builds internal symbol tables.
- `unload-symbol-file [-g] symbol-file-name` - Discards symbol data associated with a specified filename.
- `whatis expression` - Displays the data type of a specified expression.

Managing Memory

- `set heap-check attribute value` - Displays settings for commands that debug memory problems.
- `vq [segid]` - Displays information about the extended segments allocated by the current process. Specify `segid` to change the current selectable segment.

Command Syntax Elements

- `Linespec` - Specifies a single source line with the `list` or `breakpoint` commands, using:
  - `number` - A line number in the current file.
  - `filename:number` - A line number in the source.
  - `function` - A line at which the body of the function begins.
  - `filename:function` - The line at which the body of the specified function begins.
  - `*address` - The line containing the program address.
- `native-address` - Specifies a 32-bit or 64-bit address, using the following:
  - Hexadecimal (for example, 0x120001DC0).
  - Decimal (for example, 48331845824).
  - Octal (for example, 044000016700).
- `llce` - Specifies a low-level conditional expression to set conditional breakpoints with the `break`, `tbreak`, or `mab` commands as follows:
  `-e native-address [& mask] operator value`
  - `mask` is a 64-bit mask.
  - `operator` is one of `!=`, `==`, `<` or `>.`
  - `value` is an integer.
- `expression` - Specifies a list of operands and operators for evaluation. Other valid expressions are:
  - `$, $$` - Refers to the last two values printed.
  - `$number` - References previous print commands.
  - `$register-name` - Displays register contents.
- `/format` - Specifies a repeat count with the `x`, `print`, and `output` commands as follows: `/[count][format][size]`
  - `count` - An integer specifying the number of units of size to display or print.
  - `format` - Specifies the display format as follows:

a address	**i** instruction (ICODE)	**u** unsigned decimal
c char	**o** octal	**x** hexadecimal
d decimal	**s** null terminated string	
f float	**t** binary	

  - `size` - Specifies the unit size as follows:

b byte	**h** half word (16 bits)
w word (32 bits)	**g** giant (64 bits)

pTAL and COBOL Considerations:

- For pTAL programs use: `procedure[subprocedure]`
- For COBOL programs use one of the following:

```
Program-unit [program-unit]
[section.]paragraph
paragraph [of section]
```