

# Principals of Debugging

PxPlus 2017 (v14) & PxPlus 2018 (v15)

# Goals for 'Principals of Debugging'

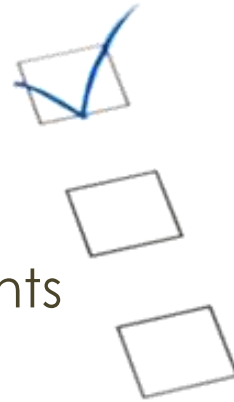
- Adapt debugging procedures to newer environments
  - Application environments are changing
    - More focus of background services
    - More “truly remote” access to application
    - Less dependence on direct application desktop
- Time to change how we debug
  - Adding ESCAPE and stepping less practical
  - Need to debug remotely



# Agenda

## Principals of Debugging

- Old School
  - Escape, Trace, and Breakpoints
- Modern Approaches
  - PxPlus debug windows
    - WindX & remote debugging
  - Tracing IF and Jumps
- Real-time debugging
  - The \*IT, DBG built in debugger
  - Debugging in *iNomads*
- Post mortem debugging
  - The Jump back-trace
  - Error history
  - Software failures



# Old School Debugging

- Real time interactive debugging
  - Put an ESCAPE in the program
  - Step through the logic
  - Use > to set breakpoints

All of the above required direct workstation/terminal access

- Background task debugging
  - SETTRACE to a file
  - Insert DUMP directives at key points
  - Post Mortem analysis of the problems



# Enhanced Old School Debugging

- ProvideX/PxPlus introduced debug windows

- **Trace**

- Shows program execution

- **Watch**

- Shows values of expressions/variables

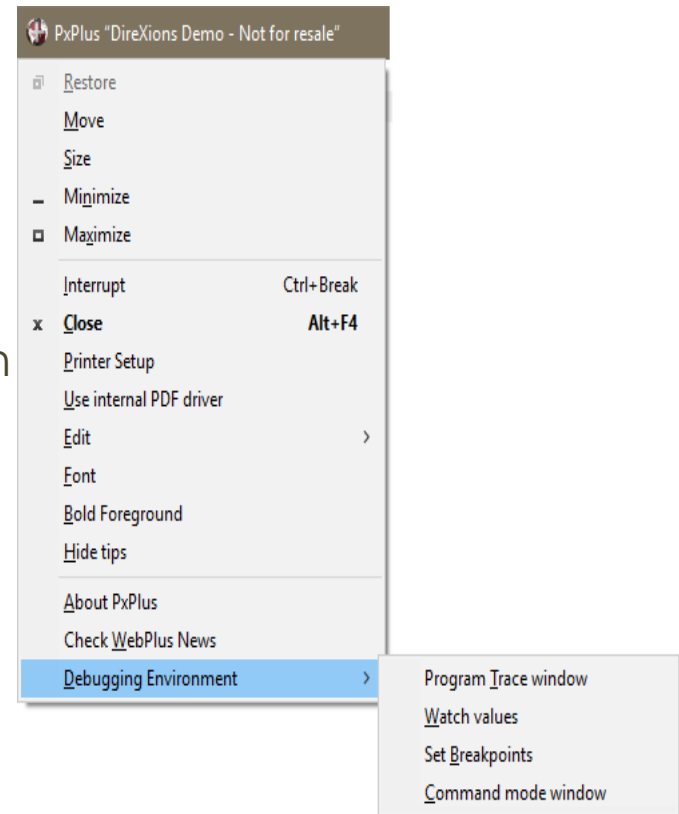
- **Breakpoint**

- Dynamically set point to **halt** execution

- **Command**

- Allows entry of commands without harming display

- Disabled by default if lead program
  - Re-enable by adding DEBUG=1 to INI

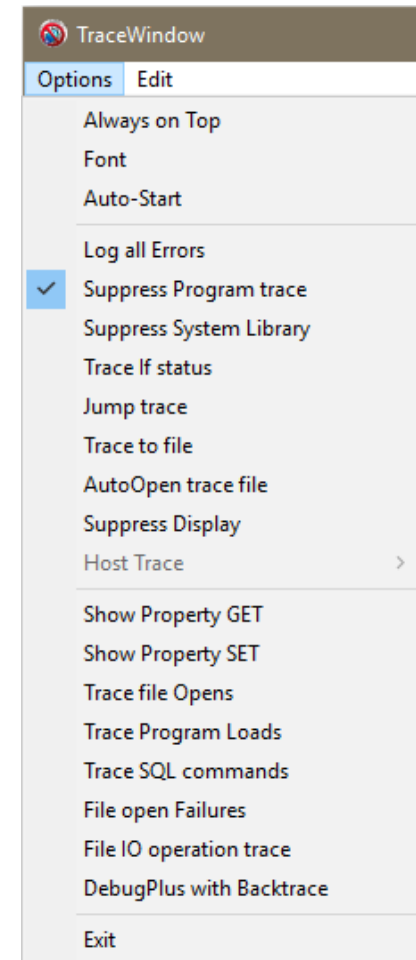


# PxPlus Debugging

## Trace Window

- Provides display of the following
  - Program statements as executed
    - Can be suppressed (also suppress system library)
  - Can output to a file
  - Auto-Start capability
  - Optional
    - Error log
    - If and Jump traces
    - Property Get/Set operations
    - File opens and program loads
    - Failed file access requests
    - Generated SQL commands
    - File IO commands

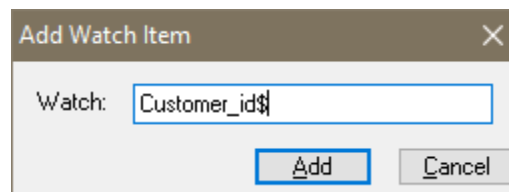
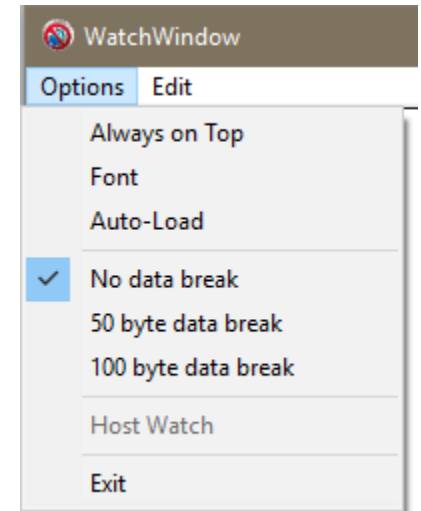
Can include back trace (stack)



# PxPlus Debugging

## Watch Window

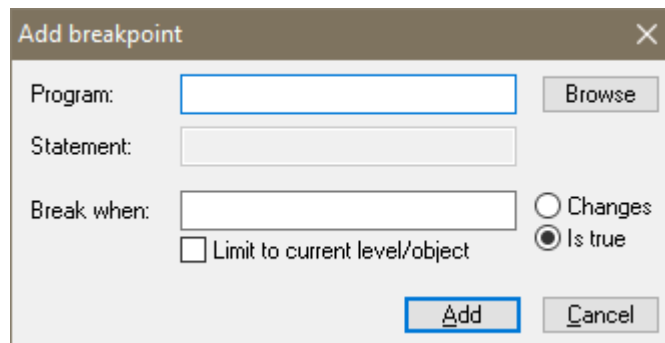
- Provides display of the variables and expressions
  - Contents change dynamically
  - To set contents select Window and
    - Press “Insert” key to add to bottom of list
    - Press “Delete” key to delete current selection
  - Ability to save what to watch
  - Optional Auto-start



# PxPlus Debugging

## Breakpoint Window

- Allows dynamic execution of ESCAPE
  - Based on expression / program
  - Requires no code changes
  - Breakpoints can be saved and auto-loaded
  - PxPlus allows breakpoint by level / Object
    - Avoids problem with conditional tests based on local variables



The screenshot shows a dialog box titled "Add breakpoint" with a close button (X) in the top right corner. The dialog contains the following fields and controls:

- Program:** A text input field followed by a "Browse" button.
- Statement:** A text input field.
- Break when:** A text input field followed by two radio buttons: "Changes" (unselected) and "Is true" (selected).
- Limit to current level/object:** A checkbox that is currently unchecked.
- Buttons:** "Add" and "Cancel" buttons at the bottom.



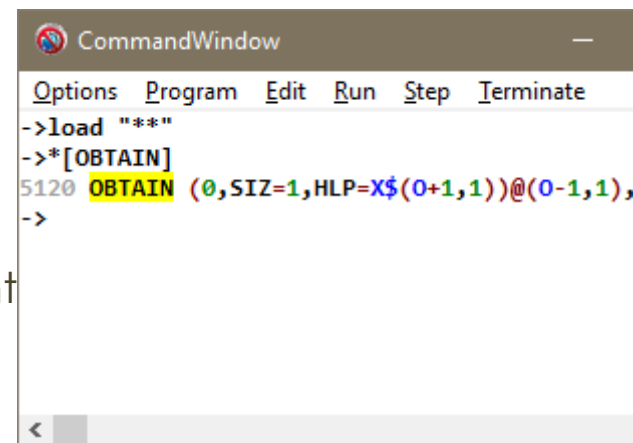
# PxPlus Debugging

## Command Window

- Allows commands to run without impacting windows display
  - Command input/output directed to separate window
  - Accessing controls performs swap behind the scenes
    - Swap back to command window occurs on next input
  - Causes issue with command like

### **PRINT ctlNo'Property**

- Output occurs on 'Live' window
- Save value to dummy variable then print



```
CommandWindow
Options Program Edit Run Step Terminate
->load "***"
->*[OBTAIN]
5120 OBTAIN (0,SIZ=1,HLP=X$(0+1,1))@(0-1,1),
->
```

# Limitations of Debug Windows

- Must be Graphical 'Windows' environment
  - No support on Text mode devices
- Still based on connected workstation
  - Windows shown and controlled by the workstation running application

# WindX and Remote Debugging

- Remote access to debug windows
  - **Trace Window**
    - Can enable host side tracing
    - Trace output will be sent to workstation
      - Host trace prefixed with <h>
    - Will significantly slow process due to transmissions
  - **Command Window**
    - Provides access to host command line interface
    - Some issues with screen redrawing
      - Vastly improved in PxPlus 2017
      - Primarily with initial cursor position during editing
  - **Breakpoint** and **Watch** also available

# Tracing IF conditions

- Outputs True/False state for 'IF' statements
  - Output breaks down each IF encountered
    - Shows True/False for condition
    - Includes colour indication of state in Trace Window
    - Requires Tracing be active to show actual statements
  - Can be enabled by
    - Setting '**IT**' (If Trace) system parameter
    - Trace Window option

# Tracing transfer of control

- How can it help debug programs?
  - Shows where program has been
  - Allows developer to follow execution flow
  - Can reduce trace output to only transfers
  - Provides ERR feedback when error transfer taken
    - Answers question:  
*Was function/method called?*
  - Enabled through Trace Window option

# Getting Additional Information

- The About PxPlus => **Info** display

The screenshot shows the PxPlus Development Suite interface. The 'About PxPlus' dialog box is open, displaying system information and a table of statistics. The 'Info' button in the dialog is highlighted with a blue arrow. To the right, the 'PxPlus Current State Information' window is open, showing detailed system information and a program backtrace. The 'Done' button in this window is also highlighted with a blue arrow.

**PxPlus Development Suite**  
by PVX Plus Technologies

Serial Number: 1060295 E-Comm	Memory in use: 1044733
Cur/Max Users: 1/5	Files Open: 8
Date Installed: May 30 2016	File Reads: 1900
Date Expires: <none>	File Writes: 33
Userid: Mike king	Program loads: 445
Workstation: Mike-acer	Cached loads: 506

© 2005-2017 PVX Plus Technologies Ltd. (Ontario, Canada)  
All rights reserved Worldwide

Task ID: 3000

**PxPlus Current State Information**

PxPlus Version 14.00.0002 -- Build date Jul 25 2017@11:13:04  
-----  
Visual Studio build  
Process dump as of: Fri Jul 28 15:15:09 2017  
-----  
Program Backtrace:  
Lvl 004: C:\pxplus\lib\\_winold\define (line:420)  
Lvl 003: C:\pxplus\lib\\_win\define (line:63560)  
Lvl 002: C:\pxplus\lib\\_nomads (line:2410)  
Lvl 001: <No program name> (line:0)  
-----  
Open Files:  
File 00000: CON  
File 00061: C:\pxplus\lib\\_dict\scrnlib.en  
Keyed file. Current record: '.DELEMENT'  
(File has EXTRACT pending)  
File 00062: C:\pxplus\lib\\_dict\scrnlib.en  
Keyed file. Current record: '.DELEMENT 0000'  
File 00063: C:\pxsrc\providex.clr  
Keyed file. Current record: <Start of file>  
File 00124: \*MEMORY\*  
(File is locked for exclusive use)  
File 00125: C:\pxplus\lib\\_win\panel.inf  
Keyed file. Current record: 'Mike King' OBJSELVT scrnlib.en'  
File 00126: \*MEMORY\*  
(File is locked for exclusive use)

Background processes

# REAL TIME DEBUGGING

# PxPlus Built-in Debugger

- Language extensions build into PxPlus
  - Enhanced TSK(\* xxxx) functions provide
    - List of processes
    - Ability to connect/debug remote process
      - Must be on same server / service
    - Access to variables and program code
    - Can control execution
      - Halt, Run, Step and Terminate functions
- Three utilities provide access
  - \*IT graphical utility
  - DBG text mode command
  - iNomads DEBUG transaction

Functions are NOT  
included in standard  
documentation

Review  
\***cmd/system/dbg**  
for usage



# PxPlus Built-in Debugger - \*IT

## Graphical Interface

- Can connect to RUNNING process

- Shows current statement

- Displays

- Call Stack

- Status of open files

Provides access to

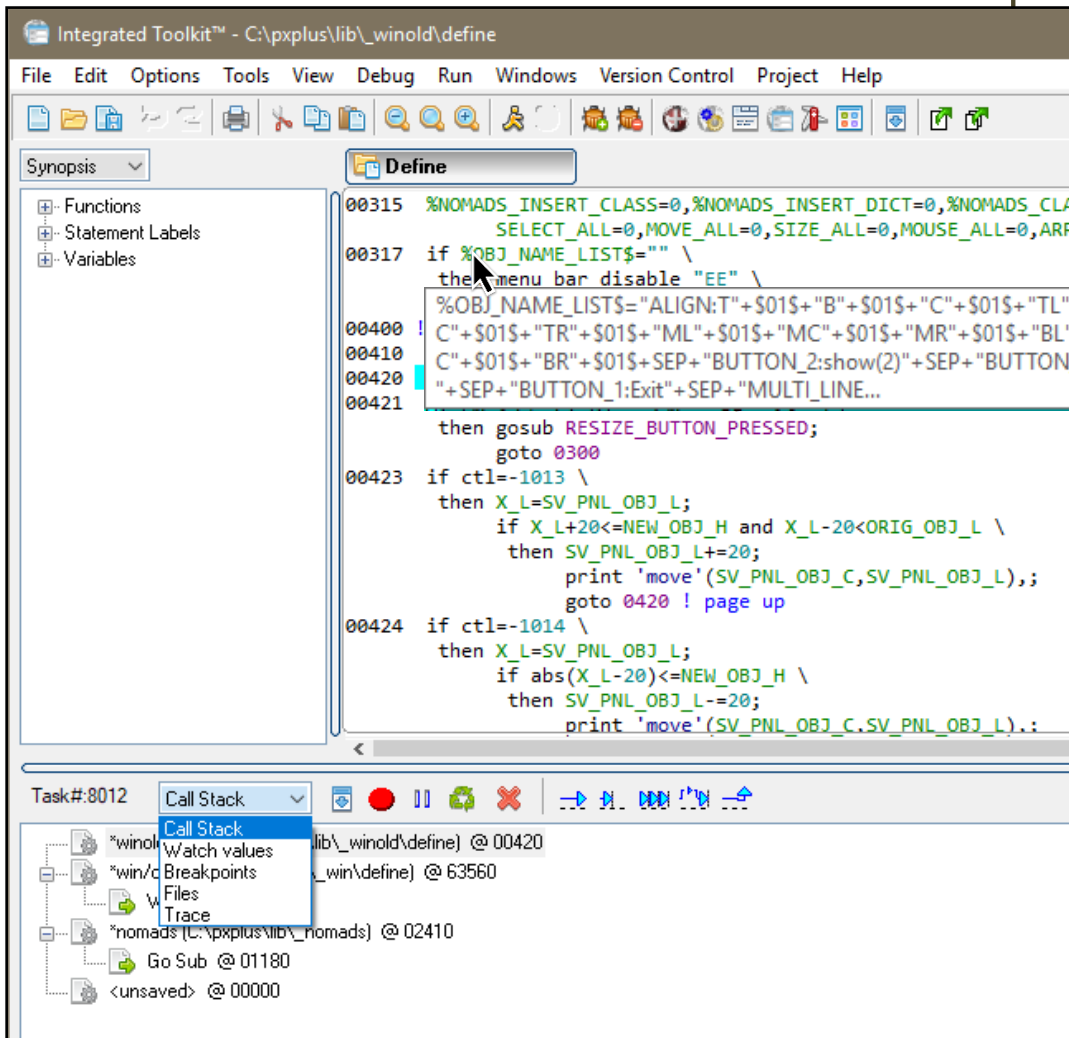
- Watch Values

- Breakpoints

- Trace

- Debug window can be docked or free standing

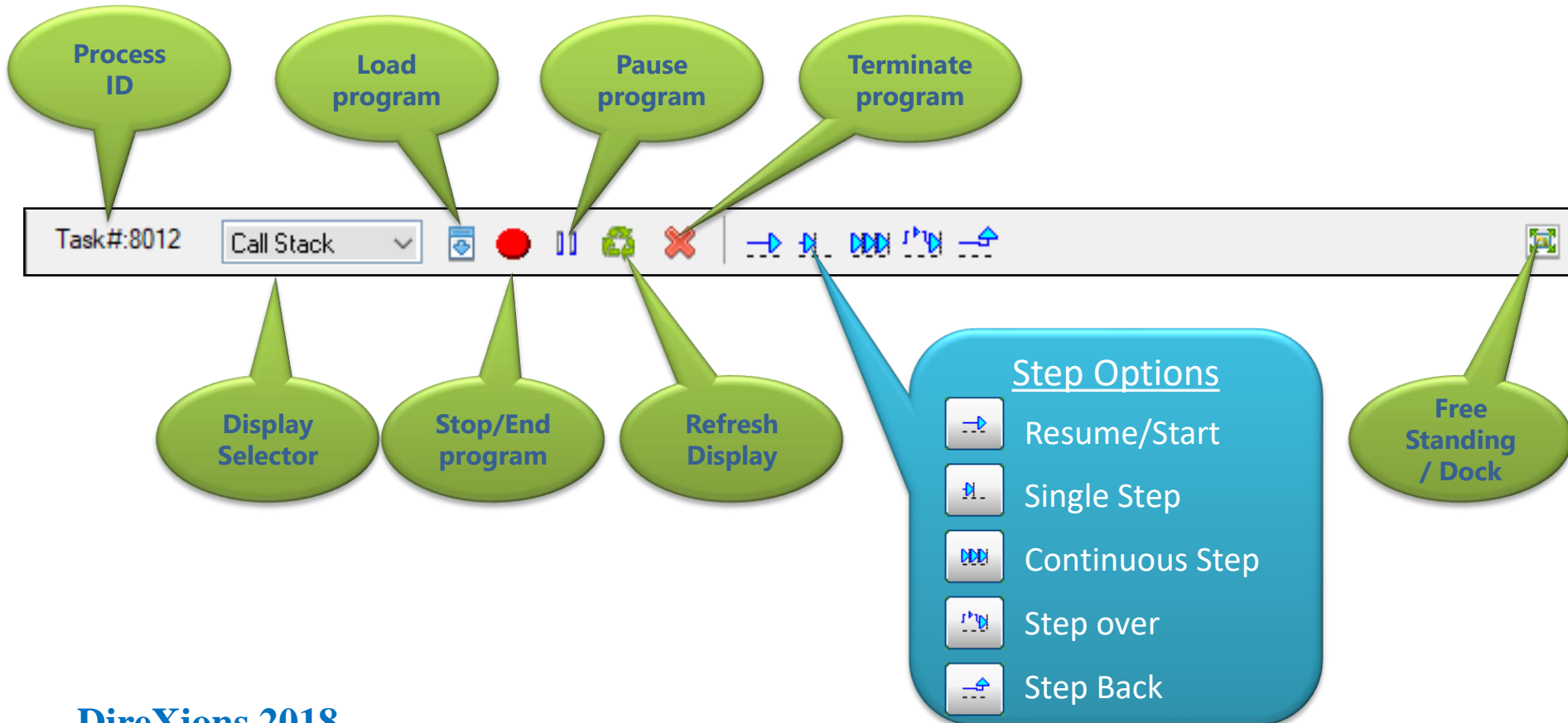
- Hover over any variable to see data



# PxPlus Built-in Debugger - \*IT

## Graphical Interface

- Control bar options:



# PxPlus Built-in Debugger - DBG

## Text Mode Interface

- Invoked by DBG command
  - Commands:

```
Connect procid      : Connect to process
Disconnect          : Disconnect from process
Tasks               : List known processes (Tasks * includes program)
Halt                : Halt/Suspend the process
Go                  : Resume the process
Execute xxxx        : Execute command within process
List [from [to]]    : List statement (option from/to)
Kill                : Terminate/kill the process
Print xxx           : Evaluate and show xxxx from process
Files               : Show files
Stack               : Show Call Stack
Where               : Where is the process
. xxx               : Step through program where xxx is:
                    :   Program - Till program change
                    :   Around  - Around next gosub/call/etc
                    :   Out    - Out of current program level
                    :   nnn    - 'nnn' Instructions
                    :   (if xxx omitted single step)
Quit                : Exits the debugger
```

All commands  
accept first  
character only  
as in "P X\$" to  
print X\$

# PxPlus Built-in Debugger - *i*Nomads

## Web Interface

- *i*Nomads includes a debug transaction

[http://\*server\*/inomads?txid=debug](http://server/inomads?txid=debug)

- Provides debug functionality in a web page
  - Contains various tabs for each option
    - **Process** Selection
    - Program **stack** contents
    - List of open **files** and status
    - Option to enable **trace** of logic
    - Text mode display of **console** output
  - *i*Nomads debug works with normal processes
    - Access not restricted to *i*Nomads tasks

# Debugging *iNomads* Applications

- Suggestions to make *iNomads* debugging easier
  - Run *iNomads* on local Windows desktop
    - Don't run as a service but as desktop application
  - Enabling Debugging in Admin configuration setup allows you to
    - Use Windows debug windows
      - Provides Tracing and Breakpoint capabilities
    - Add ESCAPE statements and step through code

## NOTES:

Browser will complain about being Non-Responsive  
If using EZWEB change 'timeout' value in ARG list

# POST MORTEM DEBUGGING

# Nomads Dump panel

- New Nomads/iNomads dump for PxPlus 2017
  - Custom HTML output provides
    - Error status
    - Program stack
    - Directory information
    - Nomads control information
    - All variables for all stack levels
      - **PERFORM**ed levels shown once
    - Global variable
    - File stats

# Jump back-trace

- Records all transfers of control
  - Same as Live jump trace in PxPlus 2018
    - Been around since version 10.10
  - Enabled by setting the 'TJ' system parameter
  - Value determines the number of transfers saved
  - The default value is 0 (no history)
  - Maximum is 10000 entries
  - Minimally invasive – low overhead



# Jump back-trace

- Special directive to view the jump history:

## DUMP GOTO

- Output of DUMP GOTO directive
  - From/To line number
  - From/To program
  - Reason for transfer (i.e. directive that caused transfer)
  - Loop count for repeated transfers
  - ERR value if transfer caused by error branch

```
->DUMP GOTO
From Dir To      Loops  Err From           To
-----
00100 GOTO 00200           C:\PVX\GotoTest
00200 GOTO 00300           C:\PVX\GotoTest
```

# Jump back-trace

- How can it help debug programs?
  - Shows where program has been
  - Allows developer to see what execution flow
  - Reduces trace output to only transfers
  - Avoids pouring over repetitive lines of code
  - Provides ERR feedback when error transfer taken
    - Answers question:  
*Was function/method called?*

# Error History

- A solution to another eternal question:

**“Am I looking at the right error?”**

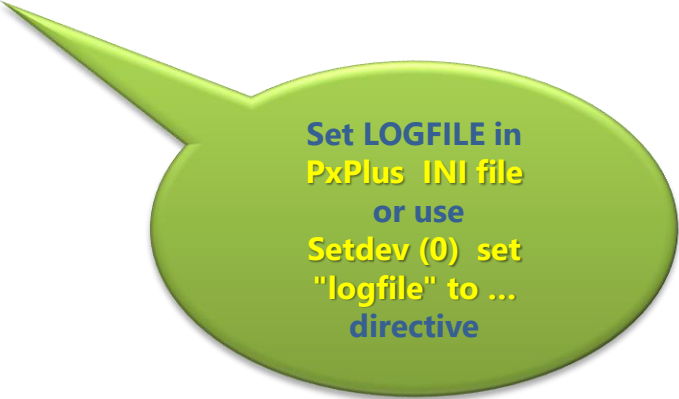
- Errors often get cascaded back
- Actual error can occur at a lower level
- Error information can be 'tainted' by error handling
  - An error in the Error Handler will overwrite ERR information

# Error History

- How to enable the Error History
  - Enabled by setting the 'EH' system parameter
    - Parameter value defines number of history entries saved
      - Default value is 0 (no history maintained)
      - Maximum is 100
      - History wraps around once limit is reached
- How to display the Error History
  - Accessed using the ERR function
  - Optional second parameter contains entry number
    - ERR(keyword, **1**) returns most recent error
    - ERR(keyword, **2**) returns prior error
    - ERR(keyword, **3**) returns error before that, etc..
  - Standard ERR(keyword) information is maintained for each entry

# Software Failure (GPF etc.)

- Most Errors are trapped internally BUT..
  - Both Windows and Linux have fatal error traps
  - If LOGFILE is present they output:
    - Date and time of failure
    - Current program and statement
    - ERR, CTL, RET, LFA, and LFO values
    - Last accessed file pathname
    - Program Stack
    - Open files and state
    - ERR() information
      - Includes ERROR HISTORY if enabled
    - Jump History if enabled



Set LOGFILE in  
PxPlus INI file  
or use  
Setdev (0) set  
"logfile" to ...  
directive

# Software Failure

- Typical output

## Software Failure Output

```
Apr 19 15:30:40 [<crash>:10420] Fatal exception error code c0000005 occurred (PID:5900)
Apr 19 15:30:40 [<crash>:10420]   - ERR=0 CTL=0 RET=258 LFA=0 LFO=63 (PID:5900)
Apr 19 15:30:40 [<crash>:10420]   - Last path used: C:\pxplus\lib\_windx.utl (PID:5900)
Apr 19 15:30:40 [<crash>:10420] --- Program stack --- (PID:5900)
Apr 19 15:30:40 [<crash>:10420]   - STK(0) = line 10420 in C:\pxplus\windx\windx (PID:5900)
Apr 19 15:30:40 [<crash>:10420]   - GO SUB... 01150 (PID:5900)
Apr 19 15:30:40 [<crash>:10420] --- Files --- (PID:5900)
Apr 19 15:30:40 [<crash>:10420]   - PTH(0) = CON (PID:5900)
Apr 19 15:30:40 [<crash>:10420]   - PTH(2) = CON (PID:5900)
Apr 19 15:30:40 [<crash>:10420]   - PTH(127) = [tcp]sue;4093;STREAM;NODELAY; (PID:5900)
Apr 19 15:30:40 [<crash>:10420] --- Err() Info --- (PID:5900)
Apr 19 15:30:40 [<crash>:10420]           Current Err:12 Prog:  Stno: 64999 LFA: 0
```

# Software Failure & Windows

- Final Word of note
  - Just because Windows says failure occurred running PxPlus doesn't mean PxPlus caused it
    - Windows reports the main EXE for the process
    - We call external modules to execute many functions if they fail/crash it gets reported as PxPlus
  - When debugging system failures we ask you to:
    - Try disabling Anti-virus if present
    - Try on a different machine (where possible)
    - Try as a desktop application if running as service

# Review

- You can debug background tasks
  - \*IT, DBG and even *iNomads* can be used
- Run *iNomads* on Windows to debug
- Jump trace shows how you got here
- Error history helps you find TRUE error
- System log helps on Software failures

Old School still works ...

But you can always learn  
a few new tricks





# Additional Resources

The help link(s) below refer to the current on-line help pages. The functionality may have been further updated since the PxPlus 2018 (version 15) release.

- [Windows Debugging Environment](#)
- [Trace IF Status, 'IT' Parameter](#)
- [IT Debug Facility](#)
- [DBG Console Command](#)
- [iNomads DEBUG Transaction](#)
- [Debug Panel](#)
- [Jump Trace, 'TJ' Parameter](#)
- [Logfile](#)
- [ESCAPE, SETTRACE, DUMP](#)
- [ERR\(\), 'EH' Parameter](#)