z/XPF®

High Definition Profiling

Release Guide

Duke Software www.duke-software.com

For Release V2R2m17

Revision date: 11/12/2013

PREFACE

PROPRIETARY LEGEND

Contact Information

Phone 281-395-5570
E-Mail: David.Day@duke-software.com

Please visit www.duke-software.com for the following services:

- General information about z/XPF
- E-mail links to Marketing and Customer service

Table of Contents

Welcome to the z/XPF V2R2M17 Release Guide 1
System Requirements 1
What's new in z/XPF Release V2R2M17 2
Time Segmenting - An Overview: 2
CPU Consumption - An Overview. 2
Time Segmenting - In Detail 2
CPU Consumption - In Detail 5
Who is Dave Day? 12

Welcome to the z/XPF V2R2M17 Release Guide

Greetings from ColeSoft Marketing, Inc. We'd like to inform you about the latest release of z/XPF Version 2, Release 2 Modification Level 17.

This Release Guide is intended to orient you to the exciting new features and capabilities in z/XPF Release V2R2M17. A great deal of work has been done to expand z/XPF's unconventional architecture to make even more powerful for you.

What is z/XPF?

Very briefly, Dave Day's z/XPF is the world's first "Event" profiler. Other profilers are "State" profilers – very good products that periodically stop/start address spaces to gather the state of programs.

Instead of using a "state" approach, z/XPF monitors Trace records from the processor Trace Table, filtering Trace records by address space, and collating them into a time-stamped log stream. Because Trace Records are generated for interrupts and other events (I/O, SVC, Program Call/Return/ Transfer, Contention, Memory management, etc.) z/XPF can capture many millions of data points – far more than the thousands one might expect from "State" profilers.

Further, z/XPF's Event profiling architecture is almost completely passive. There is NO presence in the target address space. There are no hooks to the operating system (we DO monitor two SMF exits). The data is just THERE, and z/XPF merely captures it. It's an intriguing architecture, and early customers have been both surprised and pleased with the "bones" of the product.

System Requirements

z/XPF V2R2 will execute on z/OS Release 1.10 and above.

Be advised that data capture datasets created with an earlier release of z/XPF will NOT be read by z/XPF V2R2M17.

What’s new in z/XPF Release V2R2M17

z/XPF’s Summary reporting has been enhanced in two major ways:

More granular Time Segmenting, and
A novel new way of measuring CPU consumption.

Time Segmenting - An Overview:

z/XPF allows you to separate your report into Time Segments, thereby isolating specific time periods in your reports from all others. In the earlier releases you could have one, single Time Segment or up to ten evenly divided ones. NOW, you can further subdivide your reports from 60-second Time Segments down to *one-second Time Segments*. Once you create more granular Time Segments you can use z/XPF’s Summary Reports on *each individual Time Segment*. This yields an unprecedented level of granularity in Summary Reporting.

CPU Consumption - An Overview

Most profiling products claim to report on “CPU Consumption”, but actually do NOT do this. Instead they observe PSWs and conclude that over the length of the report that this-or-that PSW is seen the most, and therefore is consuming most of the CPU. That’s how all the other profilers interpret to their customers - and they have been around a LONG time.

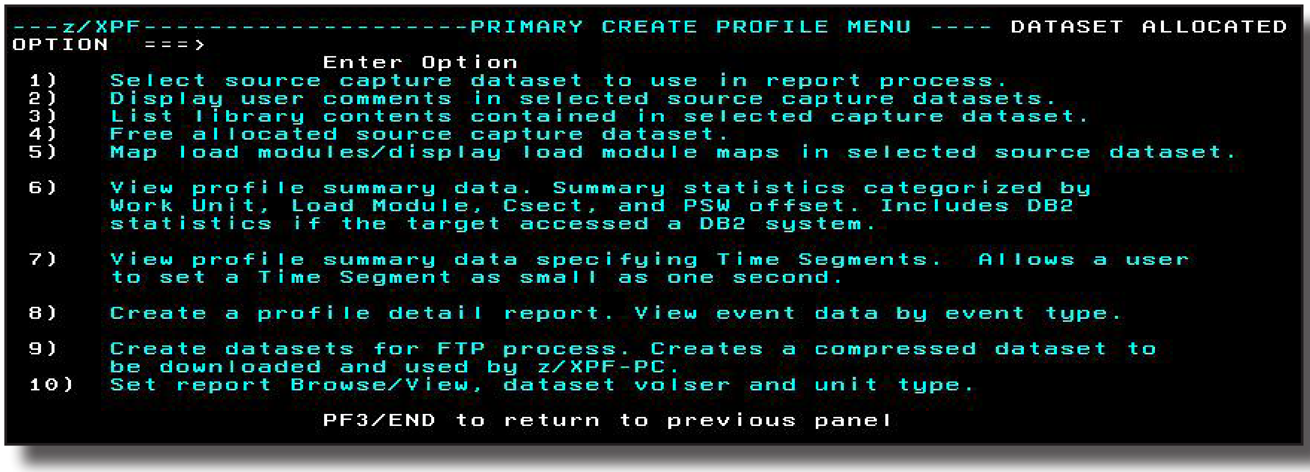
We tried “selling” this concept to a well-respected customer, whose programmer said, “This is NOT CPU Consumption. It is merely a report of the most frequently observed PSWs. What if a particular PSW is a call to a system service? Where does THAT lead?”

After thinking about this, we had to agree, which is why z/XPF calls that report, “Most Frequently Observed PSWs”. Now at least we weren’t LYING, but it stayed with Dave Day. There HAD to be a better way.

Dave Day has engineered that new approach into z/XPF. We think it’s completely novel and perhaps the ONLY true measurement of CPU consumption. That puts z/XPF WAY out in front of any other profiler, and we’ll give you much more details on it in the latter part of this book.

Time Segmenting - In Detail

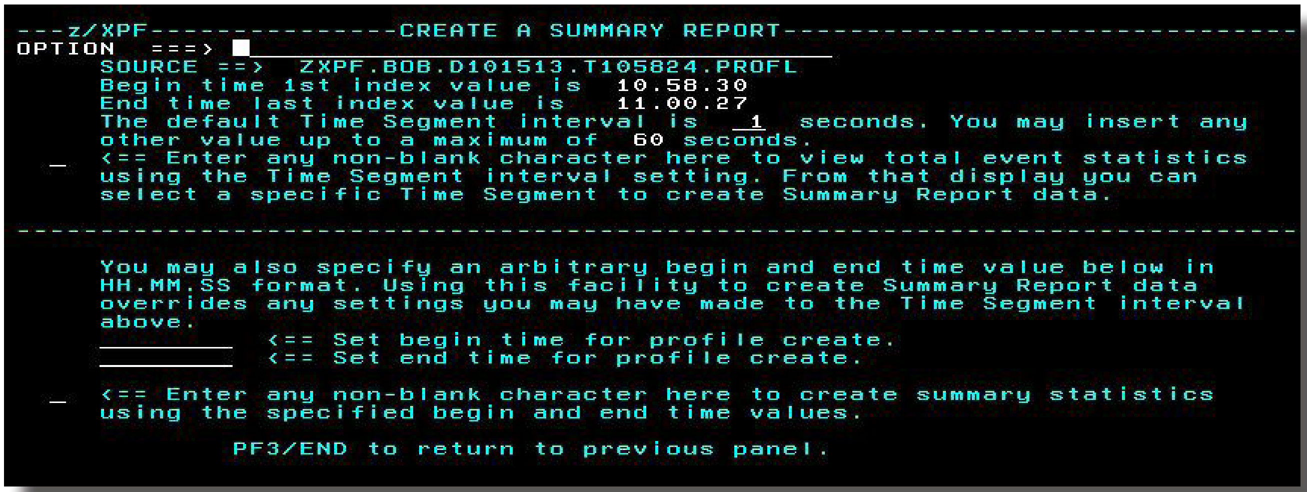
z/XPF no longer limits the user to only ten Time Segments within a report. Please examine Option 7 in Figure 1 below:



```
---z/XPF--------------------PRIMARY CREATE PROFILE MENU ---- DATASET ALLOCATED
OPTION  ===>
                       Enter Option
  1)    Select source capture dataset to use in report process.
  2)    Display user comments in selected source capture datasets.
  3)    List library contents contained in selected capture dataset.
  4)    Free allocated source capture dataset.
  5)    Map load modules/display load module maps in selected source dataset.

  6)    View profile summary data. Summary statistics categorized by
        Work Unit, Load Module, Csect, and PSW offset. Includes DB2
        statistics if the target accessed a DB2 system.

  7)    View profile summary data specifying Time Segments.  Allows a user
        to set a Time Segment as small as one second.

  8)    Create a profile detail report. View event data by event type.

  9)    Create datasets for FTP process. Creates a compressed dataset to
        be downloaded and used by z/XPF-PC.
 10)    Set report Browse/View, dataset volser and unit type.

                       PF3/END to return to previous panel
```

Figure 1.

Once you pick Option 7, you will see the panel below, in Figure 2:



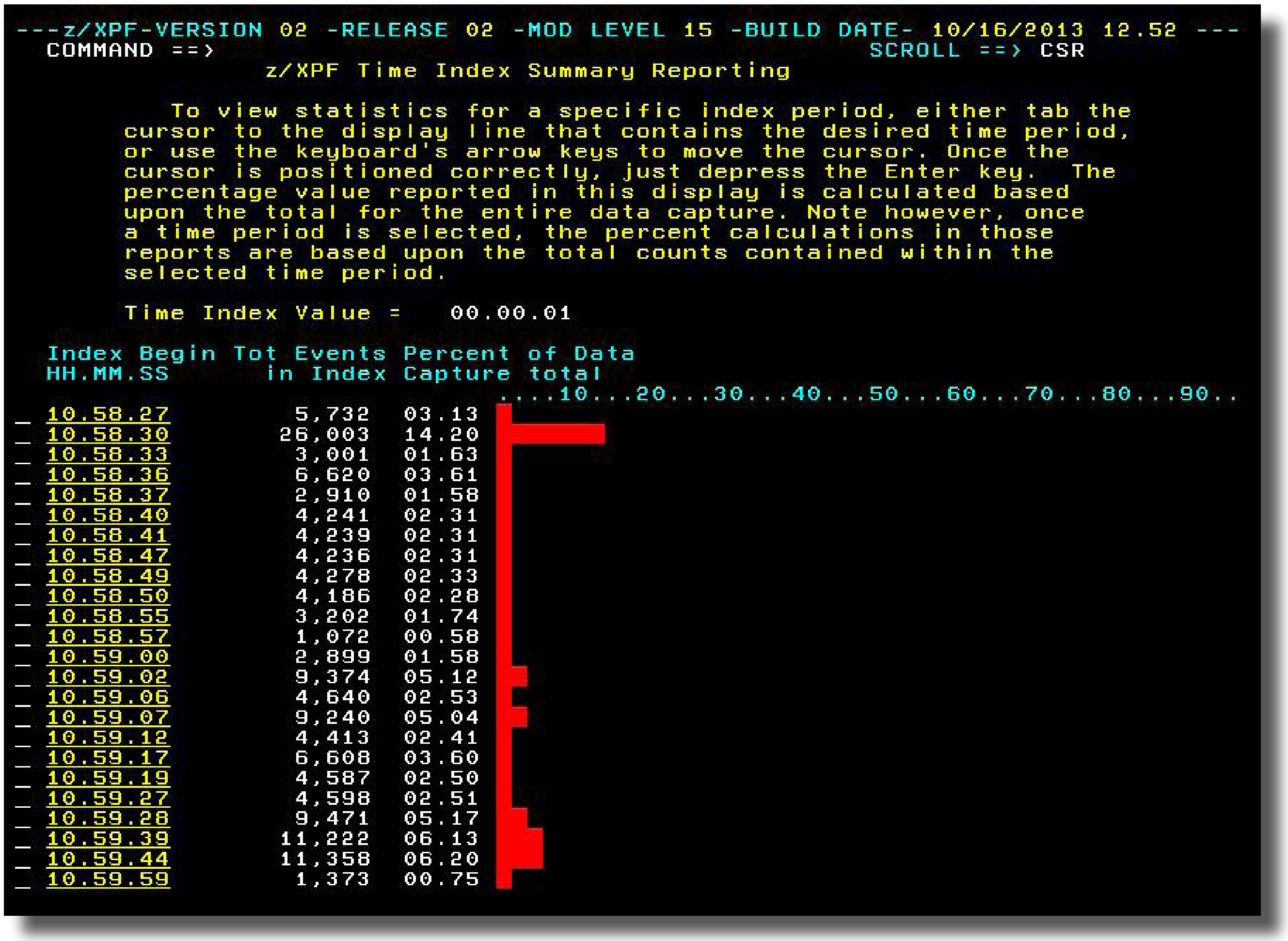
```
---z/XPF----------------CREATE A SUMMARY REPORT-------------------------------
OPTION  ===> 
     SOURCE ==>  ZXPF.BOB.D101513.T105824.PROFL
     Begin time 1st index value is   10.58.30
     End time last index value is    11.00.27
     The default Time Segment interval is   1  seconds. You may insert any
     other value up to a maximum of  60 seconds.
  _  <== Enter any non-blank character here to view total event statistics
     using the Time Segment interval setting. From that display you can
     select a specific Time Segment to create Summary Report data.

------------------------------------------------------------------------------

     You may also specify an arbitrary begin and end time value below in
     HH.MM.SS format. Using this facility to create Summary Report data
     overrides any settings you may have made to the Time Segment interval
     above.
     ________  <== Set begin time for profile create.
     ________  <== Set end time for profile create.

  _  <== Enter any non-blank character here to create summary statistics
     using the specified begin and end time values.

             PF3/END to return to previous panel.
```

Figure 2.

This screen is divided into two logical parts: You can alter the Time Segment values from 1 second to up to 60 seconds, OR (using the bottom portion of the panel) you can specify any single time period within your report. Be aware that changes you make in one section of the panel will over-ride choices you make in the other one. In other words, choose one set of criteria or the other - you can't have both.

When this choice has been made, z/XPF will process your report and allow you to perform Summary Reporting on each individual Time Segment (with the maximum value of 60 seconds down to 1-second Time Segments).

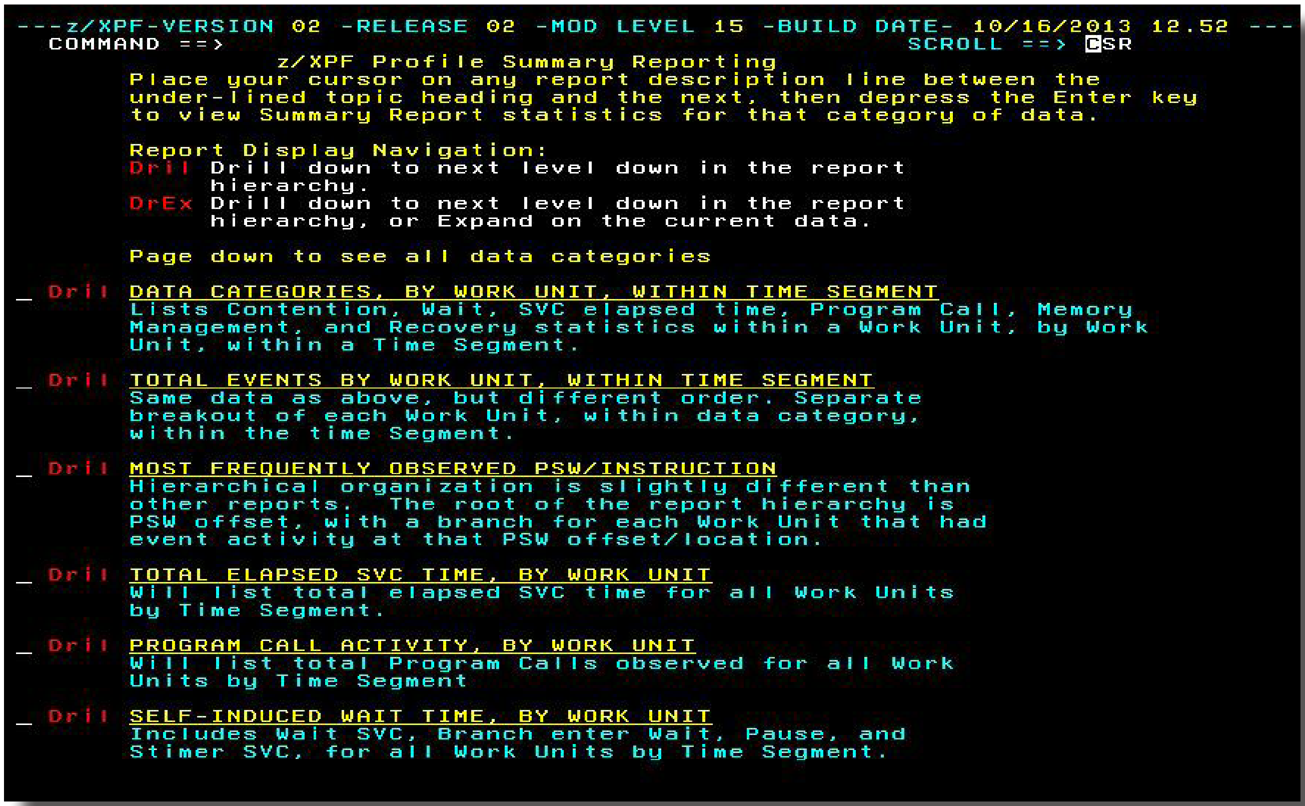
I'll allow a one-second time interval for a z/XPF data capture, and put "S" into the input field on the top portion of the screen. When I do so, z/XPF parses out my report and shows me the panel below, in Figure 5-3:



```
---z/XPF-VERSION 02 -RELEASE 02 -MOD LEVEL 15 -BUILD DATE- 10/16/2013 12.52 ---
  COMMAND ==>                                                  SCROLL ==> CSR
                  z/XPF Time Index Summary Reporting

          To view statistics for a specific index period, either tab the
        cursor to the display line that contains the desired time period,
        or use the keyboard's arrow keys to move the cursor. Once the
        cursor is positioned correctly, just depress the Enter key.  The
        percentage value reported in this display is calculated based
        upon the total for the entire data capture. Note however, once
        a time period is selected, the percent calculations in those
        reports are based upon the total counts contained within the
        selected time period.

        Time Index Value =    00.00.01

  Index Begin Tot Events Percent of Data
  HH.MM.SS        in Index Capture total
                                 ....10...20...30...40...50...60...70...80...90..
_ 10.58.27          5,732  03.13
_ 10.58.30         26,003  14.20
_ 10.58.33          3,001  01.63
_ 10.58.36          6,620  03.61
_ 10.58.37          2,910  01.58
_ 10.58.40          4,241  02.31
_ 10.58.41          4,239  02.31
_ 10.58.47          4,236  02.31
_ 10.58.49          4,278  02.33
_ 10.58.50          4,186  02.28
_ 10.58.55          3,202  01.74
_ 10.58.57          1,072  00.58
_ 10.59.00          2,899  01.58
_ 10.59.02          9,374  05.12
_ 10.59.06          4,640  02.53
_ 10.59.07          9,240  05.04
_ 10.59.12          4,413  02.41
_ 10.59.17          6,608  03.60
_ 10.59.19          4,587  02.50
_ 10.59.27          4,598  02.51
_ 10.59.28          9,471  05.17
_ 10.59.39         11,222  06.13
_ 10.59.44         11,358  06.20
_ 10.59.59          1,373  00.75
```

Figure 3.

If I cared to, I could page down through the panel to see more of my one-second time segments until I see “Bottom of Data”, but I’ll just select the second time slice (which seems to have some activity) and press Enter. z/XPF analyzes that one second Time Segment and shows me a Summary Report panel as below, in Figure 5-4:



```
---z/XPF-VERSION 02 -RELEASE 02 -MOD LEVEL 15 -BUILD DATE- 10/16/2013 12.52 ---
   COMMAND ==>                                                 SCROLL ==> CSR
                   z/XPF Profile Summary Reporting
          Place your cursor on any report description line between the
          under-lined topic heading and the next, then depress the Enter key
          to view Summary Report statistics for that category of data.

          Report Display Navigation:
          Dril Drill down to next level down in the report
               hierarchy.
          DrEx Drill down to next level down in the report
               hierarchy, or Expand on the current data.

          Page down to see all data categories

_    Dril DATA CATEGORIES, BY WORK UNIT, WITHIN TIME SEGMENT
          Lists Contention, Wait, SVC elapsed time, Program Call, Memory
          Management, and Recovery statistics within a Work Unit, by Work
          Unit, within a Time Segment.

_    Dril TOTAL EVENTS BY WORK UNIT, WITHIN TIME SEGMENT
          Same data as above, but different order. Separate
          breakout of each Work Unit, within data category,
          within the time Segment.

_    Dril MOST FREQUENTLY OBSERVED PSW/INSTRUCTION
          Hierarchical organization is slightly different than
          other reports.  The root of the report hierarchy is
          PSW offset, with a branch for each Work Unit that had
          event activity at that PSW offset/location.

_    Dril TOTAL ELAPSED SVC TIME, BY WORK UNIT
          Will list total elapsed SVC time for all Work Units
          by Time Segment.

_    Dril PROGRAM CALL ACTIVITY, BY WORK UNIT
          Will list total Program Calls observed for all Work
          Units by Time Segment

_    Dril SELF-INDUCED WAIT TIME, BY WORK UNIT
          Includes Wait SVC, Branch enter Wait, Pause, and
          Stimer SVC, for all Work Units by Time Segment.
```

Figure 4.

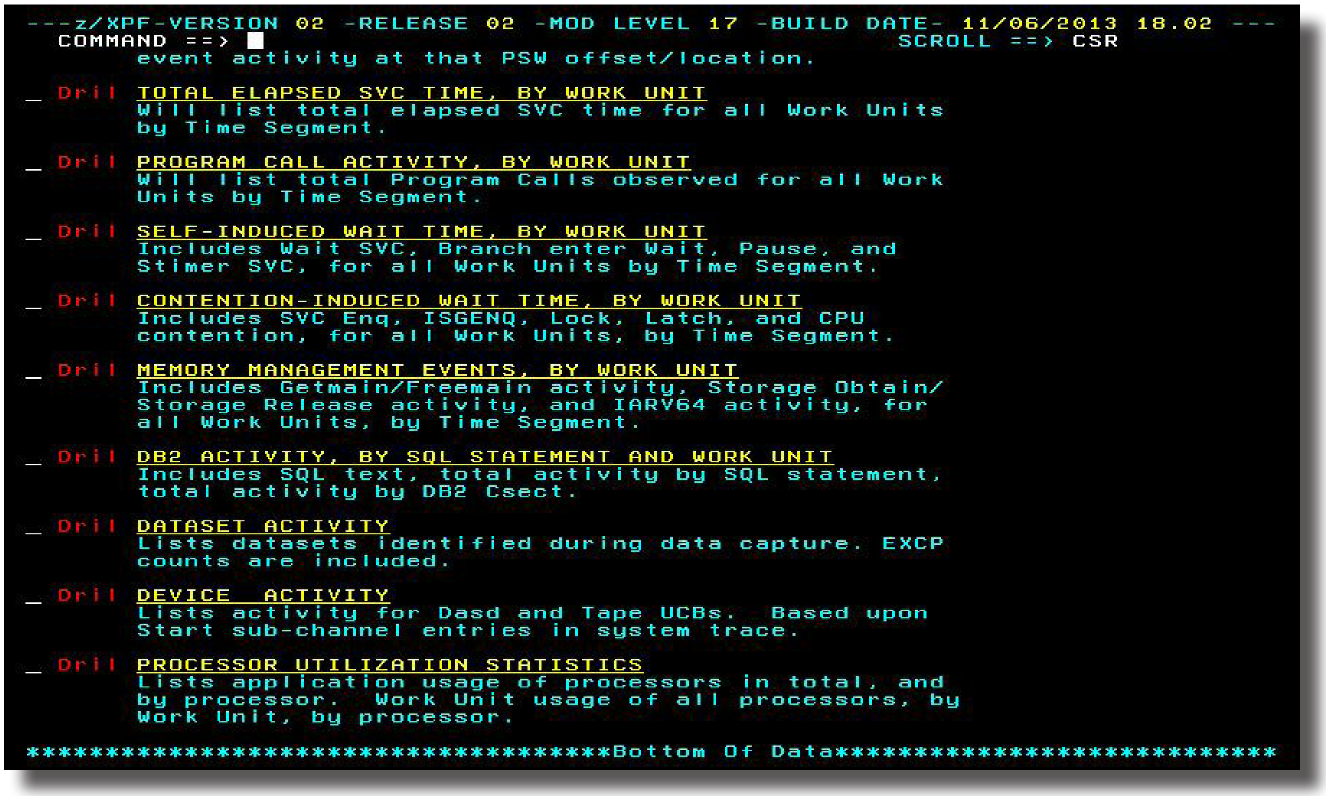
If you are using z/XPF V2R2, then this will look entirely familiar to you. You can drill down or expand just as you're used to. The difference is in the granularity of the Time Segment. You may be looking at a 60-, 45-, 20- or 1-second Time Segment. Good stuff!

CPU Consumption - In Detail

Dave Day has found what we believe to be a TRUE measurement of CPU consumption. Other profilers measure the frequency with which particular PSWs appear during sampling, and label this as "CPU Consumption". That's not really so. It's merely a report of PSWs,which fails to account for downstream calls from any PSW to other code, or systems services. But that's how "CPU Consumption has been sold to the user community for a generation.

Dave Day has figured out a way to measure true CPU Consumption by using an obscure z/OS control block and the measurement of Timer Interrupts. It's a proprietary technique that we'll discuss with our users verbally but at this time don't care to publish. With that out of the way, here's a sample "drill down' through this new report.

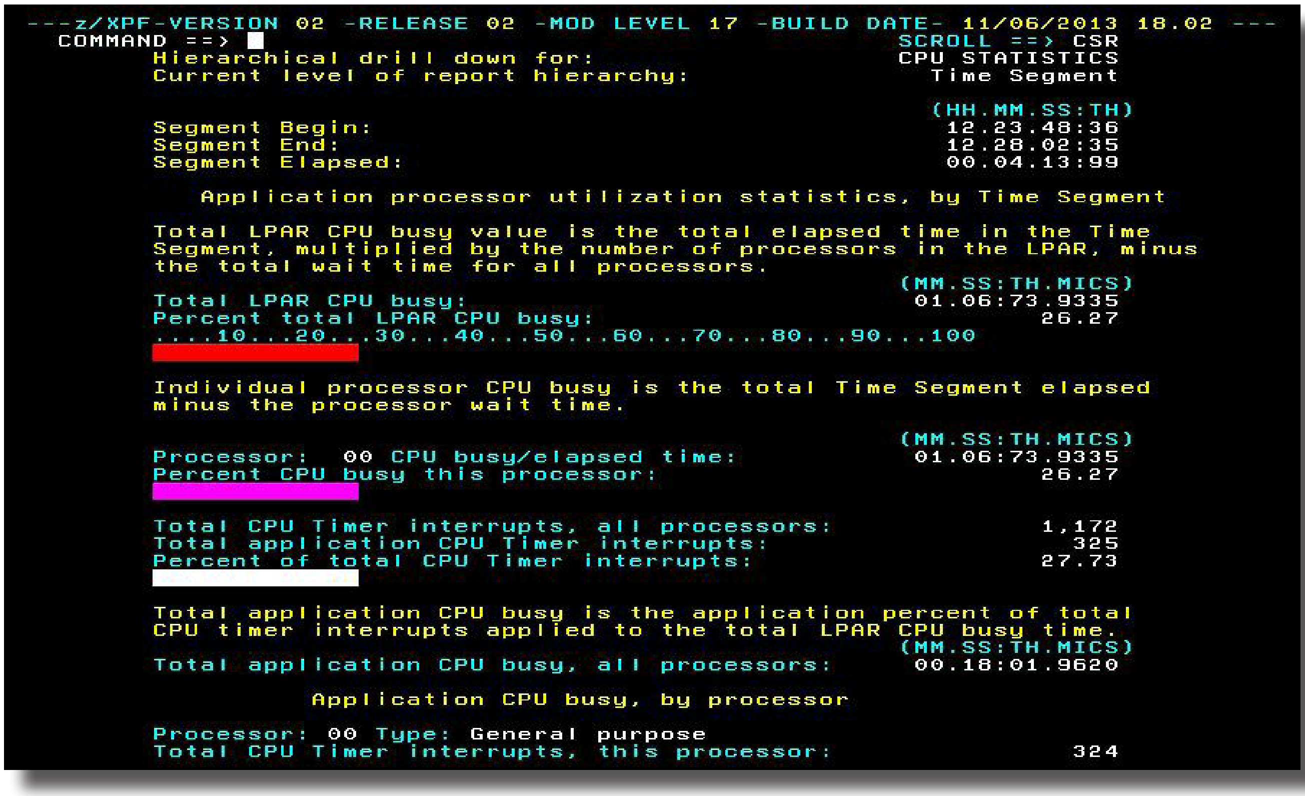
When I ask for Summary Reporting (Option 6 from the Primary Create Profile Menu), and parse out my report, I am shown all the various Summary Reports available. If I page down, I can see the entry marked "Processor Utilization Statistics" below, in Figure 5:



```
---z/XPF-VERSION 02 -RELEASE 02 -MOD LEVEL 17 -BUILD DATE- 11/06/2013 18.02 ---
   COMMAND ==>                                              SCROLL ==> CSR
         event activity at that PSW offset/location.

 _ Dril  TOTAL ELAPSED SVC TIME, BY WORK UNIT
         Will list total elapsed SVC time for all Work Units
         by Time Segment.

 _ Dril  PROGRAM CALL ACTIVITY, BY WORK UNIT
         Will list total Program Calls observed for all Work
         Units by Time Segment.

 _ Dril  SELF-INDUCED WAIT TIME, BY WORK UNIT
         Includes Wait SVC, Branch enter Wait, Pause, and
         Stimer SVC, for all Work Units by Time Segment.

 _ Dril  CONTENTION-INDUCED WAIT TIME, BY WORK UNIT
         Includes SVC Enq, ISGENQ, Lock, Latch, and CPU
         contention, for all Work Units, by Time Segment.

 _ Dril  MEMORY MANAGEMENT EVENTS, BY WORK UNIT
         Includes Getmain/Freemain activity, Storage Obtain/
         Storage Release activity, and IARV64 activity, for
         all Work Units, by Time Segment.

 _ Dril  DB2 ACTIVITY, BY SQL STATEMENT AND WORK UNIT
         Includes SQL text, total activity by SQL statement,
         total activity by DB2 Csect.

 _ Dril  DATASET ACTIVITY
         Lists datasets identified during data capture. EXCP
         counts are included.

 _ Dril  DEVICE  ACTIVITY
         Lists activity for Dasd and Tape UCBs.  Based upon
         Start sub-channel entries in system trace.

 _ Dril  PROCESSOR UTILIZATION STATISTICS
         Lists application usage of processors in total, and
         by processor.  Work Unit usage of all processors, by
         Work Unit, by processor.

 ************************************Bottom Of Data******************************
```

Figure 5.

I select this choice by either placing the cursor on that line and pressing Enter, or by putting a “D” next to the “Dril” statement, and I’m shown the panel in Figure 5-6.

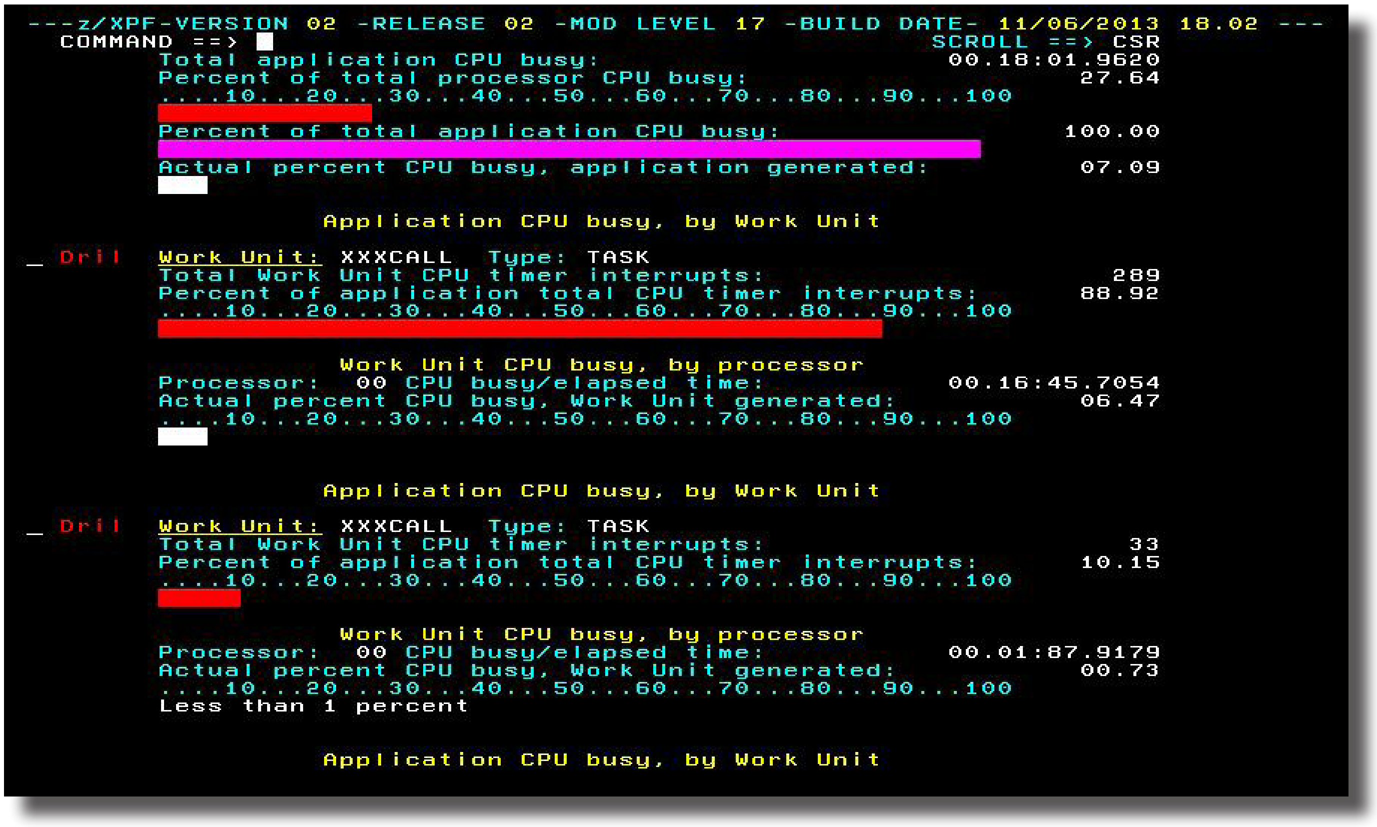


```
 ---z/XPF-VERSION 02 -RELEASE 02 -MOD LEVEL 17 -BUILD DATE- 11/06/2013 18.02 ---
   COMMAND ==>                                                   SCROLL ==> CSR
         Hierarchical drill down for:                            CPU STATISTICS
         Current level of report hierarchy:                        Time Segment

                                                                  (HH.MM.SS:TH)
         Segment Begin:                                            12.23.48:36
         Segment End:                                              12.28.02:35
         Segment Elapsed:                                          00.04.13:99

            Application processor utilization statistics, by Time Segment

         Total LPAR CPU busy value is the total elapsed time in the Time
         Segment, multiplied by the number of processors in the LPAR, minus
         the total wait time for all processors.
                                                                (MM.SS:TH.MICS)
         Total LPAR CPU busy:                                    01.06:73.9335
         Percent total LPAR CPU busy:                                    26.27
         ....10...20...30...40...50...60...70...80...90...100


         Individual processor CPU busy is the total Time Segment elapsed
         minus the processor wait time.

                                                                (MM.SS:TH.MICS)
         Processor:  00 CPU busy/elapsed time:                   01.06:73.9335
         Percent CPU busy this processor:                                26.27


         Total CPU Timer interrupts, all processors:                     1,172
         Total application CPU Timer interrupts:                           325
         Percent of total CPU Timer interrupts:                          27.73


         Total application CPU busy is the application percent of total
         CPU timer interrupts applied to the total LPAR CPU busy time.
                                                                (MM.SS:TH.MICS)
         Total application CPU busy, all processors:             00.18:01.9620

                   Application CPU busy, by processor

         Processor: 00 Type: General purpose
         Total CPU Timer interrupts, this processor:                       324
```

Figure 6.

Figure 5-6 shows overview information for all processors on the system. In our case we have only one processor which shows as “Processor 00”. So, in my example Individual processor busy (26.27%) precisely matches “Total LPAR CPU busy time”. At the bottom of the panel you’ll see the toral number of CPU Timer Interrupts for this Time Segment. We expect that in YOUR environment you’ll see more processors and a good deal more CPU Timer interrupts.

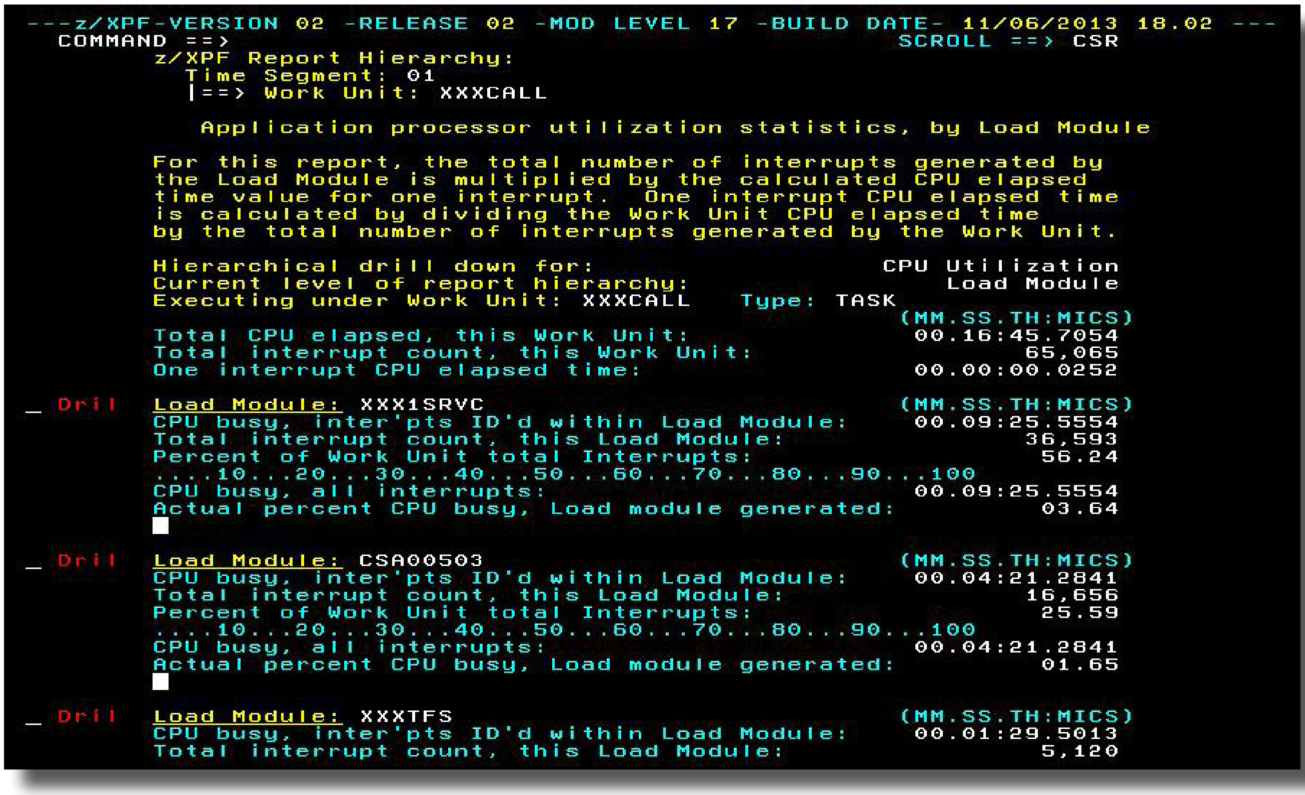
If I page down from here, I’ll see the display shown in Figure 7 below:



```
---z/XPF-VERSION 02 -RELEASE 02 -MOD LEVEL 17 -BUILD DATE- 11/06/2013 18.02 ---
  COMMAND ==>                                              SCROLL ==> CSR
        Total application CPU busy:                         00.18:01.9620
        Percent of total processor CPU busy:                        27.64
        ....10...20...30...40...50...60...70...80...90...100

        Percent of total application CPU busy:                     100.00

        Actual percent CPU busy, application generated:             07.09


                  Application CPU busy, by Work Unit

 _ Dril Work Unit: XXXCALL  Type: TASK
        Total Work Unit CPU timer interrupts:                         289
        Percent of application total CPU timer interrupts:          88.92
        ....10...20...30...40...50...60...70...80...90...100


                   Work Unit CPU busy, by processor
        Processor:  00 CPU busy/elapsed time:               00.16:45.7054
        Actual percent CPU busy, Work Unit generated:               06.47
        ....10...20...30...40...50...60...70...80...90...100


                  Application CPU busy, by Work Unit

 _ Dril Work Unit: XXXCALL  Type: TASK
        Total Work Unit CPU timer interrupts:                          33
        Percent of application total CPU timer interrupts:          10.15
        ....10...20...30...40...50...60...70...80...90...100


                   Work Unit CPU busy, by processor
        Processor:  00 CPU busy/elapsed time:               00.01:87.9179
        Actual percent CPU busy, Work Unit generated:               00.73
        ....10...20...30...40...50...60...70...80...90...100
        Less than 1 percent


                  Application CPU busy, by Work Unit
```

Figure 7.

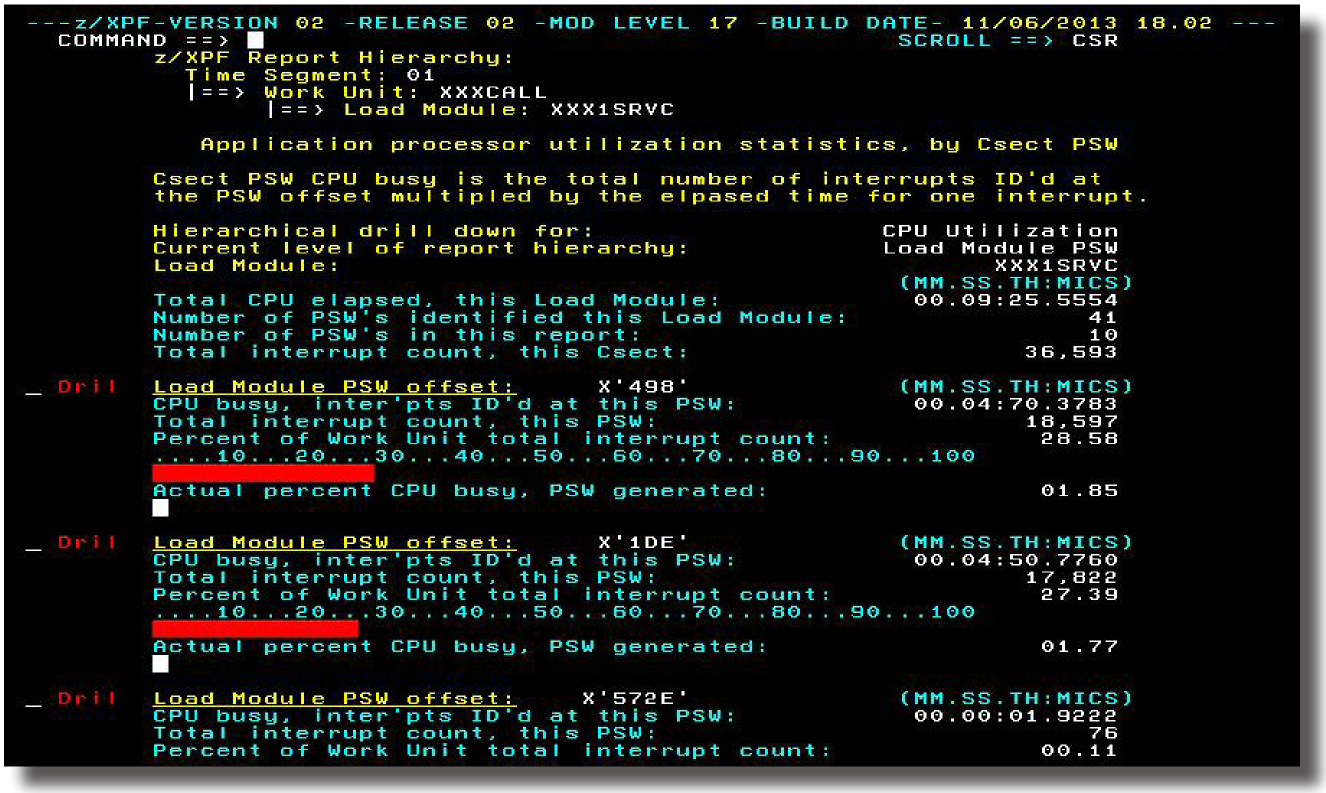
In Figure 7 I have a continuation of the “overview” information. Below that I see Work Units for the Time Segment with the biggest CPU Consumer sorted to the top. In this case, ti’s XXXCALL, a Task that used 289 of the 324 Timer Interrupts observed in the Time Segment. Let’s drill down again, and we’ll see Figure 8, below:



```
 ---z/XPF-VERSION 02 -RELEASE 02 -MOD LEVEL 17 -BUILD DATE- 11/06/2013 18.02 ---
   COMMAND ==>                                               SCROLL ==> CSR
          z/XPF Report Hierarchy:
            Time Segment: 01
             |==> Work Unit: XXXCALL

             Application processor utilization statistics, by Load Module

          For this report, the total number of interrupts generated by
          the Load Module is multiplied by the calculated CPU elapsed
          time value for one interrupt.  One interrupt CPU elapsed time
          is calculated by dividing the Work Unit CPU elapsed time
          by the total number of interrupts generated by the Work Unit.

          Hierarchical drill down for:                      CPU Utilization
          Current level of report hierarchy:                    Load Module
          Executing under Work Unit: XXXCALL    Type: TASK
                                                            (MM.SS.TH:MICS)
          Total CPU elapsed, this Work Unit:                 00.16:45.7054
          Total interrupt count, this Work Unit:                    65,065
          One interrupt CPU elapsed time:                    00.00:00.0252

 _ Dril   Load Module: XXX1SRVC                             (MM.SS.TH:MICS)
          CPU busy, inter'pts ID'd within Load Module:       00.09:25.5554
          Total interrupt count, this Load Module:                  36,593
          Percent of Work Unit total Interrupts:                     56.24
          ....10...20...30...40...50...60...70...80...90...100
          CPU busy, all interrupts:                          00.09:25.5554
          Actual percent CPU busy, Load module generated:            03.64
          █

 _ Dril   Load Module: CSA00503                             (MM.SS.TH:MICS)
          CPU busy, inter'pts ID'd within Load Module:       00.04:21.2841
          Total interrupt count, this Load Module:                  16,656
          Percent of Work Unit total Interrupts:                     25.59
          ....10...20...30...40...50...60...70...80...90...100
          CPU busy, all interrupts:                          00.04:21.2841
          Actual percent CPU busy, Load module generated:            01.65
          █

 _ Dril   Load Module: XXXTFS                               (MM.SS.TH:MICS)
          CPU busy, inter'pts ID'd within Load Module:       00.01:29.5013
          Total interrupt count, this Load Module:                   5,120
```

Figure 8.

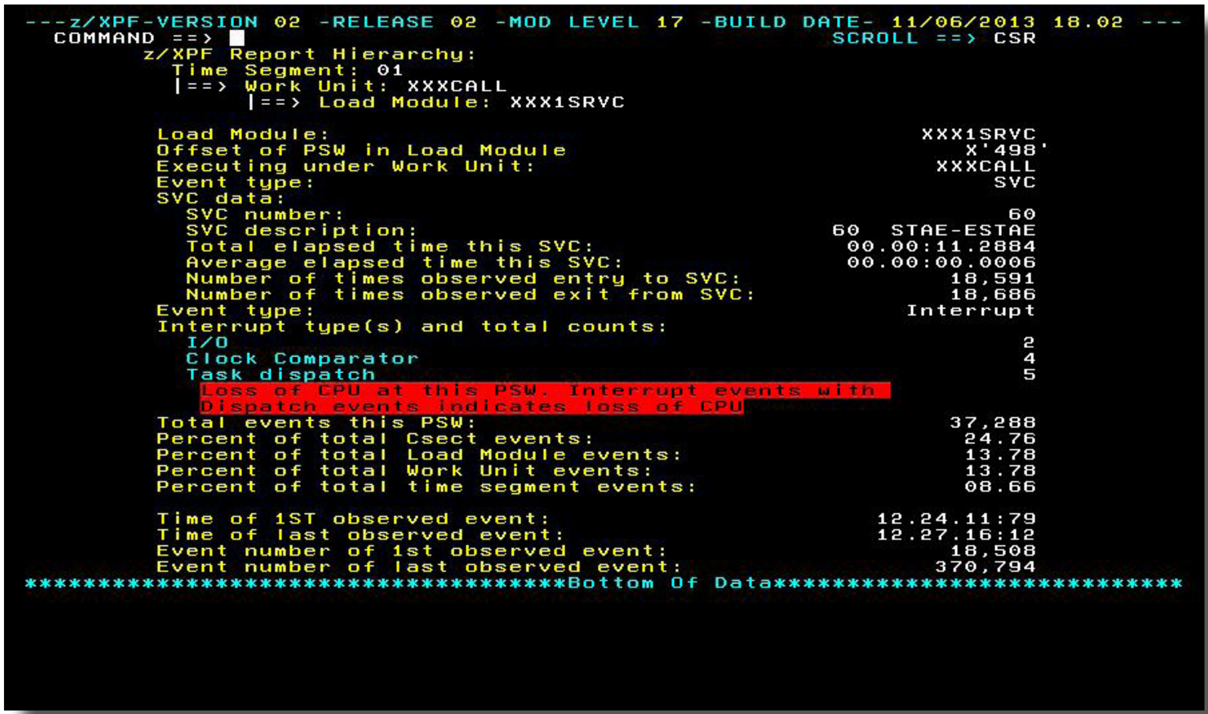
Now z/XPF has dropped me from the Work Unit level to the Load Module Level (just as it does in all the other Summary Reports. Here I see that Load Module XXX1SRVC generated 36,593 Interrupts (now we're tracking ALL interrupts not just Timer Interrupts), or 56.24% of the Work Unit. Let's drill again, and we'll see Figure 9, below:



```
---z/XPF-VERSION 02 -RELEASE 02 -MOD LEVEL 17 -BUILD DATE- 11/06/2013 18.02 ---
  COMMAND ==>                                                 SCROLL ==> CSR
         z/XPF Report Hierarchy:
           Time Segment: 01
           |==> Work Unit: XXXCALL
                |==> Load Module: XXX1SRVC

            Application processor utilization statistics, by Csect PSW

         Csect PSW CPU busy is the total number of interrupts ID'd at
         the PSW offset multipled by the elpased time for one interrupt.

         Hierarchical drill down for:                     CPU Utilization
         Current level of report hierarchy:               Load Module PSW
         Load Module:                                            XXX1SRVC
                                                          (MM.SS.TH:MICS)
         Total CPU elapsed, this Load Module:              00.09:25.5554
         Number of PSW's identified this Load Module:                  41
         Number of PSW's in this report:                               10
         Total interrupt count, this Csect:                        36,593

_ Dril   Load Module PSW offset:      X'498'              (MM.SS.TH:MICS)
         CPU busy, inter'pts ID'd at this PSW:             00.04:70.3783
         Total interrupt count, this PSW:                          18,597
         Percent of Work Unit total interrupt count:                28.58
         ....10...20...30...40...50...60...70...80...90...100

         Actual percent CPU busy, PSW generated:                    01.85


_ Dril   Load Module PSW offset:      X'1DE'              (MM.SS.TH:MICS)
         CPU busy, inter'pts ID'd at this PSW:             00.04:50.7760
         Total interrupt count, this PSW:                          17,822
         Percent of Work Unit total interrupt count:                27.39
         ....10...20...30...40...50...60...70...80...90...100

         Actual percent CPU busy, PSW generated:                    01.77


_ Dril   Load Module PSW offset:     X'572E'              (MM.SS.TH:MICS)
         CPU busy, inter'pts ID'd at this PSW:             00.00:01.9222
         Total interrupt count, this PSW:                              76
         Percent of Work Unit total interrupt count:                00.11
```

Figure 9.

Now we've arrived at the PSW offset within the csect. Normally, we would have seen a csect-oriented display before this one, but in this case the code I'm drilling into doesn't have a module map, so that level of the report is bypassed. No matter, let's drill a final time to see Figure 5-40:



```
---z/XPF-VERSION 02 -RELEASE 02 -MOD LEVEL 17 -BUILD DATE- 11/06/2013 18.02 ---
   COMMAND ==>                                                SCROLL ==> CSR
         z/XPF Report Hierarchy:
           Time Segment: 01
           |==> Work Unit: XXXCALL
                |==> Load Module: XXX1SRVC

          Load Module:                                              XXX1SRVC
          Offset of PSW in Load Module                                X'498'
          Executing under Work Unit:                                 XXXCALL
          Event type:                                                    SVC
          SVC data:
            SVC number:                                                   60
            SVC description:                                   60  STAE-ESTAE
            Total elapsed time this SVC:                        00.00:11.2884
            Average elapsed time this SVC:                      00.00:00.0006
            Number of times observed entry to SVC:                    18,591
            Number of times observed exit from SVC:                   18,686
          Event type:                                              Interrupt
          Interrupt type(s) and total counts:
            I/O                                                            2
            Clock Comparator                                               4
            Task dispatch                                                  5
             Loss of CPU at this PSW. Interrupt events with
             Dispatch events indicates loss of CPU
          Total events this PSW:                                      37,288
          Percent of total Csect events:                               24.76
          Percent of total Load Module events:                         13.78
          Percent of total Work Unit events:                           13.78
          Percent of total time segment events:                        08.66

          Time of 1ST observed event:                            12.24.11:79
          Time of last observed event:                           12.27.16:12
          Event number of 1st observed event:                         18,508
          Event number of last observed event:                       370,794
*************************************Bottom Of Data******************************
```

Figure 10.

We’ve arrived at the final level of the report, and we can see that at this PSW address a call has been to SVC 60, the STAE-ESTAE Supervisor call.

So there you have it. z/XPF now gives you what we think is the world’s ONLY TRUE indication of CPU consumption.

Who is Dave Day?

David Day has spent forty-one years working on and around IBM mainframe computers. He spent four years as an applications programmer, then served for seventeen years as a systems programmer. During the last fourteen years (and counting), Dave Day has been a software development engineer with a concentration in Assembler language programming. Thus, Dave Day has amassed a store of system-level knowledge of z/OS and DB2 internals – a lifetime of architectural knowledge and engineering background.

Dave has worked in a development capacity at Candle Corporation, Programart, Neon Systems, Rocket Software and Tone Software – all highly-regarded development and marketing organizations in the mainframe software industry.

During Dave's work in programming he got the idea which became Event-based profiling. Examining raw Trace records, he saw that the information was "just there", provided at the processor level by z/OS as part of its normal function. Armed with this concept, Dave has spent the last seven years designing and coding z/XPF, the Extended Profile Facility.

These days, Dave spends his time supporting and enhancing z/XPF for a growing audience of users. z/XPF is in an exciting period of growth,

Dave Day can be reached at David.Day@duke-software.com or via telephone at (281) 395-5570.

z/XPF®