z/XPF®

High Definition Profiling

User's Guide

Duke Software www.duke-software.com

For Release V2.R2.M27 and later

revision date 05-27-2015

PREFACE

PROPRIETARY LEGEND

CONTACTING DUKESOFTWARE

Phone: **281-395-5570**
E-Mail: David.Day@duke-software**.com**
Home Page: **http://www.duke-software.com**

Table of Contents

Chapter 1 - Welcome to z/XPF

Welcome to z/XPF, an entirely new way of profiling applications in z/OS. This book is intended to make using z/XPF as easy as possible, but support is always only a phone call away. If we can be of help to you, please don't hesitate to call us.

1-1 What is z/XPF?

z/XPF is the world's first "Event" profiler, an absolutely unique way to measure resource consumption by applications (and systems level) programs running in address spaces.

In order to define the term "Event profiler", it's useful to define the rest of the profilers. All other profilers can be considered "State profilers". By that we mean that the other profilers periodically stop a target address space in order to run control block chains, investigate the state of the programs running there and to gather any other information you may need. Then, they re-start the address space until the next sample interval. Thus, older profiling technologies are based on recording the "State" of an address space.

There are several advantages in using State profiling:

- These products are highly evolved, and have been on the market for decades.
- The technology is therefore robust.
- The market is mature, and many users have spent many years with these products. There is a certain "comfort" factor.

There are also disadvantages to State profiling:

- In between sampling intervals, State profilers are entirely "blind" to what's going on in the target address space. If an action takes place entirely within an interval cycle, a State profiler may not be able to detect it.
- State profilers depend upon stopping an address space, thereby directly affecting that which they are trying to measure.
- State profilers "hook" into the system and the target address space.
- State profilers are *profoundly* expensive.

z/XPF is the world's first "Event" profiler. It works in an entirely different way. Instead of stopping/starting an address space, z/XPF monitors and captures Trace Records from the processor's Trace Table. It captures this data and stores it for reporting.

Whenever an interrupt is processed, a Trace Record is written to the Trace Buffers. That happens for an I/O operation, an SVC, and much more.

There are two kinds of Trace Records: Explicit and Implicit.

- Explicit Trace Records come with a system time-stamp and are generated for interrupts

and I/O sub-channel events.
- Implicit Trace records do not have a time-stamp and are used for Program Call/Return/Transfer. However, z/XPF is able to make a reasonable determination for a time-stamp by comparing Program Calll/Return/Transfer to the Trace Records nearest to them that DO bear a time stamp.

z/XPF "watches" Trace Records and can make a time-stamped log stream out of them. By doing this, it can detect the elapsed time of an Event with either complete precision (in the case of explicit Trace records), or with a high degree of accuracy (in the case of implicit Trace Records). This is unprecedented - a completely new architecture for profiling.

As a result of this approach, z/XPF's Event profiling archtecture has certain advantages:

- Because z/XPF reads Trace Records, it can capture MILLIONS of data points, rather than thousands. This delivers a much higher level of granularity than a State profiler can.
- z/XPF has NO interaction with the target address space. It has no effect on what it measures.
- z/XPF has no "hooks" to the operating system. It "watches" a couple of exit points.
- z/XPF's own footprint is very small. After all, *the data is just there*, and z/XPF merely captures and collates it.
- z/XPF requires no JCL changes in order to operate. You merely point it at an address space.
- z/XPF can measure the elapsed time for SVCs, I/O, memory management events, wait times, contention, page faults, locks, latches and more.
- z/XPF can profile against Task-mode code AND SRB-mode code.
- You may even create your own "Trace records" to keep track of events within your own programs.
- z/XPF offers the ability to set SLIP traps for Instruction Fetch or Branch Trace. That's truly "high-definition" profiling!

The list goes on, but the point is this: *For the first time, a Software Engineer or Capacity Planner can get complete information on the actions of a program in a target address space.* Since Trace Records are generated for every interrupt, you get much more granularity in the data, which creates greater statistical certainty. z/XPF's reporting allows you to drill down through to the PSW level and allows you to view individual Trace Records if you need to go that deeply into your program's performance.

Event profiling is an entirely new architecture. It is truly an elegant solution, in the classic sense.

STOP. Please read this.

Everyone is busy, with a huge task list. We're all in a hurry to get to the next job. However, z/XPF is SO different from other profilers that we urge you NOT TO SKIP STEPS when installing it. For example, z/XPF's abilities to measure DB2 applications will not work if you haven't installed that portion of the product.

The installation itself is straightforward enough. Please follow all the steps.

z/XPF runs at a very low level in the z/OS environment. z/XPF reads Trace Records which are generated by the interaction of every program executing on a z/OS image with the operating system. There can be millions of trace records generated for any application, which means that:

- z/XPF has to execute at high enough priority to examine Trace Records before the Trace Table wraps. The factory default logic will issue an operator command to set the server address space to the SYSSTC WLM workload class. In some environments where there are many address spaces already assigned to this class, this may not be enough priority to insure that z/XPF gets control often enough to keep abreast of the activity occurring in the LPAR. The factory default logic will also set the number of processor trace buffers to 512 from the z/OS default of 256. If/when the ZXPFLOG contains messages indicating a loss of time, or gap, in the copying of trace records, consider setting the number of processor trace buffers higher than 512. This can be accomplished via an operator command, or using a control statement in the z/XPF start-up control dataset.

- z/XPF profiles job steps, not entire jobs.

- If the target application you are profiling is extremely active, and you set z/XPF to run the entire span of the jobstep, then you WILL consume a fair amount of available DASD space for the VSAM capture dataset. It is impossible to predict the number of cylinders needed to contain the complete data capture. It is driven by the activity of the target application. Consider setting aside a separate pool of available space for capture datasets if DASD is at a premium in your installation.

- You have to wait for the data capture to terminate to be able to produce statistics. The environmental data that allows z/XPF to make ‘sense’ of the trace data is written to the end of the dataset at data capture termination.

So, while other profilers are like field glasses, z/XPF is more like an electron microscope: very useful in taking VERY granular measurements at an extremely low level. That’s why the ‘factory default’ setting for any data capture session is 500K records. We think that’s enough data for you to make an informed judgment.

We’re convinced that there’s nothing more useful than z/XPF for getting definitive information on resource consumption at the instruction level, but you have to set up and use the tool properly. *A hammer makes a poor screwdriver.*

We’ll gladly help you, so call us whenever you have a question. OK, let’s move on.

1-2 What Is z/XPF's Anatomy?

z/XPF's anatomy is simple. It consists of:

The Data Capture component:

- A server address space that runs a highly parallelized Global SRB that captures and stores Trace Records.
- Two SMF exits for watching address space initialization and termination. They are IEFUSI and IEFACTRT.

The Reporting component:

- Front-end processing for collating captured Trace Records.
- ISPF-based dynamic ISPF reporting for Summary Data.
- A robust series of "filters" for Detail Reporting, so that you can see individual events.
- z/XPF-PC, a GUI interface that runs on the "Wintel" desktop.

z/XPF's requirements are rather simple as well. z/XPF needs:

- A z/OS operating system at V1R10 and above;
- The ability for z/XPF to issue certain Operator commands;
- APF-authorized storage for z/XPF's load libraries;
- The ability to get control as often as possible in order to read Trace Records;
- A goodly amount of VSAM file storage;
- Virtual memory above the 2GB "bar";
- Security permissions for all of the above.

This book is intended to guide you through creating and terminating data capture sessions, and subsequently manipulating z/XPF's reports to yield the information you need. There's quite a bit of information here, and we're happy to help you if you have questions. Support, and product education for z/XPF is readily available, and we respond in real-time whenever we can.

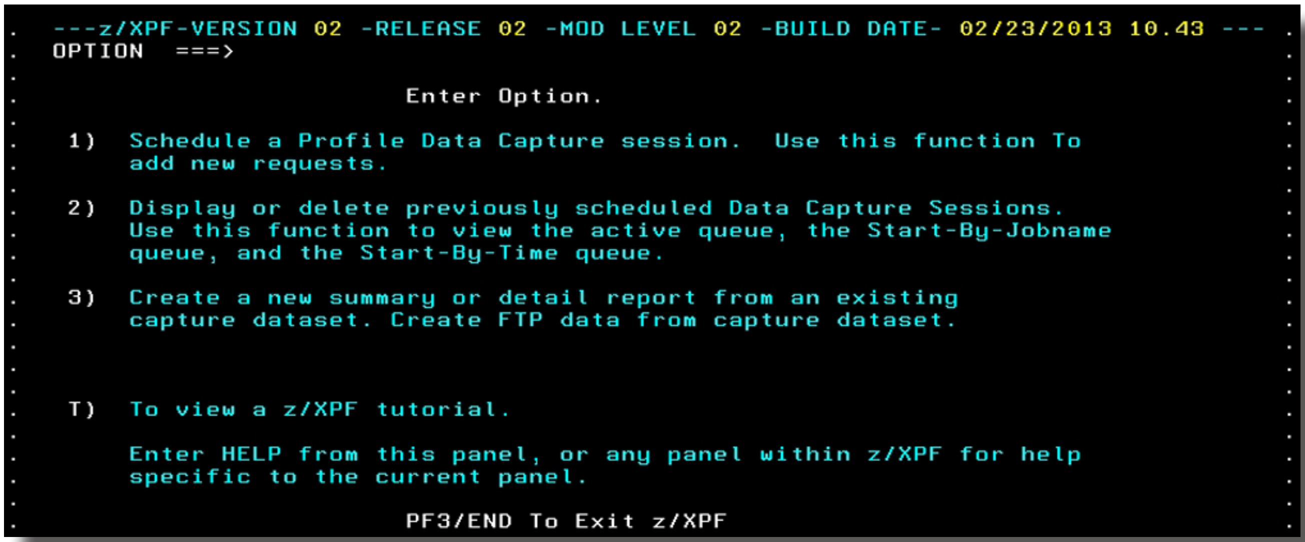
Welcome to z/XPF. We hope that it serves you well.

1-3 Scheduling a New Data Capture

Upon entry to z/XPF's ISPF interface, the user is presented with the panel cleverly entitled, "PRIMARY OPTION PANEL". There are four selections on this panel. The user can:

1. Schedule a new data capture session;
2. Display or delete a previously scheduled data capture session;
3. Create reports from a previous data capture.
4. View a z/XPF Tutorial.

See Figure 1-3-1 below:



```
---z/XPF-VERSION 02 -RELEASE 02 -MOD LEVEL 02 -BUILD DATE- 02/23/2013 10.43 ---
OPTION  ===>

                             Enter Option.

  1)   Schedule a Profile Data Capture session.  Use this function To
       add new requests.

  2)   Display or delete previously scheduled Data Capture Sessions.
       Use this function to view the active queue, the Start-By-Jobname
       queue, and the Start-By-Time queue.

  3)   Create a new summary or detail report from an existing
       capture dataset. Create FTP data from capture dataset.



  T)   To view a z/XPF tutorial.

       Enter HELP from this panel, or any panel within z/XPF for help
       specific to the current panel.

                             PF3/END To Exit z/XPF
```

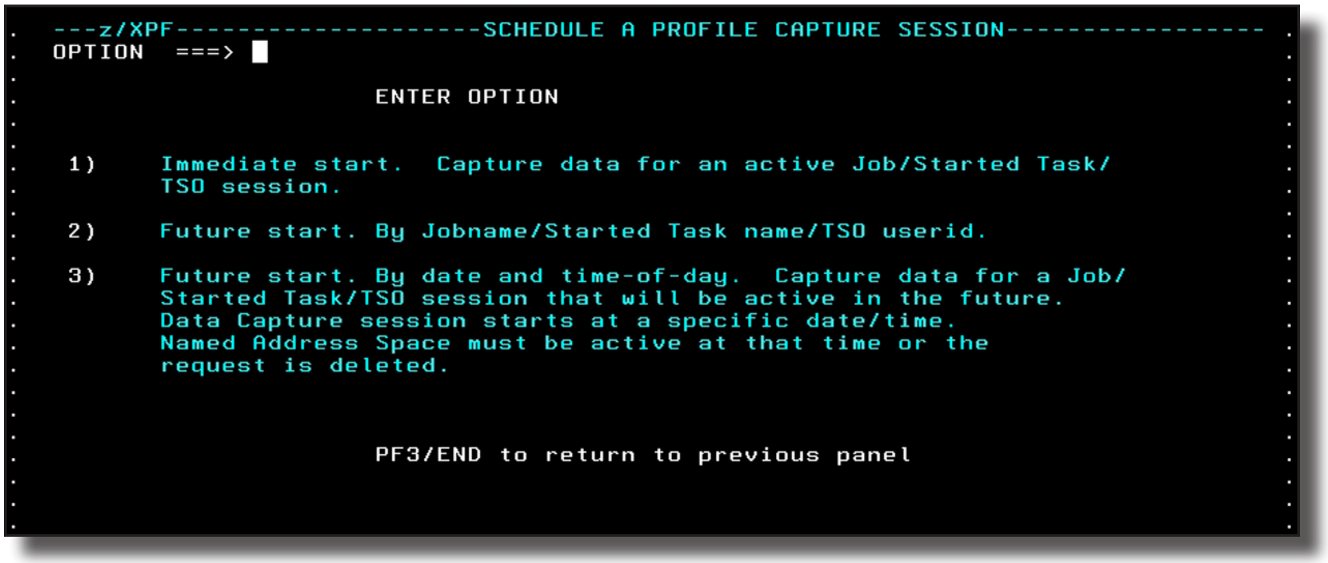
Figure 1-3-1.

Options 1 & 2 require the z/XPF server address space to be active. The user must be logged on to the same system where that server address space is resident.

Option 3 (report generation) does not require communication with the server address space, and may be performed on any z/OS system that has access to the desired data capture dataset.

Option "T" (the Tutorial) may also be used without the z/XPF server running. The z/XPF Tutorial is what we refer to as an "evolving" facility. It's intended to give the user a broad overview of how to use z/XPF. This facility will continue to grow with the product.

When Option 1 is selected from the primary panel, the user is presented with a panel titled, "SCHEDULE A PROFILE CAPTURE SESSION". See Figure 1-3-2 below:



```
---z/XPF-------------------SCHEDULE A PROFILE CAPTURE SESSION-----------------
OPTION  ===> 

                            ENTER OPTION


 1)     Immediate start.  Capture data for an active Job/Started Task/
        TSO session.

 2)     Future start. By Jobname/Started Task name/TSO userid.

 3)     Future start. By date and time-of-day.  Capture data for a Job/
        Started Task/TSO session that will be active in the future.
        Data Capture session starts at a specific date/time.
        Named Address Space must be active at that time or the
        request is deleted.



                            PF3/END to return to previous panel
```

Figure 1-3-2.

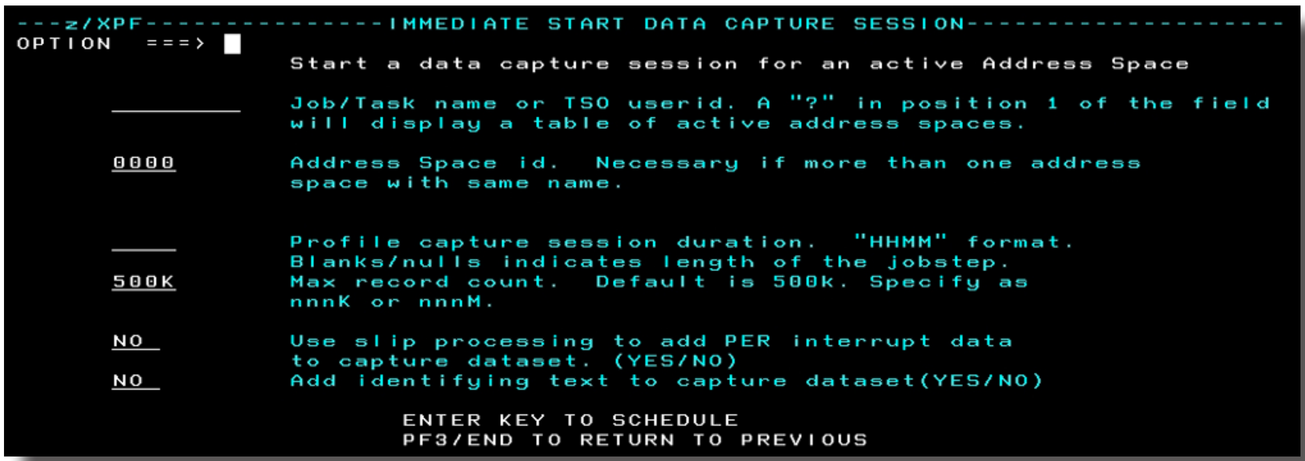
There are three types of data capture requests: Immediate start, future start-by-jobname, and future start-by-time.

1. If you wish to schedule the immediate start of a data capture session then the target application MUST be active at that time.
2. If you schedule a "Future start. By Jobname/Started Task" data capture session, then the data capture session will start the next time that address space becomes active.
3. "Future start. By date and time-of-day" will start a data capture session for a specific address space at a specific time. When this request is in the queue, the address spaces active in the system are searched periodically. The first one located that matches the name in the queue entry satisfies the request.

[z/XPF establishes an SMF exit for exit point IEFUSI during server initialization. This exit is used to communicate address space start information to the z/XPF server. SMF exit IEFACTRT is also used to communicate jobstep termination. The default session time for data capture, once started, is to run to jobstep termination.]

1-4 Immediate Start

When Option 1 is selected from the “SCHEDULE A PROFILE CAPTURE SESSION” panel, the next panel the user sees is entitled, “IMMEDIATE START DATA CAPTURE SESSION”.



```
---z/XPF---------------IMMEDIATE START DATA CAPTURE SESSION-------------------
OPTION   ===>
                          Start a data capture session for an active Address Space

         ________         Job/Task name or TSO userid. A "?" in position 1 of the field
                          will display a table of active address spaces.

         0000             Address Space id.  Necessary if more than one address
                          space with same name.


         _____            Profile capture session duration.   "HHMM" format.
                          Blanks/nulls indicates length of the jobstep.
         500K             Max record count.  Default is 500k. Specify as
                          nnnK or nnnM.

         NO               Use slip processing to add PER interrupt data
                          to capture dataset. (YES/NO)
         NO               Add identifying text to capture dataset(YES/NO)

                                ENTER KEY TO SCHEDULE
                                PF3/END TO RETURN TO PREVIOUS
```

Figure 1-4-1.

Here is an explanation of the parameters on this panel:

Job/Task name or TSO userid: The name of the address space must be entered. If the name is not known, enter a “?”(question mark, without the quotes). z/XPF will display a table of active address spaces and you may select one from the table.

Address Space ID: The second variable from the top accepts the Address Space ID number. Enter this value if the target address space name is not unique among active address spaces on the z/OS image.

Profile Capture Session Duration: Enter “HHMM” (hours-minutes) to tell z/XPF when to shut down the data capture session.

Max Record Count: Enter the maximum number of records you want z/XPF to capture during the session. The default is 500,000 records. To capture ALL Trace Records insert “0M” into this field. *However, be warned that z/XPF may capture far too much data and may overflow available VSAM space.*

Use slip processing to add PER interrupt data to capture dataset: During data capture z/XPF utilizes the interrupt activity generated by all of the active address spaces on the z/OS image to create its profile. The more activity, the better your data capture will be. Consider using “YES” in this field to add SLIP/PER events to the data capture session ONLY when the system is lightly loaded. *HOWEVER, be aware that the use of SLIP can be CPU intensive.*

When you enter “YES” in this field, z/XPF presents another panel that will help you to use SLIP with the product. There is a sub-topic below that discusses the use of SLIP/PER with z/XPF entitled "Using SLIP/PER with z/XPF".

Add Identifying Text to capture datasets (YES/NO): You may add comments to a data capture request to remind yourself later why you initiated the data capture. The comment area is 240 bytes in length. Any valid characters that can be entered on a keyboard can be added as a comment. The comment area is recorded with the data capture dataset, and is viewable in the report generation process.

1-5 Using SLIP/PER with z/XPF

z/OS has a powerful debugging mechanism within the operating system known as SLIP processing. Use of this facility is commonly known as "setting a SLIP" or "setting a SLIP trap". One of the types of SLIPs that can be set is called a PER trap ("Program Event Recording"). With this type of SLIP, the operating system can record events that indicate execution of instructions within a program. z/XPF allows you to use SLIP. Bear in mind that this facility provides extremely finite reporting, BUT ALSO consumes system resources, which is why the use of SLIP may not be permitted at your site.

There are two types of PER traps that may be invoked. One records Instruction Fetch, and the other records Successful Branch instructions. You may use only one of them at a time. The event data can be directed to either of two locations; system trace, or a GTF trace. SLIP/PER data contained in a GTF trace dataset can be formatted by IPCS. SLIP/PER data in the system Trace Table is also formatted by IPCS as part of creating a dump dataset.

Within SLIP, begin and end addresses may be specified for a load module. The user may create PER events for an entire load module, or an extent within the load module. z/XPF submits an Operator command to set the SLIP at the end of the first interval during data capture, and after z/XPF has identified the load module in the target address space. This command uses the RANGE=(start,end) parameter, as opposed to specifying the load module name.

A z/XPF Detail Report that specifies SLIP/PER records will list the records in time sequence. The user will see the instruction address within the application load module, and can see the sequence of instructions executed.

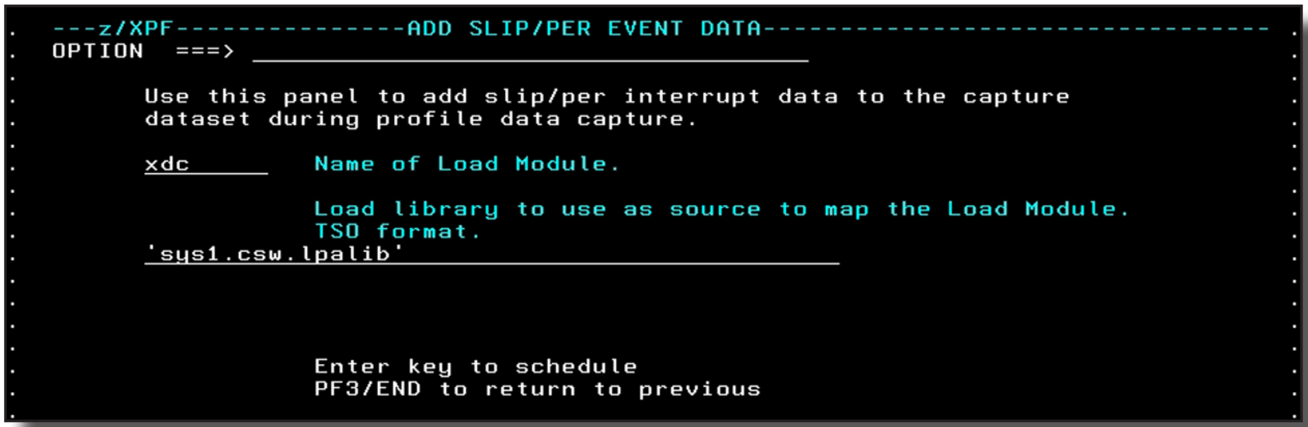
The SLIP command is considered "fire and forget" by z/XPF in that no check is made to determine if the command was successful. Bear in mind that only one PER SLIP may be active at a time. If another SLIP is already active when z/XPF tries to start one for a z/XPF user, z/XPF's will be ignored.

If the command is accepted, z/OS SLIP will start monitoring the virtual storage range in the target address space, and will create events when the execution of application code falls within the range specified on the SLIP command. <u>You should keep in mind that the larger the virtual storage area SLIP has to monitor, the higher the overhead.</u> When the data capture session terminates, another Operator command is executed to delete the SLIP.

Do NOT run a SLIP on your system without consulting with your installation systems programmer(s). Consult the MVS Operator Commands Reference for further information on how to use the SLIP command.

z/XPF's use of SLIP/PER recording is controlled by the z/XPF Control Statement, "SLIP_COMMANDS=YES/NO".

When you DO specify "YES" in the "Use slip processing" field (from Figure 1-3-1), you will be presented with the panel below in Figure 1-5-1 below:

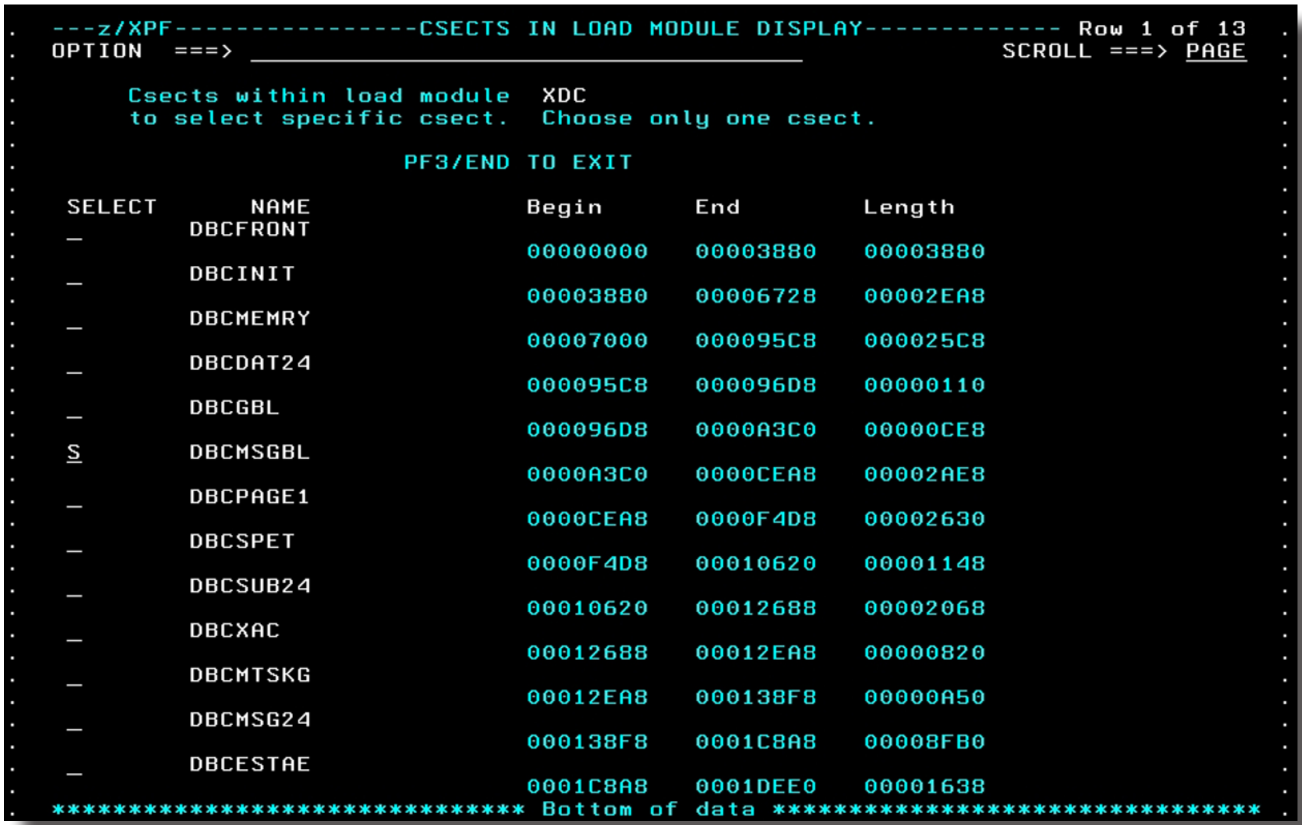


```
---z/XPF---------------ADD SLIP/PER EVENT DATA--------------------------------
OPTION  ===> ____________________________________

       Use this panel to add slip/per interrupt data to the capture
       dataset during profile data capture.

       xdc_____    Name of Load Module.

                   Load library to use as source to map the Load Module.
                   TSO format.
       'sys1.csw.lpalib'_____________________________



                   Enter key to schedule
                   PF3/END to return to previous
```

Figure 1-5-1.

The name of the load module must be entered in the top variable. The name of the load library where the module is located is entered in the 2nd variable. z/XPF treats the value to be used for the load library as a TSO name. In other words, it will add the TSO userid to the value entered and then try to locate that library. Enclose the value in single quote marks to have z/XPF use the value verbatim. In the example below, load module XDC located in dataset 'SYS1.CSW.LPALIB' will be used.

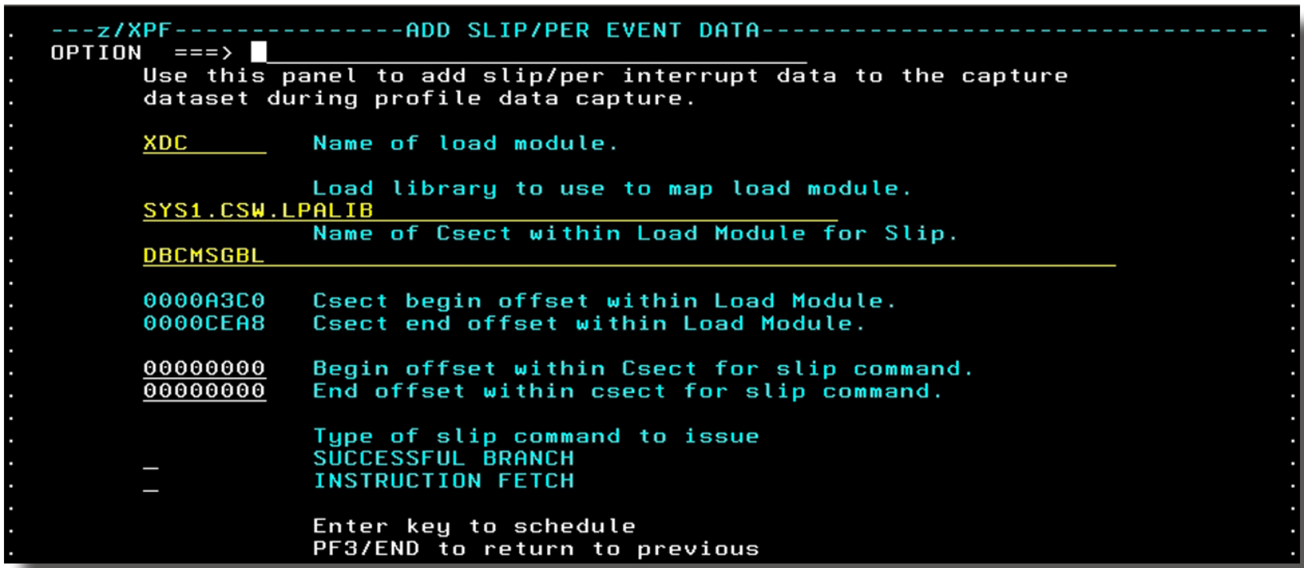
When the information is correct, press the Enter key. z/XPF will then use dynamic allocation to allocate the dataset, and invoke the Binder API to map the load module. If both of these functions are successful, the next screen you see will contain a table with all the csects and their positions within the specified load module. Refer to Figure 1-5-2 below:



```
---z/XPF---------------CSECTS IN LOAD MODULE DISPLAY------------- Row 1 of 13
OPTION  ===> ______________________________________        SCROLL ===> PAGE

          Csects within load module  XDC
          to select specific csect.  Choose only one csect.

                             PF3/END TO EXIT

  SELECT       NAME                    Begin       End         Length
                DBCFRONT
    _                                  00000000    00003880    00003880
    _           DBCINIT
                                       00003880    00006728    00002EA8
    _           DBCMEMRY
                                       00007000    000095C8    000025C8
    _           DBCDAT24
                                       000095C8    000096D8    00000110
    _           DBCGBL
                                       000096D8    0000A3C0    00000CE8
    S           DBCMSGBL
                                       0000A3C0    0000CEA8    00002AE8
    _           DBCPAGE1
                                       0000CEA8    0000F4D8    00002630
    _           DBCSPET
                                       0000F4D8    00010620    00001148
    _           DBCSUB24
                                       00010620    00012688    00002068
    _           DBCXAC
                                       00012688    00012EA8    00000820
    _           DBCMTSKG
                                       00012EA8    000138F8    00000A50
    _           DBCMSG24
                                       000138F8    0001C8A8    00008FB0
    _           DBCESTAE
                                       0001C8A8    0001DEE0    00001638
******************************* Bottom of data ********************************
```

Figure 1-5-2.

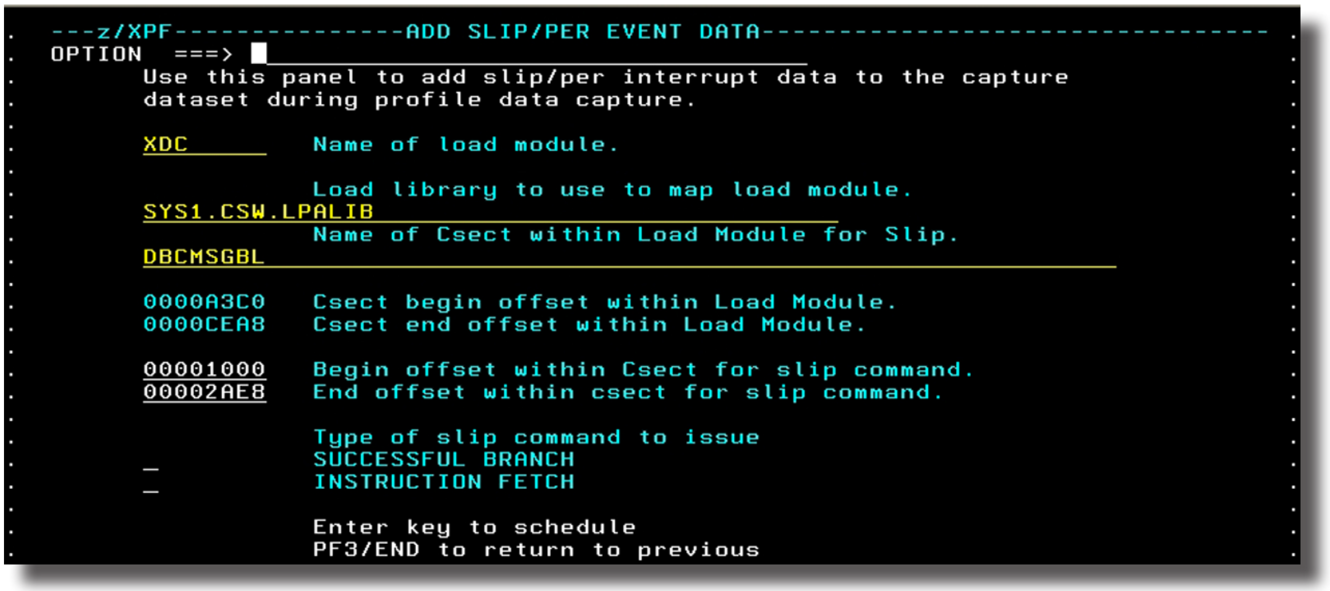
Place any non-blank character in the select column to the left of the csect name to select that csect, then depress the enter key. You will be returned to the previous panel, but with the selected csect information displayed. In this example, csect DBCMSGBL was selected from the map table. See Figure 1-5-3 below:



```
---z/XPF---------------ADD SLIP/PER EVENT DATA---------------------------------
OPTION  ===> _
            Use this panel to add slip/per interrupt data to the capture
            dataset during profile data capture.

            XDC         Name of load module.

                        Load library to use to map load module.
            SYS1.CSW.LPALIB
                        Name of Csect within Load Module for Slip.
            DBCMSGBL

            0000A3C0    Csect begin offset within Load Module.
            0000CEA8    Csect end offset within Load Module.

            00000000    Begin offset within Csect for slip command.
            00000000    End offset within csect for slip command.

                        Type of slip command to issue
            _           SUCCESSFUL BRANCH
            _           INSTRUCTION FETCH

                        Enter key to schedule
                        PF3/END to return to previous
```

Figure 1-5-3.

At this point, there is enough information known about the load module and csect to construct a SLIP SET command. However, if you are interested in tracing only a part of the csect, the Begin and End offset values can be altered. Figure 1-8 contains the same panel with the offsets changed. When changing offsets, they must be specified as 8-byte values, with leading zeroes. See Figure 1-5-4 below:



```
.  ---z/XPF---------------ADD SLIP/PER EVENT DATA--------------------------------- .
.  OPTION  ===> _                                                                   .
.        Use this panel to add slip/per interrupt data to the capture              .
.        dataset during profile data capture.                                      .
.                                                                                   .
.        XDC         Name of load module.                                          .
.                                                                                   .
.                    Load library to use to map load module.                       .
.        SYS1.CSW.LPALIB                                                            .
.                    Name of Csect within Load Module for Slip.                    .
.        DBCMSGBL                                                                   .
.                                                                                   .
.        0000A3C0    Csect begin offset within Load Module.                        .
.        0000CEA8    Csect end offset within Load Module.                          .
.                                                                                   .
.        00001000    Begin offset within Csect for slip command.                   .
.        00002AE8    End offset within csect for slip command.                     .
.                                                                                   .
.                    Type of slip command to issue                                 .
.        _           SUCCESSFUL BRANCH                                              .
.        _           INSTRUCTION FETCH                                              .
.                                                                                   .
.                    Enter key to schedule                                          .
.                    PF3/END to return to previous                                  .
```

Figure 1-5-4.

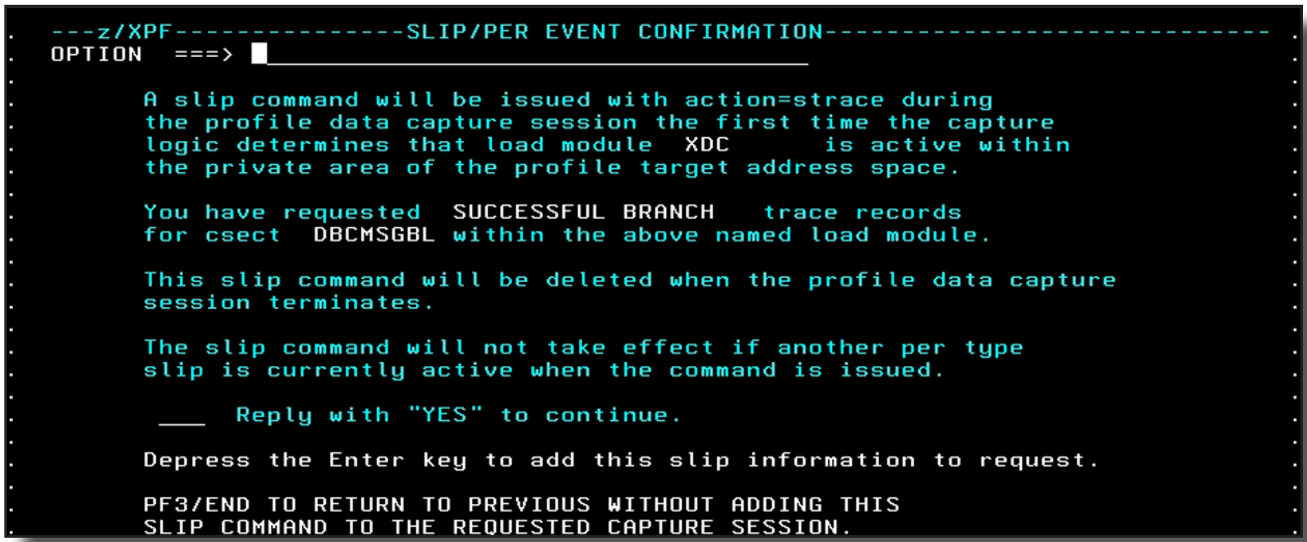
The last variable to fill in on this panel is the type of PER trap you want. Use “S” to select one of them.

- **SUCCESSFUL BRANCH** will create an event for each branch instruction that is taken within the offset, within the csect, and within the load module.
- **INSTRUCTION FETCH** will create an event for each instruction as it is fetched to be executed.
- *You may pick one parameter or the other, but not both.*

[The MVS Operator Commands Reference states there isn’t much difference in overhead between using SLIP against SUCCESSFUL BRANCH and INSTRUCTION FETCH. Both of these event types will add size to the data capture dataset. INSTRUCTION FETCH will increase the size at a greater rate than SUCCESSFUL BRANCH.]

When the type of SLIP has been selected, press Enter.

The next panel to be displayed (in Figure 1-8) is a verification panel. The verification panel gives the user the ability to cancel SLIP, change the settings or continue. Figure 1-5-5 below shows the verification panel.



```
---z/XPF---------------SLIP/PER EVENT CONFIRMATION-----------------------------
OPTION  ===> _

            A slip command will be issued with action=strace during
            the profile data capture session the first time the capture
            logic determines that load module  XDC      is active within
            the private area of the profile target address space.

            You have requested  SUCCESSFUL BRANCH   trace records
            for csect  DBCMSGBL within the above named load module.

            This slip command will be deleted when the profile data capture
            session terminates.

            The slip command will not take effect if another per type
            slip is currently active when the command is issued.

             ___  Reply with "YES" to continue.

            Depress the Enter key to add this slip information to request.

            PF3/END TO RETURN TO PREVIOUS WITHOUT ADDING THIS
            SLIP COMMAND TO THE REQUESTED CAPTURE SESSION.
```

Figure 1-5-5.

If the information is correct, enter YES into the field, and press the Enter key. *If, when the Enter key is pressed, any value other than YES is in the field, then the SLIP processing will not be done.* PF3/END can also be used to return to the previous panel.

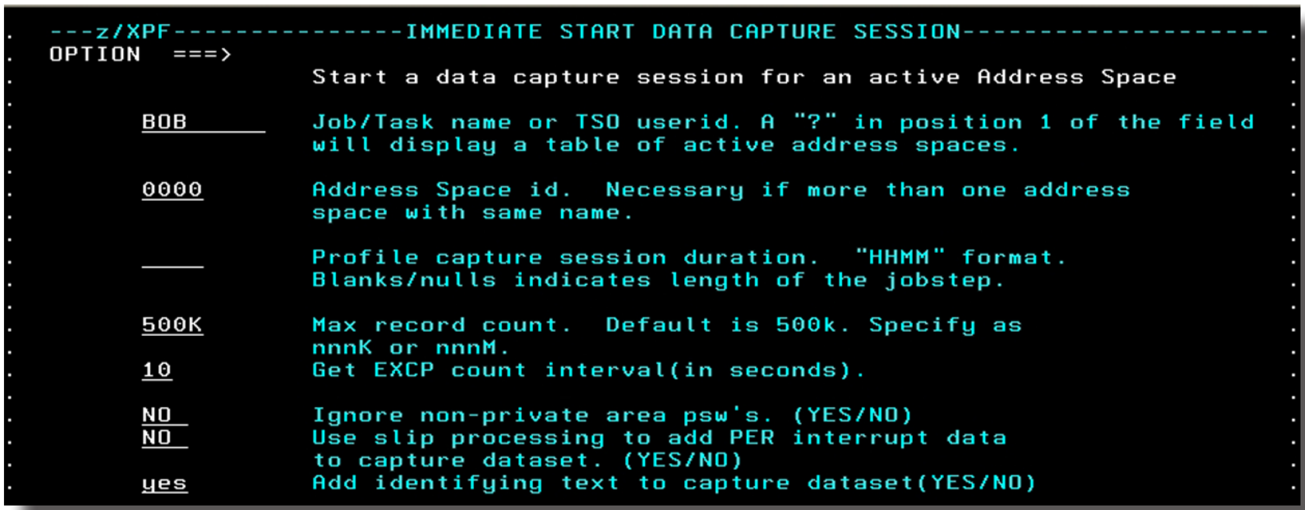
z/XPF's interaction with SLIP processing takes a very powerful debugging tool previously available only/mostly to Systems Programmers, and makes it available to anyone with access to z/XPF. Use of SLIP/PER can be costly in system resources, so make sure that this is what you really intend to do.

z/XPF's SLIP/PER facility is also discussed in the z/XPF Installation Guide.

1-6 Adding Text/Comments To A Data Capture Request

It is a very good idea to get into the habit of adding comments to a capture request. The capture dataset name will have a date, time, and address space name as part of the dataset name, and that alone may be sufficient for your purposes. However, at some point in the tuning process, it may be necessary for someone other than the original requestor to have access to report data contained in the dataset. That's when the use of comments may reward you and your team.

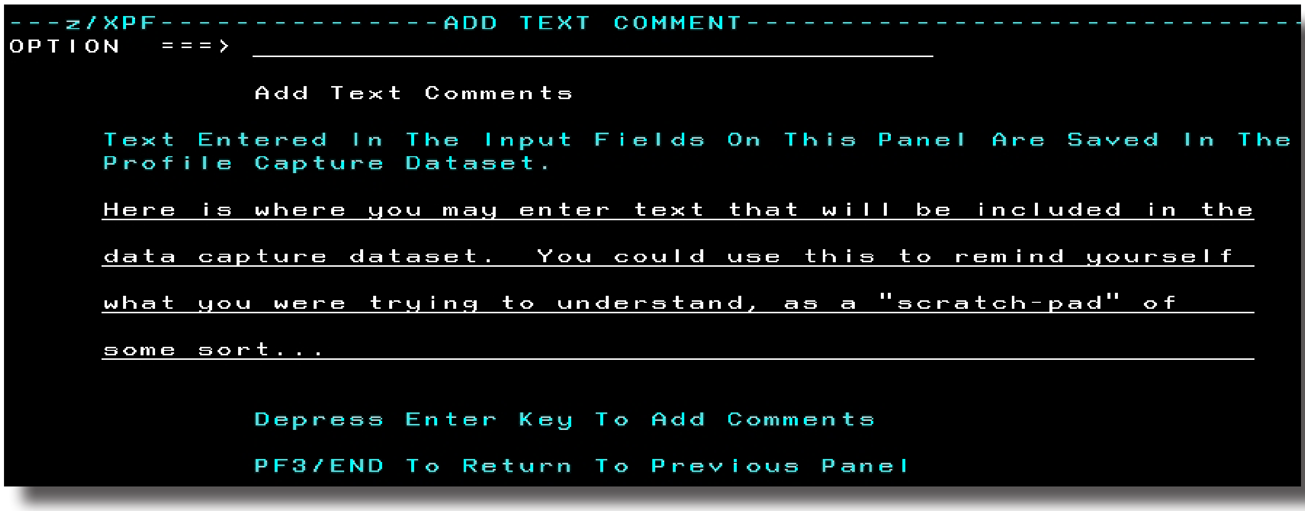
Figure 1-6-1 below shows the panel with the YES variable filled in:



```
---z/XPF---------------IMMEDIATE START DATA CAPTURE SESSION--------------------
OPTION  ===>
                          Start a data capture session for an active Address Space

        BOB_____          Job/Task name or TSO userid. A "?" in position 1 of the field
                          will display a table of active address spaces.

        0000              Address Space id.  Necessary if more than one address
                          space with same name.

        ____              Profile capture session duration.  "HHMM" format.
                          Blanks/nulls indicates length of the jobstep.

        500K              Max record count.  Default is 500k. Specify as
                          nnnK or nnnM.
        10                Get EXCP count interval(in seconds).

        NO                Ignore non-private area psw's. (YES/NO)
        NO                Use slip processing to add PER interrupt data
                          to capture dataset. (YES/NO)
        yes               Add identifying text to capture dataset(YES/NO)
```

Figure 1-6-1.

If one chooses to add commentary to a data capture dataset (as in Figure 1-10), then z/XPF displays the panel below, in Figure 1-6-3:

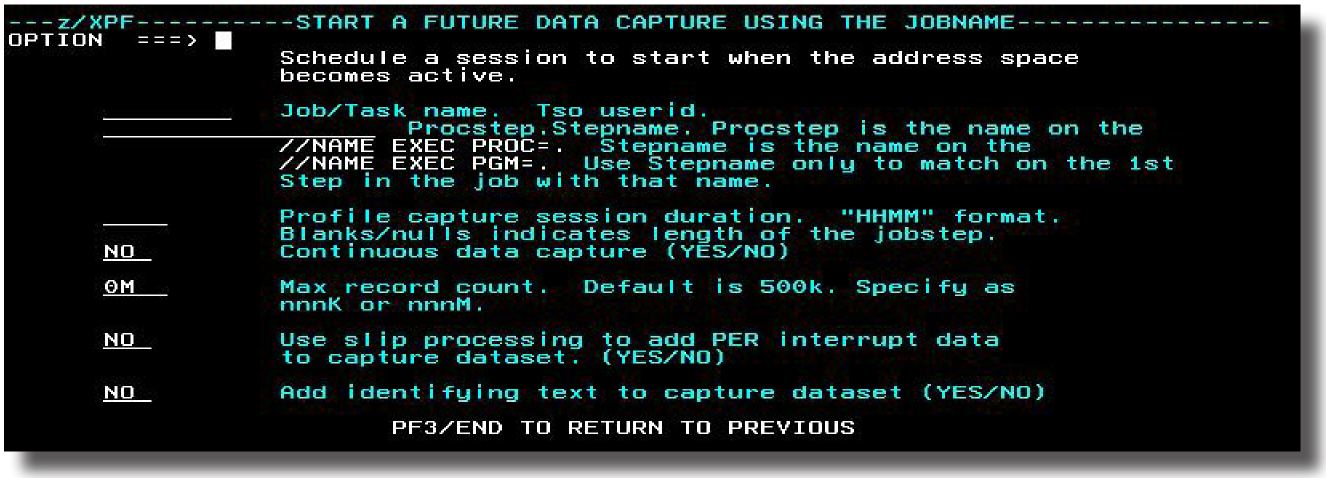


```
---z/XPF---------------ADD TEXT COMMENT------------------------------
OPTION   ===> ____________________________________________

                Add Text Comments

        Text Entered In The Input Fields On This Panel Are Saved In The
        Profile Capture Dataset.

        Here is where you may enter text that will be included in the

        data capture dataset.  You could use this to remind yourself

        what you were trying to understand, as a "scratch-pad" of

        some sort...


                Depress Enter Key To Add Comments

                PF3/END To Return To Previous Panel
```

Figure 1-6-3.

1-7 Future Start-By-Jobname

When you specify Option 2 on the panel entitled, "SCHEDULE A PROFILE CAPTURE SESSION" (Figure 20), you will see the panel displayed as Figure 1-7-1 below:



```
---z/XPF----------START A FUTURE DATA CAPTURE USING THE JOBNAME----------------
OPTION   ===> █
                  Schedule a session to start when the address space
                  becomes active.

      ________    Job/Task name.   Tso userid.
      __________________  Procstep.Stepname. Procstep is the name on the
                  //NAME EXEC PROC=.  Stepname is the name on the
                  //NAME EXEC PGM=.  Use Stepname only to match on the 1st
                  Step in the job with that name.

      ____        Profile capture session duration.  "HHMM" format.
                  Blanks/nulls indicates length of the jobstep.
      NO          Continuous data capture (YES/NO)

      0M          Max record count.  Default is 500k. Specify as
                  nnnK or nnnM.

      NO          Use slip processing to add PER interrupt data
                  to capture dataset. (YES/NO)

      NO          Add identifying text to capture dataset (YES/NO)

                         PF3/END TO RETURN TO PREVIOUS
```

Figure 1-7-1.

Use this panel to create a data capture request for an address space that will become active at some point in the future. If the JCL contains multiple steps, and the step to be captured is not the first step, enter the jobstep name as well. Entries must conform to standard Job/Task and jobstep naming criteria.

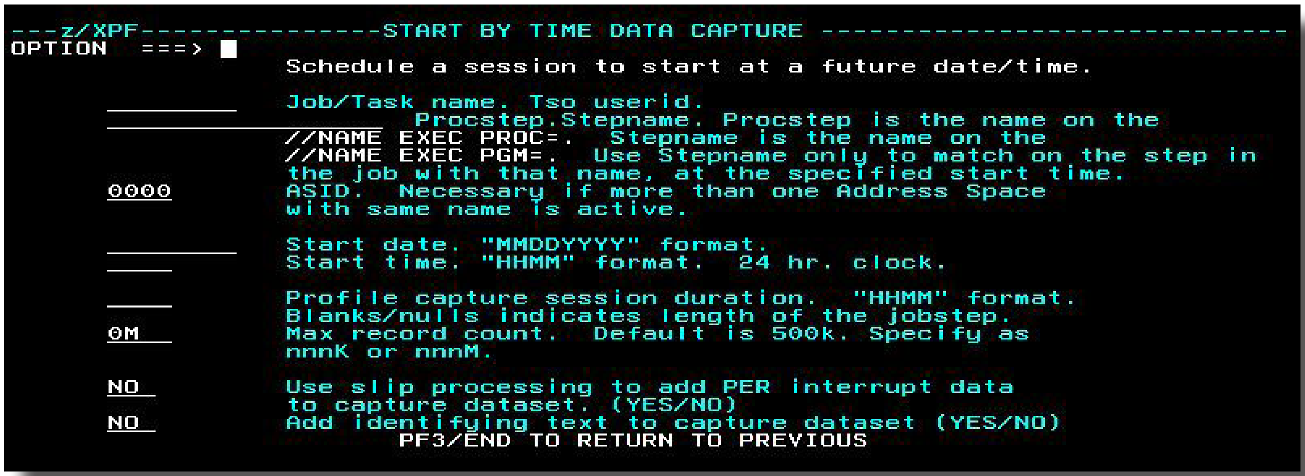
Entering the parameters for Future Start-by-Jobname is identical to the procedure in "Immediate Start".

[z/XPF utilizes z/OS SMF exit IEFUSI to notify z/XPF's server address space of job and step initiation. If a start-by-jobname request is in the queue but does not move from that queue to active when the named address space starts, then the IEFUSI exit is not getting control. This can occur when the z/OS parmlib definitions used to define SMF exit processing do not specify an IEFUSI exit point for the type of address space named in the start-by-jobname request. If this happens, then the ZXPFLOG dataset will contain messages from z/XPF initialization that describe the exit points where the IEFUSI exit was dynamically installed using the dynamic exits facility. Consult with your installation's System Programming staff to rectify this situation.]

1-8 Future Start by Date and Time-of-Day

Option 3 from the SCHEDULE A PROFILE CAPTURE SESSION panel presents you with a panel entitled, “START BY TIME DATA CAPTURE. This panel is used to place a request into the start-by-time queue. Entries are placed in this queue in start time order, the entry with the earliest start time at the head of the queue.

At the designated time, the entry at the top of the queue is removed from the queue, and the z/OS system is searched for an active address space that matches the request. If found, data capture is initiated for that address space. If not found, a message is logged in the ZXPFLOG dataset, and the entry is discarded. Figure 1-8-1 below shows the start-by-time panel:



```
---z/XPF----------------START BY TIME DATA CAPTURE -----------------------------
OPTION   ===> █
                 Schedule a session to start at a future date/time.

      ________   Job/Task name. Tso userid.
      _________________ Procstep.Stepname. Procstep is the name on the
                 //NAME EXEC PROC=.  Stepname is the name on the
                 //NAME EXEC PGM=.  Use Stepname only to match on the step in
                 the job with that name, at the specified start time.
      0000       ASID.  Necessary if more than one Address Space
                 with same name is active.

      ________   Start date. "MMDDYYYY" format.
      ____       Start time. "HHMM" format.  24 hr. clock.

      ____       Profile capture session duration.  "HHMM" format.
                 Blanks/nulls indicates length of the jobstep.
      0M__       Max record count.  Default is 500k. Specify as
                 nnnK or nnnM.

      NO_        Use slip processing to add PER interrupt data
                 to capture dataset. (YES/NO)
      NO_        Add identifying text to capture dataset (YES/NO)
                        PF3/END TO RETURN TO PREVIOUS
```

Figure 1-8-1.

You can specify Job/Task name, Procstep.Stepname, the Address Space identifier, date/time values and other variables from the other start menus. After you have reviewed your choices here, depress the Enter key.

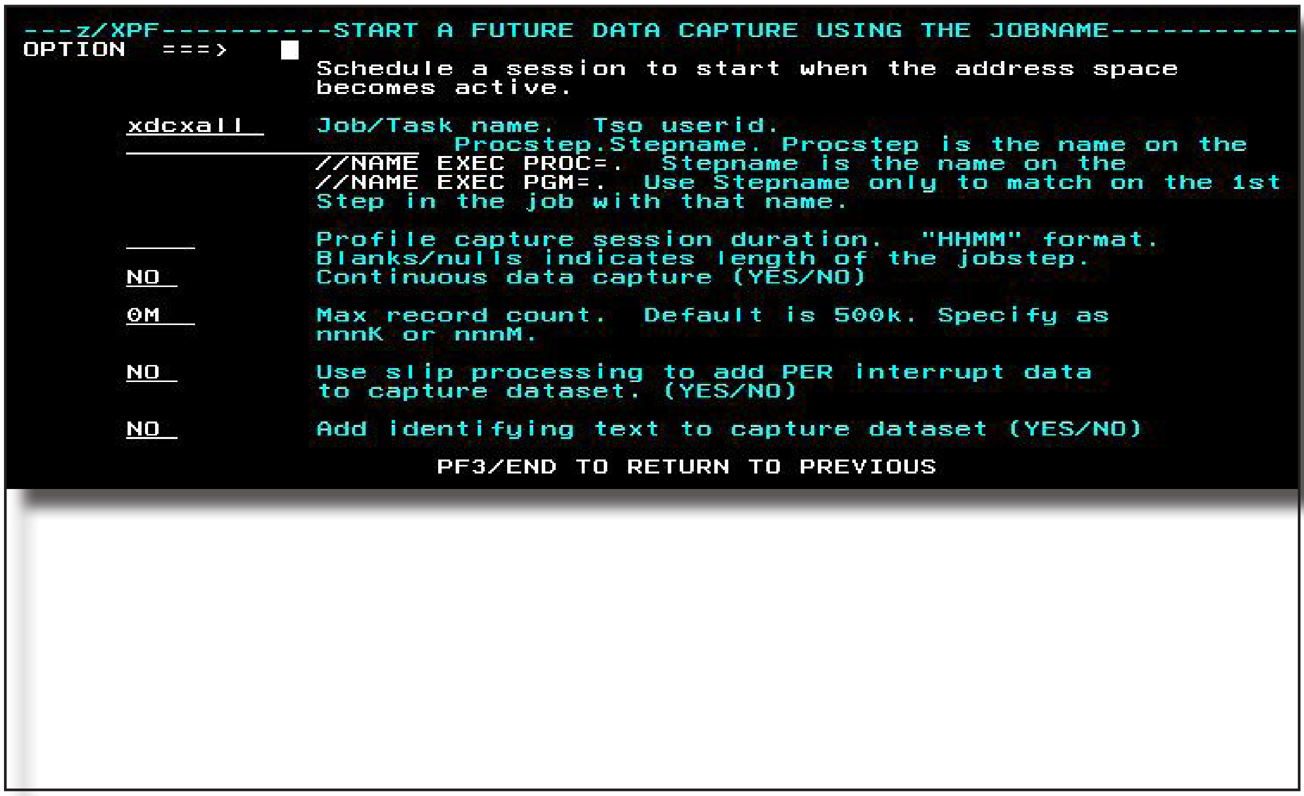
1-9 About ZXPFLOG

z/XPF keeps track of its actions in the ZXPFLOG dataset. This dataset is allocated as a JES SYSOUT dataset, and the DD statement used is "ZXPFLOG". It can be found in z/XPF's started task JCL.

The ZXPFLOG is a very good resource, for everything that z/XPF encounters normally ends up as a line (or more) in the log. You can use the ZXPFLOG log to verify what's going on, and it can be a valuable resource for users AND for us, your support people.

If you experience unusual results or problems, then please retain the ZXPFLOG dataset from you z/XPF session so that we can use it for problem determination.

An example of the ZXPFLOG appears below, in Figure 1-9-1:



```
---z/XPF----------START A FUTURE DATA CAPTURE USING THE JOBNAME-----------
OPTION  ===>
                      Schedule a session to start when the address space
                      becomes active.

      xdcxall         Job/Task name.  Tso userid.
                              Procstep.Stepname. Procstep is the name on the
                      //NAME EXEC PROC=.  Stepname is the name on the
                      //NAME EXEC PGM=.  Use Stepname only to match on the 1st
                      Step in the job with that name.

                      Profile capture session duration.  "HHMM" format.
                      Blanks/nulls indicates length of the jobstep.
      NO              Continuous data capture (YES/NO)

      0M              Max record count.  Default is 500k. Specify as
                      nnnK or nnnM.

      NO              Use slip processing to add PER interrupt data
                      to capture dataset. (YES/NO)

      NO              Add identifying text to capture dataset (YES/NO)
                             PF3/END TO RETURN TO PREVIOUS
```

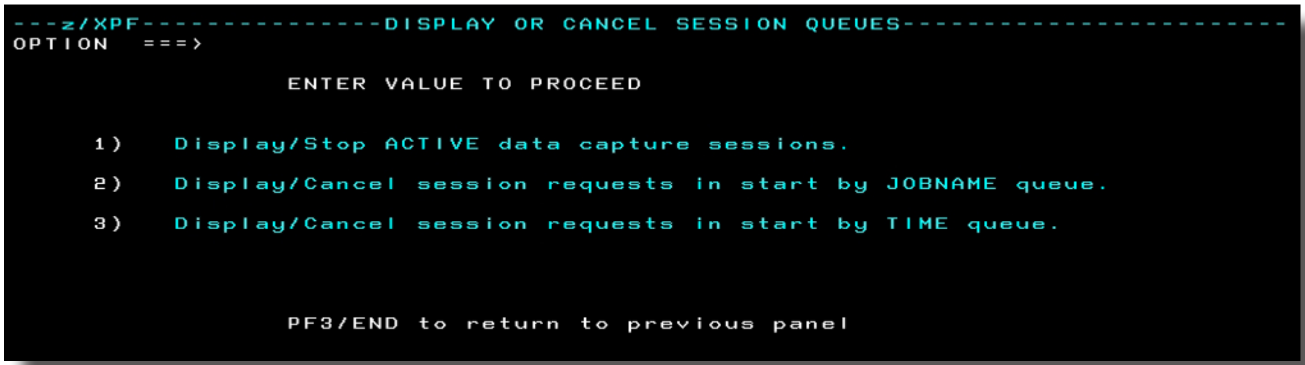
Figure 1-9-1.

Chapter 2 - Working With z/XPF Request Queues

Option 2 from the Primary Option Panel provides a mechanism for the user to manipulate the z/XPF request queues. Entries in the start-by-job and start-by-time queues can be viewed or deleted. An active data capture session may also be terminated.

2-1 Stop an Active Data Capture

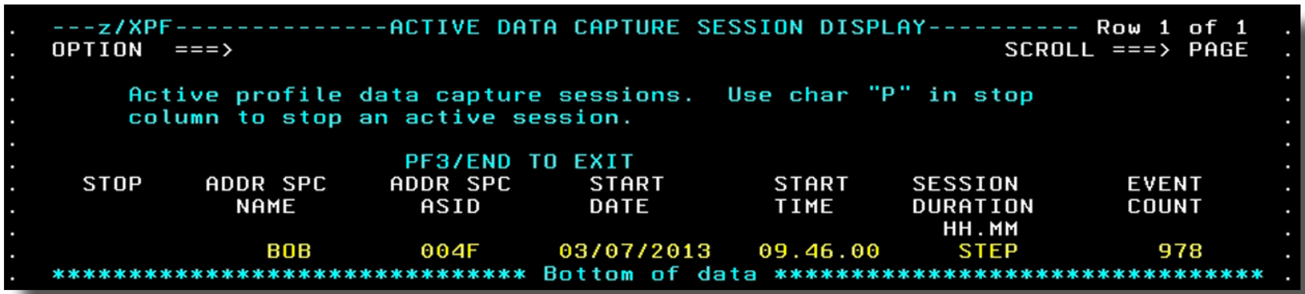
When Option 2 from the primary panel is selected, the user is presented with a panel entitled, “DISPLAY OR CANCEL SESSION QUEUES”. Refer to Figure 2-1-1 below.



```
---z/XPF---------------DISPLAY OR CANCEL SESSION QUEUES-----------------------
OPTION   ===>

                  ENTER VALUE TO PROCEED


      1)    Display/Stop ACTIVE data capture sessions.

      2)    Display/Cancel session requests in start by JOBNAME queue.

      3)    Display/Cancel session requests in start by TIME queue.



                  PF3/END to return to previous panel
```

Figure 2-1-1.

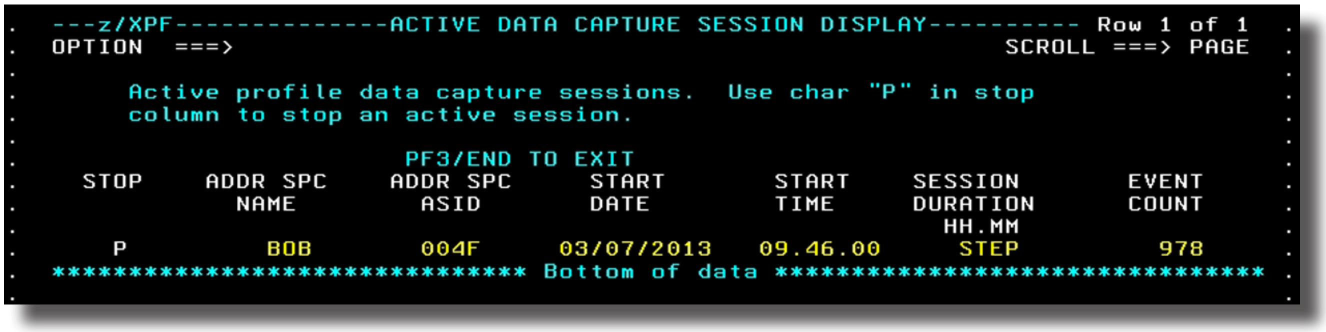
The next three Figures depict the sequence for stopping/terminating an active data capture session. When Option 1 is selected from the panel in Figure 2-1-1. you will see a panel similar to the one below, in Figure 2-1-2. In this example, only one data capture session is active.



```
.  ---z/XPF-------------ACTIVE DATA CAPTURE SESSION DISPLAY---------- Row 1 of 1  .
.  OPTION  ===>                                                  SCROLL ===> PAGE  .
.                                                                                  .
.        Active profile data capture sessions.  Use char "P" in stop               .
.        column to stop an active session.                                         .
.                                                                                  .
.                          PF3/END TO EXIT                                         .
.    STOP      ADDR SPC    ADDR SPC     START        START     SESSION       EVENT  .
.                NAME        ASID       DATE         TIME      DURATION      COUNT  .
.                                                                HH.MM              .
.                  BOB        004F      03/07/2013   09.46.00     STEP          978  .
.  ******************************* Bottom of data ********************************* .
```

Figure 2-1-2.

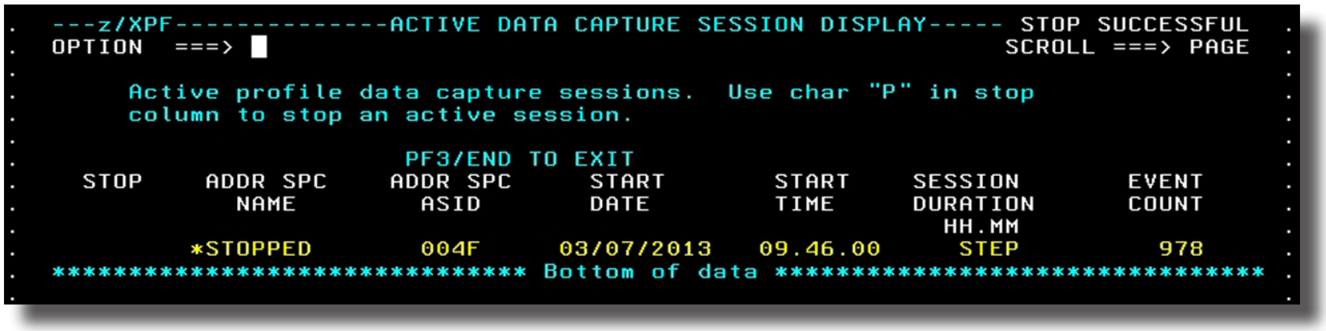
Figure 2-1-3 shows the panel with the character "P " in the STOP column, prior to the Enter key being depressed.



```
---z/XPF--------------ACTIVE DATA CAPTURE SESSION DISPLAY---------- Row 1 of 1
OPTION  ===>                                                  SCROLL ===> PAGE

     Active profile data capture sessions.  Use char "P" in stop
     column to stop an active session.

                                  PF3/END TO EXIT
  STOP     ADDR SPC    ADDR SPC      START        START    SESSION       EVENT
              NAME        ASID        DATE         TIME     DURATION      COUNT
                                                              HH.MM
   P          BOB         004F     03/07/2013   09.46.00     STEP           978
******************************* Bottom of data ********************************
```

Figure 2-1-3.

Figure 2-1-4 shows the panel after the active data capture session has been terminated. You can see the "STOP SUCCESSFUL" message in the upper right portion of the screen as well as "*STOPPED" where the address space name was previously displayed

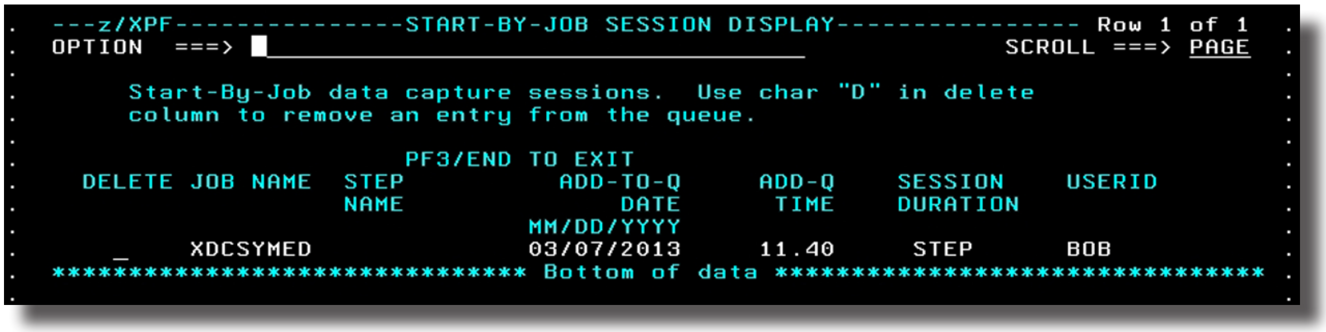


```
---z/XPF--------------ACTIVE DATA CAPTURE SESSION DISPLAY----- STOP SUCCESSFUL
OPTION  ===>                                                  SCROLL ===> PAGE

     Active profile data capture sessions.  Use char "P" in stop
     column to stop an active session.

                                  PF3/END TO EXIT
  STOP     ADDR SPC    ADDR SPC      START        START    SESSION       EVENT
              NAME        ASID        DATE         TIME     DURATION      COUNT
                                                              HH.MM
          *STOPPED        004F     03/07/2013   09.46.00     STEP           978
******************************* Bottom of data ********************************
```

Figure 2-1-4.

2-2 Display Requests In Start-By-Jobname Queue

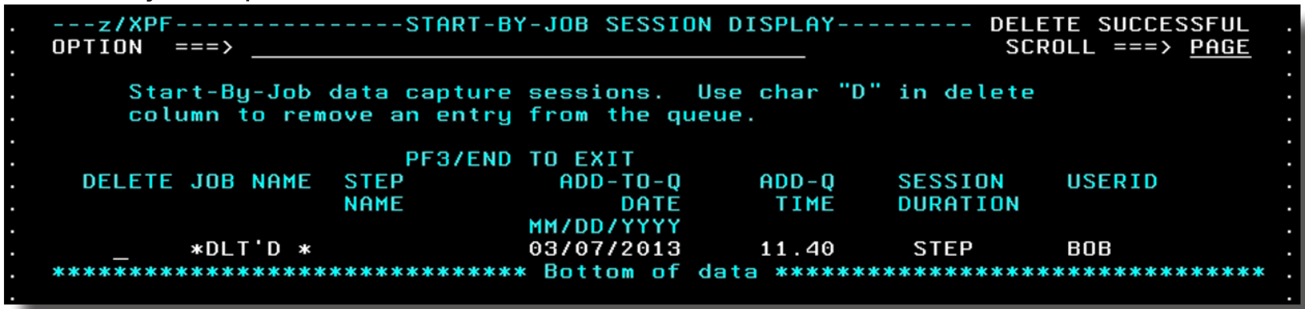
When Option 2 is selected from the panel entitled, "DISPLAY OR CANCEL SESSION QUEUES" (see Figure 2-1-1), and when you select Option 2 AND when there are one or more requests in the start-by-jobname queue, the panel entitled, "START-BY-JOB SESSION DISPLAY" is displayed. Refer to figure 2-2-1 below:



```
---z/XPF---------------START-BY-JOB SESSION DISPLAY---------------- Row 1 of 1
OPTION  ===>                                                  SCROLL ===> PAGE

     Start-By-Job data capture sessions.  Use char "D" in delete
     column to remove an entry from the queue.

                                  PF3/END TO EXIT
  DELETE JOB NAME  STEP            ADD-TO-Q     ADD-Q    SESSION    USERID
                   NAME                DATE      TIME    DURATION
                                   MM/DD/YYYY
     _    XDCSYMED                 03/07/2013    11.40     STEP      BOB
******************************* Bottom of data ********************************
```

Figure 2-2-1.

Enter a “D” next to any job to stop the data capture. You can see the “DELETE SUCCESSFUL” message in the upper right portion of the screen as well as the “*DLT’D” message next to the cancelled job-step.

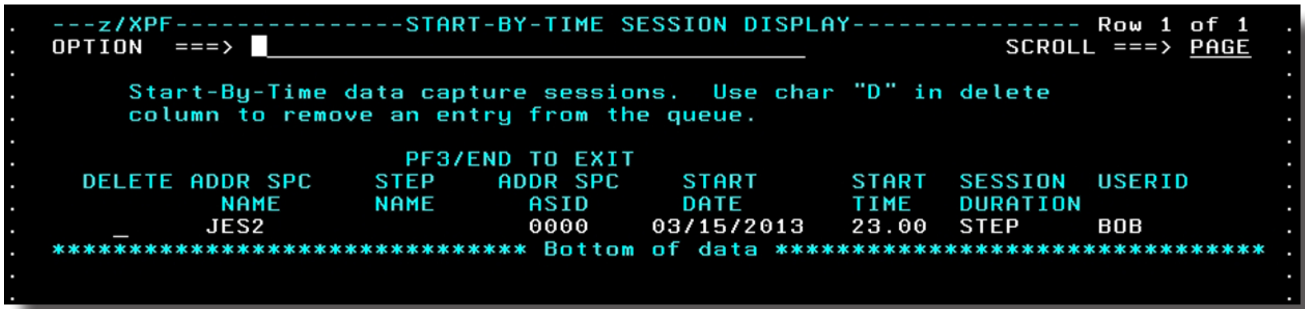


```
---z/XPF---------------START-BY-JOB SESSION DISPLAY--------- DELETE SUCCESSFUL
OPTION  ===> ____________________________________           SCROLL ===> PAGE

     Start-By-Job data capture sessions.  Use char "D" in delete
     column to remove an entry from the queue.

                       PF3/END TO EXIT
  DELETE JOB NAME  STEP          ADD-TO-Q     ADD-Q    SESSION    USERID
                   NAME              DATE      TIME    DURATION
                                 MM/DD/YYYY
     _    *DLT'D *               03/07/2013     11.40     STEP       BOB
******************************* Bottom of data ********************************
```

Figure 2-2-2.

2-3 Display/Cancel Session Requests In Start-By-Time Queue

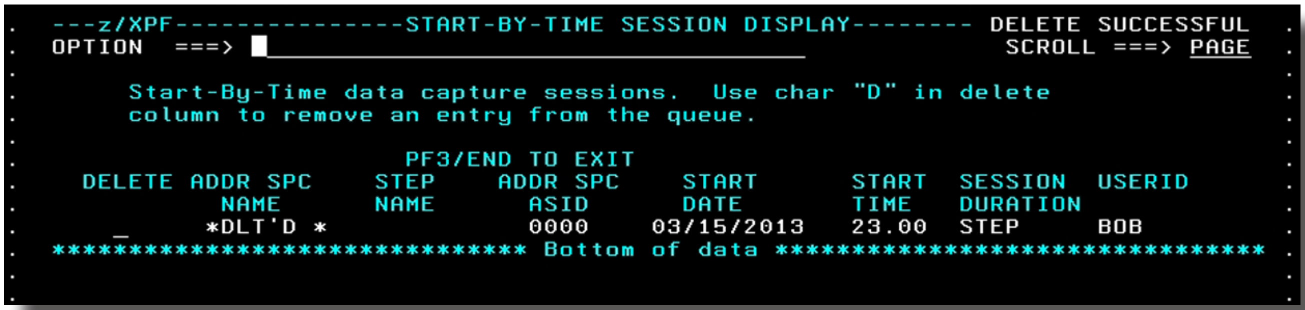
When Option 3 is selected from the panel entitled, “DISPLAY OR CANCEL SESSION QUEUES” (see Figure 2-1), and when there are one or more requests in the start-by-time queue, the panel entitled, “START-BY-TIME SESSION DISPLAY” is displayed. Again, put a “D” next to the data capture you want to stop. Refer to figure 2-3-1 below:



```
---z/XPF---------------START-BY-TIME SESSION DISPLAY--------------- Row 1 of 1
OPTION  ===> ____________________________________           SCROLL ===> PAGE

     Start-By-Time data capture sessions.  Use char "D" in delete
     column to remove an entry from the queue.

                       PF3/END TO EXIT
  DELETE ADDR SPC    STEP    ADDR SPC    START       START  SESSION  USERID
           NAME      NAME      ASID      DATE        TIME   DURATION
     _    JES2                 0000    03/15/2013   23.00  STEP      BOB
******************************* Bottom of data ********************************
```

Figure 2-3-1.

Figure 2-3-2 below shows the display function, after the second entry in the list has been deleted:



```
---z/XPF---------------START-BY-TIME SESSION DISPLAY-------- DELETE SUCCESSFUL
OPTION  ===> ____________________________________           SCROLL ===> PAGE

     Start-By-Time data capture sessions.  Use char "D" in delete
     column to remove an entry from the queue.

                       PF3/END TO EXIT
  DELETE ADDR SPC    STEP    ADDR SPC    START       START  SESSION  USERID
           NAME      NAME      ASID      DATE        TIME   DURATION
     _    *DLT'D *             0000    03/15/2013   23.00  STEP      BOB
******************************* Bottom of data ********************************
```

Figure 2-3-2.

Chapter 3 - z/XPF's Control Statements

3-1 Input statements for z/XPF's started task (the server address space)

At installation time, the installing Systems Programmer will have made decisions about how to configure and run z/XPF at your site. These global settings are communicated to z/XPF through Control Statements contained in an INPUT DD dataset in the z/XPF Server's JCL. z/XPF scans this dataset whenever its server address space is started.

It's possible to run z/XPF with only one control statement (the "LC=xxxxxx" card which activates the product). All other control statements are optional. However, as your understanding and use of z/XPF progresses, you may wish to insert or modify Control Statements to change how it interacts with your environment.

Wherever there is a default value for a Control Statement, that value will be underlined in this book.

Note that it is possible to run multiple instances of z/XPF. In the unlikely event that this becomes desirable, the SSNAME Control Statement must be used, and must be uniquely named from other instances of z/XPF.

3-2 z/XPF Control Statements Explained

LC=xxxx-xxxx-xxxx-xxxx

Where the parameter "xxxx-xxxx-xxxx-xxxx" is a 20-character activation code supplied by Duke Software. **This control statement is mandatory. z/XPF will not initialize without it.**

When a new trial of z/XPF is requested, a start and end date is negotiated by management at the installing customer site. When z/XPF has been installed and if the current date is before the formal start date of the trial, z/XPF will run, but with a limitation on the number of events it will capture for any data capture job. This is so the installing Systems Programmer can verity z/XPF's proper installation. When the formal start date arrives, then z/XPF will run without restriction.

That's the most important control statement, so we put it first. The rest appear in alphabetic order.

ALOCVOL=xxxxxx

Use this control statement to place profile data capture datasets on a specific volume.

With this statement set, the dynamic allocation routine constructs the parameter list requesting the allocation on this volume.

There is no default value. When specified, the parameter “xxxxxx” is any valid DASD VOLSER within the installation.

If this control statement is not present, then the storage request will be satisfied using whatever storage management rules are already in place.

ALOCUNIT=xxxxxxxx

If not present, ALOCUNIT defaults to the installation’s default unit type. If stated, this control statement will direct allocation of data capture datasets to a specific unit or generic unit type. ALOCUNIT accepts a 1 to 8-character string. No validity checking is done on this character string.

DATA_CAPTURE_DSN_HLQ=xxxxxxxx

“xxxxxxxx” specifies a one- to eight-character string to be used as the high level qualifier on profile data capture datasets allocated by the started task. Any characters that are valid for a dataset name may be used.

If this control statement is not present, then the high level qualifier defaults to the sub-system name. At present, z/XPF will accept a maximum of eight characters for this value. We will expand this limitation in future releases of z/XPF.

If the character string of “USERID” is entered as the parameter for this statement, then the started task will allocate the capture dataset using the TSO USERID of the individual that scheduled the profile capture request. If this choice is made, then the user must ensure that the z/XPF started task has the authority to allocate and open for output datasets with those userids.

To reduce administrative overhead, set this value, and then give individual users authority to the datasets created by their profile capture requests.

DATA_CAPTURE_DS_BUFFERS=nnn

This Control Statement establishes the number of buffers z/XPF uses during data capture. If this control statement is not present, the default value is “15”. The higher the value, the more virtual storage is used to hold the buffers, but the number of I/Os executed to write to this dataset is reduced. The use of this control statement is highly recommended.

If specified, the parameter “nnn” is a numeric value valid for the BUFND setting for an

ACB. The specified value is used with the profile data capture dataset. The block-size for this dataset is 4K.

DB2=XXXX,SDSNLOAD=YYYYY.ZZZZZZZ

This control statement is highly desirable for measuring DB2-related programs. XXXX is a version identifier for a version of DB2 (The version identifier used to be a three character value. For DB2 Release 10 and above, it is a four-character value). YYYY.ZZZZZZ is the dataset name for that version's SDSNLOAD Library. Below is an example of this control statement for DB2 Version 8.1:

DB2=810, SDSNLOAD=DSN810.SDSNLOAD

The presence or absence of this control statement indicates to z/XPF that DB2 catalog information for DBRMs, Packages, and Plans should be acquired and added to the data capture dataset. z/XPF will then use the Call Attach Facility to connect to the target DB2 system identified during data capture.

For each version of DB2 present on the target system one statement is needed. Multiple DB2 systems at the same version level can share the same SDSNLOAD dataset.

For DBRMs bound into packages, DB2 catalog tables SYSIBM.SYSPACKAGE and SYSIBM.SYSPACKDEP are queried for bind and dependency information, and SYSIBM.SYSPACKSTMT is queried for SQL text. In order for z/XPF to access these catalog tables, the DBRM shipped with z/XPF must be bound on the target DB2 system. Also, z/XPF must be allowed to access the DBRM via installation security systems.

When a query of SYSIBM.SYSPACKAGE returns a "not found" condition for a DBRM, catalog tables SYSIBM.SYSPLAN, SYSIBM.SYSPLANDEP, and SYSIBM.SYSSTMT are accessed.

FORCE_VENDOR_TABLE=YES

This control statement will store z/XPF's common area data block in the vendor table anchor entry assigned by IBM for use by z/XPF. It is only to be used in circumstances wherein (for some reason) z/XPF's vendor table anchor slot has become corrupted (meaning the slot contained non-zero values but also did not contain z/XPF's data area).

INTERVAL_TOLERANCE_PERCENTAGE=nn

The parameter "nn" is a percentage value. The default value is 10 percent.

In a heavily loaded environment it is important to verify that z/XPF is capturing ALL the data available, without lapses. The INTERVAL_TOLERANCE_PERCENTAGE is used to compute whether z/XPF is getting control often enough.

The “Interval Rate” is the number of times per second that z/XPF gets control of the processor to scan trace records. The Interval Tolerance Value of “nn” is used as a percentage to calculate whether the achieved interval rate is within range of a “desired” interval rate.

[*For example, on our development system, we may expect to get fifty “intervals” per second, and at the end of ten seconds, we’d expect to see a total of 500 intervals. However, we may not be able to achieve exactly that number of intervals, so a ten percent “tolerance” value would be used. If the Interval Rate fell below 450 in ten seconds (90% of the “desired” Interval Rate) z/XPF would begin to generate messages.*]

Put in other words, if the achieved Interval Rate is greater than the desired Interval Rate minus the tolerance value then all is well.

After the first ten seconds of execution, z/XPF compares the achieved Interval Rate with the actual Interval Rate and computes it against the Interval Tolerance percentage. If all is well, z/XPF checks again twenty seconds later. If all is still well, z/XPF checks again thirty seconds later. If an exception is seen, z/XPF will generate messages to the session log and the 10-20-30-second monitor cycle will begin again.

MAP_LMOD_DURING_CAPTURE=<u>YES</u>/NO

z/XPF normally performs mapping functions DURING data capture. This causes the z/XPF server address space to allocate and open datasets that reside in Joblib/Steplib and Linklist as input to the Binder. This may cause a security problem for the z/XPF Server Address space. If so, you can turn off this behavior by adding the above Control Statement with a parameter of “NO” to turn mapping off during data capture. Mapping can then be done later, during the report generation phase.

If this control statement is not present, then the default value of the parameter is “YES”.

MAPLPA=NO

By default, z/XPF will attempt to map modules in the Link Pack Area (LPA) in order to create BInder Maps. Depending on the number of modules in the LPA, this process can take several minutes and consume a large amount of the CPU. This defaul action can be over-ridden by includeing the MAPLPA=NO control statement in z/XPF’s startup deck.

MAP_LPAMOD=xxxxxxxx,DSN=yyyyy

Use this control statement to inform z/XPF of the location of LPA resident Load Modules that are not mapped during z/XPF's normal initialization. z/XPF will thereafter use this information when calling the Binder to create Csect maps for the named Load Module.

In this control statement, xxxxxxxx should contain the Load Module name, and yyyy the dataset name to be used for the Binder dialogue.

You may define no more than 100 MAP_LPAMOD statements to z/XPF.

z/XPF's initialization logic uses the LPAT table, mapped by Dsect IHALPAT as the source for the calls to the Binder. When the mapping logic has processed the last dataset in the LPAT, and there are LPA resident modules not yet mapped, message XPF000E-03 is written to the ZXPFLOG for each Load Module not yet mapped. These modules could be "candidates" for this special mapping function. If you wish to have z/XPF map these modules, add a MAP_LPAMOD control statement for each module.

MAX_MSG_DURING_CAPTURE=<u>1000</u>/NNNNNN

This parameter sets an upper limit on the number of messages generated by z/XPF's server address space. It prevents z/XPF's data capture from writing redundant messages to the z/XPF log. If this number is exceeded, then z/XPF may be in a loop, and all active data capture sessions are stopped.

If this control statement is not present, the default value of the parameter is "1000". This can be set to any value desired up to 999999. It may be overridden by keying in all zeros, thusly: "000000" In that situation, there will be no limit on the number of messages logged.

NBR_COPYCYCLES_PER_SECOND=<u>50</u>/nnn

The parameter "nnn" is used to compute the target for the number of times per second z/XPF will scan for events. At the beginning of each interval, z/XPF notes the time the interval started. When all of the interval processing is completed, the start time for the next interval is computed, and the current time is subtracted from the next interval start time, to give the pause time.

If this control statement is not present, the defaults value is "50". If specified, "nnn" is any value between 1 and 100.

If events occur that z/XPF doesn't seem to capture, then it is possible that this parameter has been set to a value that is too low.

PR_BUFFERS=nnn

In order to minimize “wrapping” of system Trace Buffers (and subsequent data loss by z/XPF), z/XPF can adjust the number of processor Trace Buffers by executing the “TRACE ST” Operator command during z/XPF’s initialization. The larger the Trace Buffers, the greater the chance that z/XPF will be able to keep up with execution in very fast-throughput environments.

z/XPF will enter the commands thusly: “TRACE ST,nM”

For z/OS V1R10 and above, the system default is 256 4K buffers for a trace table of 1 megabyte. In these systems, z/XPF will set the processor trace table to 512 buffers per processor, for a trace table of 2 megabytes. The maximum value z/XPF will accept on this statement is 1280, or 5 megabytes per processor. Any value greater than that will be set to 1280. Any value less than 256 will be set to 256. Any value entered in the control statement that is less than the currently set amount will be ignored.

PR_BUFFERS=ASIS

If this statement is present, z/XPF will not adjust the trace table size. Be advised that this may prevent z/XPF from capturing all of the trace records it needs in order to do its job.

RACF_PROFILE=”hlq”

If specified, the parameter “hlq” is the High Level Qualifier for a set of RACF security profiles. If unspecified, the value is a default of “ZXPF”.

Later, during initialization, a RACROUTE is executed to create a list inside the z/XPF address space of the profiles that are relevant to z/XPF.

When a request is to be added to one of the queues, a RACROUTE authorization check is made. The entity used for the check will use the parameter given in the RACF_PROFILE statement.

Potential issue: As long as a user does not specify a RACF_PROFILE= statement in any instance of z/XPF, or specifies the same value in all instances, all is well. HOWEVER, if one instance of z/XPF specifies a RACF_PROFILE statement, and then the installation starts another instance of z/XPF, but DOES NOT specify the same high level qualifier for that instance, *then that instance of z/XPF will run un-protected.*

The easiest course of action is to take the default, or specify the same value in the RACF_PROFILE= statement on all instances of z/XPF.

RESET_SRVCLASS=XXXXXX

If this statement is used, it will cause z/XPF to issue an Operator RESET command to set z/XPF's service class to the named service class. The default value for z/XPF is "SYSSTC".

RESTARTDSN=dsname

If specified, the parameter "dsname" is the name of a dataset. This dataset is used to hold profile requests in the z/XPF started task queues when the z/XPF server terminates. During start-up it is allocated and read. All requests in the start-by-jobname queue are restored. Any request in the start-by-time queue that has not expired is restored.

If the RESTARTDSN statement is used, then the parameter "DSNAME" dynamically allocates a dataset of that name (if it is not already present when the task initiates) as a physical sequential, fixed block file, with a block-size of 9600. It is a one-track dataset, with a one-track secondary allocation.

SLIP_COMMANDS=YES/<u>NO</u>

If this control statement is not present, the default value of the parameter is "NO".

Specify "YES" to allow profile data capture sessions to include SLIP PER interrupts. z/OS allows only one SLIP of this type to be active at a time.

When an active profile capture session contains a request for SLIP records, z/XPF compares the identified load modules in the target profile address space to the load module name in the user's request. When a match occurs, a SLIP command is created and sent to z/OS via MGCRE (SVC 34) if z/XPF does not already have a SLIP active at that time for another profile session.

When the profile data capture session terminates, z/XPF checks to see if a SLIP was issued. If it was, it constructs another command to terminate SLIP processing, and submits that to z/OS again using MGCRE.

SLIP_ID=xxxx

If this control statement is not present, z/XPF will default to using the sub-system name given in the SSNAME statement. Specify any set of one- to four-characters that are valid for the ID used in a SLIP command. Note: this ID should be unique among SLIP IDs used in the installation. That is, pick a SLIP ID that will ONLY be used by z/XPF.

SSNAME=ZXPF/xxxx

The value specified here is used as the z/XPF sub-system name. If this control statement is not present, the default value is “ZXPF”.

Many instances of z/XPF may run concurrently, but each must be uniquely named. If specified, the parameter “xxxx” can be any one- to four-character name.

SSCLEAR=YES/NO

If this control statement is not present, the default value for the parameter is “NO”.

Use this statement with caution. If stated, this statement will cause z/XPF to clear the Subsystem Control Table for a previous instance of z/XPF with the same name as given in the SSNAME parameter. This is useful in situations where a previous instance of z/XPF has terminated abnormally.

If there is NOT another z/XPF server address space active using the name specified by the SSNAME control statement, then setting this to YES cannot do any harm. HOWEVER, when another z/XPF server address space is active and is using the same SSNAME parameter, using SSCLEAR=YES will cause errors within the other active server address space.

USER_TRACE_NBR=0-F

If this control statement is not present, the default value of the parameter is “F”.

This control statement is used in conjunction with the ZXPTRAC “User Trace” feature. ZXPFTRAC allows you to create your own “Trace Records” that are written to system Trace Buffers and later reported on by z/XPF. The feature allows you to signal events in complex code flows,

For example, you could execute a ZXPFTRAC with TYPE=BEGIN into your code prior to scheduling it to run on a zIIP processor, and then execute a ZXPFTRAC with a TYPE=END once the code is running on the zIIP processor. In this way, you can measure the amount of system overhead it takes to get your code ported over to the zIIP processor.

WRITE_TO_LOGREC=YES

This control statement is a diagnostic tool for special circumstances. It is used to force the writing of log records when trouble-shooting z/XPF’s SRB-based data capture logic. Specify this control statement only at the direction of z/XPF’s Technical Support personnel.

Chapter 4: z/XPF's Auxiliary Macros

4-1 ZXPFTRAC – z/XPF's "User Trace" Function

The z/XPF install library &HLQ.INSTALL.JCL contains a member named ZXPFTRAC. The ZXPFTRAC member in this library is a macro that, when assembled into and executed within a user application, causes a Trace Record to be written to the system Trace table.

ZXPFTRAC is a way for z/XPF customers to create their own "Trace records" in order to signal various events in their applications. So, one could use the "TEXT=" operand to send a "HELLO WORLD" message to the Trace table (and thereafter see it in z/XPF's Detail reports). One could also use the "BEGIN" and "END" operands to measure the elapsed time for execution between any two places in a program.

ZXPFTRAC can be executed in any environment: TCB-mode, SRB-mode, Authorized, unauthorized, PASN=HASN, or PASN not equal HASN – any environment that your code can execute in - with the exception of a TIMER DIE exit.

For example, you could execute a ZXPFTRAC with TYPE=BEGIN into your code prior to scheduling it to run on a zIIP processor, and then execute a ZXPFTRAC with a TYPE=END once the code is running on the zIIP processor. In this way, you can measure the amount of system overhead it takes to get your code ported over to the zIIP processor.

ZXPFTRAC will only work when the z/XPF Server is active and will have no effect during un-monitored program execution.

Currently, ZXPFTRAC-generated Trace Records will only appear in z/XPF's Detail reporting, NOT in Summary reporting.

[Note: If more than one z/XPF Server address space is active, then the ZXPFTRAC macro will work ONLY on that first instance of the z/XPF server address space.]

4-2 How to use ZXPFTRAC

To assemble correctly, the application source will need to contain:

1. Register equates for R1, R14 and R15;
2. CVT dsect (CVT DSECT=YES, LIST=YES);
3. ECVT dsect (IAHECVT LIST=YES).

Copy the macro from the &HLQ.INSTALL.JCL library to a macro library on your system.

The macro has two operands: “TYPE=” and “TEXT=”.

- Use TYPE=BEGIN to signal the beginning of a process.
- Use TYPE=END to signal the end of a process.
- Use TEXT=’<string>’ will accept any text stream coded within single quotes.
- You MUST use both TYPE= and TEXT= operands within a single occurrence of ZXPFTRAC.

ZXPFTRAC executes a space switch Program Call (The logic for the PC resides within the z/XPF server, and is extremely short). When the ZXPFTRAC macro is executed, a user Trace Record is written to the system Trace Table using PTRACE. This user Trace Record is picked up from the Trace Table and is included in the capture dataset.

The overhead involved in using this functionality is as follows:

- Space Switch Program Call.
- CPOOL GET to acquire a cell from a pool.
- ESTAEX to protect the function.
- PTRACE.
- ESTAEX to remove the estae
- CPOOL FREE
- Program Return to return to the application

The user trace number used on the PTRACE call is a default of x’F’ (decimal 15). You may alter this value by using a z/XPF input control statement of:

USER_TRACE_NBR=[0-F] (where “F” is the z/XPF default value).

Note that the logic in the macro checks z/XPF control blocks to insure that the z/XPF server address space is active, so no harm should come from executing the code if the server is not currently active. However, it is recommended that the execution of the ZXPFTRAC code be controlled within the user’s application by some form of user flag setting.

4-3 Understanding the output of ZXPFTRAC

- *ZXPFTRAC’s results appear only in z/XPF’s Detail Reports* (not in Summary reports).
- There is a new INCLUDE filter for ZXPFTRAC in z/XPF’s Detail Reports. If you use this filter then your Detail report will contain only your ZXPFTRAC records. Now, they’ll be isolated!

Here is an example taken from the test-bed application we used to verify z/XPF’s accuracy:

```
CPU=00, TIME RECORDED=11.02.16:53.8350, RECNUM=46425/015C,CMPT’D, PSW ID’D
EVENT=USR TRC 0F7F, TIME=11.02.16:50.7364, TYPE=BEGIN TRACE
PASN=0028,HASN=0028, PSW=078520008000000000000000007E58, LOC=PRV AREA
KEY=8, STATE=PROB, ASC=PRIM, MODE=31,INSTR ADDR=00007E58
LM=APIVPZRO,OFFSET=000009C8
CSECT=APIVP,OFFSET=000009C8
```

```
WORK UNIT=APIVPZRO, TOKEN=000000A000000000100000044008E66D0
TEXT=BRACKET GETMAIN/STORAGE FUNCTIONS

CPU=00, TIME RECORDED=11.02.16:53.8439, RECNUM=46450/015C,CMPT'D, PSW ID'D
EVENT=USR TRC 0F7F, TIME=11.02.16:50.7771, TYPE=END TRACE
PASN=0028,HASN=0028, PSW=07852000800000000000000000008256, LOC=PRV AREA
KEY=8, STATE=PROB, ASC=PRIM, MODE=31,INSTR ADDR=00008256
LM=APIVPZRO,OFFSET=00000DC6
CSECT=APIVP,OFFSET=00000DC6
WORK UNIT=APIVPZRO, TOKEN=000000A000000000100000044008E66D0
TEXT=BRACKET GETMAIN/STORAGE FUNCTIONS
```

Explanation:

- In both paragraphs, note the character string "EVENT=USR TRC 0F7F" in the second record.
- The TIME= value is taken from the user trace record created by the PTRACE call. Note the time-stamp.
- Note the PSW address. This is the location in the application program where the ZXPFTRAC was executed.
- Finally notice the TYPE= in the second record line of the formatted trace entry. This field contains the value from the TYPE=BEGIN execution of the macro in csect APIVP, at offset 9C8. The entry for TYPE=END appears at offset DC6 in the second paragraph.
- The last line of data for the event is the content of the TEXT= coded on the ZXPFTRAC.

4-4 ZXPFDYNL - Tracking the “Life-span” of modules loaded by Directed Loads

Load Modules that are loaded by a Directed Load do not add entries to the application CDE chain. And, obviously, executable code that is generated while the application is executing is not known to any z/OS function or service, and therefore these routines also do not have CDE chain entries.

z/XPF’s data capture logic now has the ability to identify Load Modules that are loaded via a Directed Load. z/XPF can handle two distinct types of Load Modules:

- Type 1. A standard s/OS Load Module or Program Object .
- Type 2. A Load Module that is “generated code’” This is executable code that is built by the application logic while the application is executing, and is moved to a virtual storage location.

To give z/XPF the ability to report on these Load Modules, it is necessary to give the target application the ability to communicate with the z/XPF server. That communication is in the form of a Program Call from the application to the server. The Program Call will be added to the application’s logic by incorporating a z/XPF supplied macro named “ ZXPFDYNL”.

Once you have used the ZXPFDYNL macro, then the modules you identify to z/XPF will appear in both Summary and Detail reporting. At that point they’re “just another module” to z/XPF, and you’ll be able to gather statistics on them just as you do with “normally loaded” modules.

[Note: If more than one z/XPF Server address space is active, then the ZXPFDYNL macro will work ONLY on that first instance of the z/XPF server address space.]

4-5 ZXPFDYNL macro specifications.

The macro has one optional keyword, and that keyword will have two possible values:

- ZXPFDYNL TYPE=TEST.
  - TYPE =TEST will execute logic to determine if the z/XPF server is active on the z/OS image. R15 will contain a return code for this invocation of the macro.
  - If Register 15 = 0’s, then the z/XPF server address space is active, and is capturing data for this process.
  - If R15 = 4, then the z/XPF server is active, but is not capturing data for this process.
  - If R15 is > 4, then the z/XPF server is not currently active.

- ZXPFDYNL TYPE=GENERATE
  - TYPE=GENERATE will define the data fields needed by the application to communicate the loading and deletion of these dynamically loaded modules. The data fields are:

```
ZXPFDNAM      DS     CL8   Name of the Load Module
ZXPFDADR      DS     XL8   Virtual Storage address of module load point.
ZXPFDLEN      DS     XL4   Length of Load Module
ZXPFDTOD      DS     XL8   STCK (clock) value when a module is loaded/deleted
ZXPFDTYP      DS     CL1   Action indicator.
```

- Value = "L" to indicate a load
- Value = "D" to indicate a delete
- Value = "G" to indicate generated code

ZXPFDYNL without any keyword coded triggers the logic to place the address of the generated ZXPFDNAM field in R1, and then execute the Program Call to the server address space.

The Program Call logic within the z/XPF server uses the home ASID value to determine if data capture is active for the address space executing the Program Call. If not active, the logic will immediately return, and R15 on the return will contain 0's. No indication will be given to the caller that the call was not successful.

When the ZXPFDTYP field is either "L" or "G", z/XPF's internal data structures are updated to reflect the contents of the other fields passed on the call.

When the type field is "L" (indicating a load operation), the Binder is called to get a Csect map for the Load Module, using the target application's Joblib/Steplib and current Link list datasets.

No attempt to get a Binder map for Load Modules with a type of "G" is be made, because it is not possible to obtain mapping data for generated code.

When the ZXPFDTYP value is "D" the server logic removes the previously identified Load Module from its current, active target application Load Module identification data structures, and places this entry in the non-active chain. Note that each execution of ZXPFDYNL to identify a load and a delete of a Load Module causes the server to update its data structures.

It is possible to track the same Load Module loaded at different times and/or virtual storage locations.

When/if the target application executes two successive calls with a type of "L" for the same virtual storage location, the 2nd call will cause an implicit delete for the previously defined Load Module.

To allow the user to verify the information contained within the ZXPFDYNL data structures, the z/XPF server logs the contents and type of call to the ZXPFLOG dataset in the server address space.

ZXPFTRAC executes a space switch Program Call (The logic for the PC resides within the z/XPF server, and is extremely short). When the ZXPFTRAC macro is executed, a user Trace Record is written to the system Trace Table using PTRACE. This user Trace Record is picked up from the Trace Table and is included in the capture dataset.

The overhead involved in using this functionality is as follows:

- Space Switch Program Call.
- CPOOL GET to acquire a cell from a pool.
- ESTAEX to protect the function.
- PTRACE.
- ESTAEX to remove the estae
- CPOOL FREE
- Program Return to return to the application

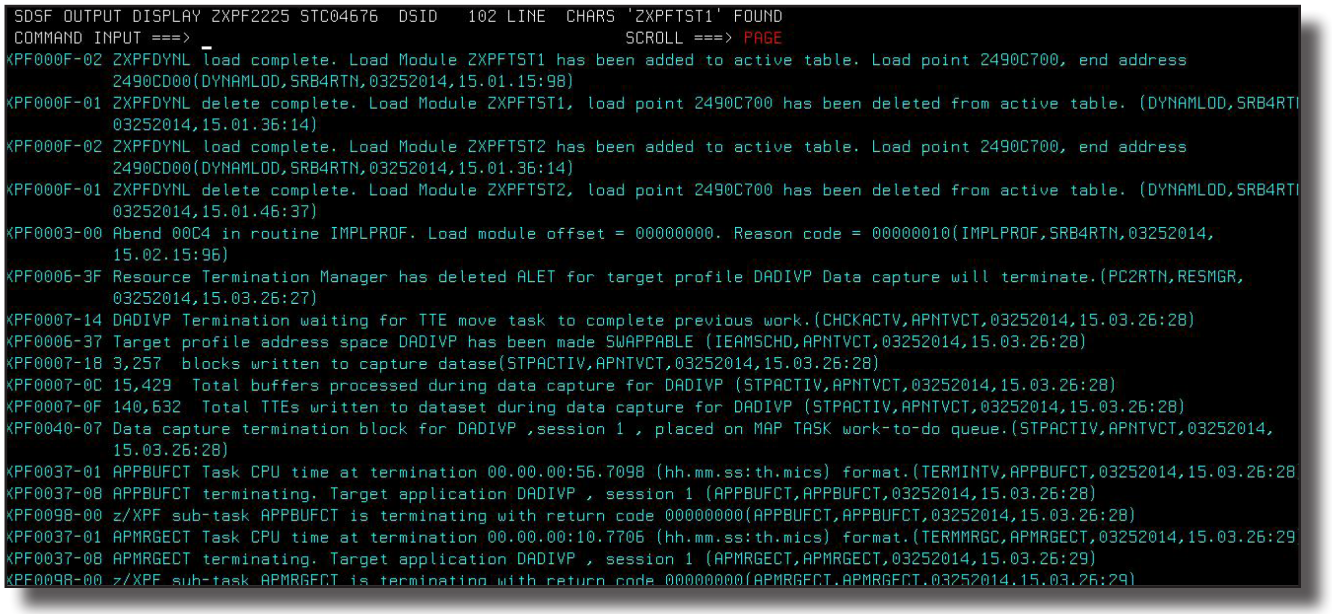
The user trace number used on the PTRACE call is a default of x'F' (decimal 15). You may alter this value by using a z/XPF input control statement of:

USER_TRACE_NBR=[0-<u>F</u>] (where "F" is the z/XPF default value).

Note that the logic in the macro checks z/XPF control blocks to insure that the z/XPF server address space is active, so no harm should come from executing the code if the server is not currently active. However, it is recommended that the execution of the ZXPFTRAC code be controlled within the user's application by some form of user flag setting.

4-6 Tracking the activity of the ZXPFDYNL Macro

Whenever the ZXPFDYNL macro is invoked, you will be able to see its activity in the ZXPFLOG dataset. In our example two modules, "ZXPFTST1" and "ZXPFTST2" were reported on. Below in Figure 4-6-1 you will see an example:



```
 SDSF OUTPUT DISPLAY ZXPF2225 STC04676  DSID   102 LINE  CHARS 'ZXPFTST1' FOUND
 COMMAND INPUT ===> _                                      SCROLL ===> PAGE
XPF000F-02 ZXPFDYNL load complete. Load Module ZXPFTST1 has been added to active table. Load point 2490C700, end address
           2490CD00(DYNAMLOD,SRB4RTN,03252014,15.01.15:98)
XPF000F-01 ZXPFDYNL delete complete. Load Module ZXPFTST1, load point 2490C700 has been deleted from active table. (DYNAMLOD,SRB4RTI
           03252014,15.01.36:14)
XPF000F-02 ZXPFDYNL load complete. Load Module ZXPFTST2 has been added to active table. Load point 2490C700, end address
           2490CD00(DYNAMLOD,SRB4RTN,03252014,15.01.36:14)
XPF000F-01 ZXPFDYNL delete complete. Load Module ZXPFTST2, load point 2490C700 has been deleted from active table. (DYNAMLOD,SRB4RTI
           03252014,15.01.46:37)
XPF0003-00 Abend 00C4 in routine IMPLPROF. Load module offset = 00000000. Reason code = 00000010(IMPLPROF,SRB4RTN,03252014,
           15.02.15:96)
XPF0006-3F Resource Termination Manager has deleted ALET for target profile DADIVP Data capture will terminate.(PC2RTN,RESMGR,
           03252014,15.03.26:27)
XPF0007-14 DADIVP Termination waiting for TTE move task to complete previous work.(CHCKACTV,APNTVCT,03252014,15.03.26:28)
XPF0006-37 Target profile address space DADIVP has been made SWAPPABLE (IEAMSCHD,APNTVCT,03252014,15.03.26:28)
XPF0007-18 3,257  blocks written to capture datase(STPACTIV,APNTVCT,03252014,15.03.26:28)
XPF0007-0C 15,429  Total buffers processed during data capture for DADIVP (STPACTIV,APNTVCT,03252014,15.03.26:28)
XPF0007-0F 140,632  Total TTEs written to dataset during data capture for DADIVP (STPACTIV,APNTVCT,03252014,15.03.26:28)
XPF0040-07 Data capture termination block for DADIVP ,session 1 , placed on MAP TASK work-to-do queue.(STPACTIV,APNTVCT,03252014,
           15.03.26:28)
XPF0037-01 APPBUFCT Task CPU time at termination 00.00.00:56.7098 (hh.mm.ss:th.mics) format.(TERMINTV,APPBUFCT,03252014,15.03.26:28
XPF0037-08 APPBUFCT terminating. Target application DADIVP , session 1 (APPBUFCT,APPBUFCT,03252014,15.03.26:28)
XPF0098-00 z/XPF sub-task APPBUFCT is terminating with return code 00000000(APPBUFCT,APPBUFCT,03252014,15.03.26:28)
XPF0037-01 APMRGECT Task CPU time at termination 00.00.00:10.7706 (hh.mm.ss:th.mics) format.(TERMMRGC,APMRGECT,03252014,15.03.26:29
XPF0037-08 APMRGECT terminating. Target application DADIVP , session 1 (APMRGECT,APMRGECT,03252014,15.03.26:29)
XPF0098-00 z/XPF sub-task APMRGECT is terminating with return code 00000000(APMRGECT,APMRGECT,03252014,15.03.26:29)
```

Figure 4-6-1.

In the panel above you'll see Message XPF000F-02 showing, "ZXPFDYNL load complete". You can see that Load Module ZXPFTST! was loaded, it's load point and its ending address. Next you can see Message XPF000F-01 showing, "ZXPFDYNL delete complete". Later you see the same pair of messages for ZXPFTST2.

Chapter 5 - Creating Reports

OK. By this point you have created a data capture session, and z/XPF has captured events in a VSAM database. Now it's time for the reporting phase of your work with z/XPF.

During data capture, z/XPF captures Trace Records for the address-space-of-interest you request, and filters out Trace Records that are irrelevant to your enquiry. Then, it saves the data into VSAM datasets that you can query. Some of these report datasets can be VERY large.

The two tools you'll use are Summary Reporting, and Detail Reporting.

5-1 Summary Reporting vs. Detail Reporting

z/XPF offers two categories of reports: Summary reports, and Detail reports. This chapter will cover z/XPF's basic reporting concepts and Summary Reporting. A subsequent chapter will discuss Detail reporting.

Summary reports contain information about program activity, Wait time, Contention, memory management and the like right down to the PSW level. You merely put the cursor next to any field that you care to see, and press Enter. z/XPF will either "drill down" into the next level or will drill and expand, which is a way to drill down while changing categories. It's probable that you will be able to do most of your work using Summary reports.

z/XPF will allow you to see your source statements in Summary Reports. You'll find information on how to do that further along in the book, towards the end of Chapter Five.

Detail reports go down to individual Trace Records themselves. If you're using the ZXPFTRAC facility (which allows you to create your own trace records), then Detail reporting is the ONLY way to view them in z/XPF's reports.

Detail reporting involves learning to use and employ "filters" that include/exclude extraneous data in order to get to the information you need. The better you get at using filters, the quicker you'll get what you need with z/XPF's Detail Reporting. The trick is to eliminate the potential millions of data points that don't matter to you in order to see what you DO need to see.

5-2 z/XPF's Reporting Hierarchy

z/XPF's reporting hierarchy proceeds from the general to the specific, just as you might do in your own investigative process. There are five "levels" to a z/XPF report.

1. The top level is the Time Segment. You can have one Time Segment that spans the entire length of your data capture session, or you can divide as finely as you like down to one-second Time Segments. Time Segments allow you to isolate the most active

portions of a data capture dataset.

2. The next level is the Work Unit. This is either a Task or an SRB.
3. Below the Work Unit level is the load module.
4. Below the load module is the csect.
5. The final level is the actual PSW offset within a csect.

With z/XPF's dynamic ISPF reporting you can navigate up and down through the hierarchy (A "drill-down" operation) or drill-and-change-categories (a "drill and expand" operation).

5-3 z/XPF and Virtual Storage

z/XPF trades virtual storage usage for speed. It is impossible to predict the amount of virtual storage that z/XPF will need to create its reports because the amount of virtual storage required is dictated by the size of the dataset created during data capture.

With z/XPF V2R1 and above, both the z/XPF Server and the Report Generation functions use virtual storage obtained above the 2-gigabyte "bar". Therefore:

You MUST have enough space made available to your user's TSO userids to process the reports. If your users get storage ABENDs during the report phase in z/XPF, that's an indicator that they need more space. The ultimate fix is to define a region size of 2 gigabytes, which means that z/XPF will get all it needs.
ISPF must be able to allocate storage above the bar. Just how much storage needed above the bar depends on the number of Work Units and load modules that are reported on during z/XPF's data capture.
it is recommended that the installing Systems Programmer set the MEMLIMIT value in the SMFPRMxx member to NOLIMIT.

z/XPF also expects to write to its ISPF log dataset. If no log dataset is allocated, then z/XPF will not be able to log ISPF error messages, and diagnosis of z/XPF problems within ISPF will be far more difficult

5-4 How z/XPF Maps Load Modules

"Mapping" allows z/XPF to match PSW-based Trace Records to specific csects and load modules. That way, in z/XPF's reports you can easily tell where your applications are consuming system resources, which is a vital feature of the product. z/XPF does NOT map to source code in any language - just to PSW within csect within load module.

Mapping is normally done during data capture, and at the end of data capture maps are written into the data capture dataset. This is the system default. You can over-ride this function if you care to by using the z/XPF Control Statement "MAP_LMOD_DURING_CAPTURE=NO" where the system default is "YES".

If your z/XPF is set not to map during data capture, you can influence this later. In this case you can ask z/XPF to perform the mapping function when you create your reports. That choice becomes available to you at the panel entitled “ALLOCATE DATA CAPTURE DATASET”. Enter “YES” in the field labeled, “Map load modules during allocation process. (yes/no)”.

You can say “NO” to mapping at this juncture. If you do this, and maps WERE created during data capture, then if they do exist, they’ll be ignored. I don’t know why anyone would make this choice, but it’s available to you.

5-5 Mapping “User-defined” Load Modules

z/XPF can also map “user-defined” load modules. These load modules do not possess a CDE entry in the CDE chain. They’re loaded in other ways by clever programmers, but z/XPF can still map them. The way to map “non-CDE” modules is to create a dataset that specifies the name, location and length of the non-CDE load module(s).

The dataset contains records that follow this format:

The keyword “LMOD=”, in columns 1-5.
The load module name starting in column 6, followed by a comma.
The virtual storage address where the load module is loaded, followed by a comma, and
The length of the load module, in hexadecimal format.

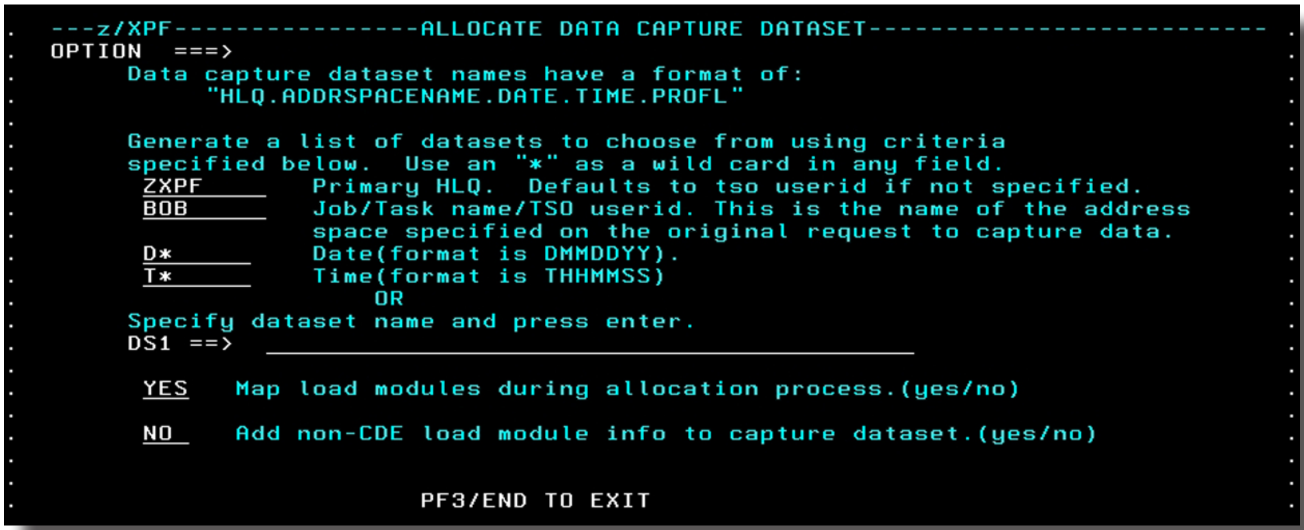
Once you have created this dataset, then in the allocation process you will enter “YES” to the panel choice “Add non-CDE load modules info to capture dataset yes/no” (See Figure 5-1). Then, z/XPF will direct you to another panel that will prompt you for the location of the dataset containing the values you stated in the bullet points above.

Another useful tool in this area is the ZXPFDYNL macro, which allows you to gather statistics on multiple instances of dynamically loaded modules that are (or are not) loaded to the same address every time. See Chapter Four for more information on the ZXPFDYNL macro.

5-6 Allocating a dataset for reporting

Before you can begin reviewing z/XPF's reports you must allocate the VSAM dataset that was created when you performed your data capture. z/XPF can only report on one data capture dataset at at time.

If z/XPF detects that no datasets are already allocated, it will automatically direct you to the panel below, entitled "ALLOCATE DATA CAPTURE DATASET".



```
---z/XPF----------------ALLOCATE DATA CAPTURE DATASET--------------------------
OPTION  ===>
     Data capture dataset names have a format of:
          "HLQ.ADDRSPACENAME.DATE.TIME.PROFL"

     Generate a list of datasets to choose from using criteria
     specified below.  Use an "*" as a wild card in any field.
      ZXPF         Primary HLQ.  Defaults to tso userid if not specified.
      BOB          Job/Task name/TSO userid. This is the name of the address
                   space specified on the original request to capture data.
      D*           Date(format is DMMDDYY).
      T*           Time(format is THHMMSS)
                       OR
     Specify dataset name and press enter.
     DS1  ==>  __________________________________________

      YES   Map load modules during allocation process.(yes/no)

      NO    Add non-CDE load module info to capture dataset.(yes/no)


                          PF3/END TO EXIT
```

Figure 5-6-1.

Note, that if you ALREADY have a data capture dataset allocated, you will not see Figure 5-1. Instead you'll be directed to the PRIMARY CREATE PROFILE MENU which appears in Figure 5-3 below. From THAT panel, you may choose to FREE the data capture dataset you already have allocated, and then select a new one from the screen depicted in Figure 5-1.

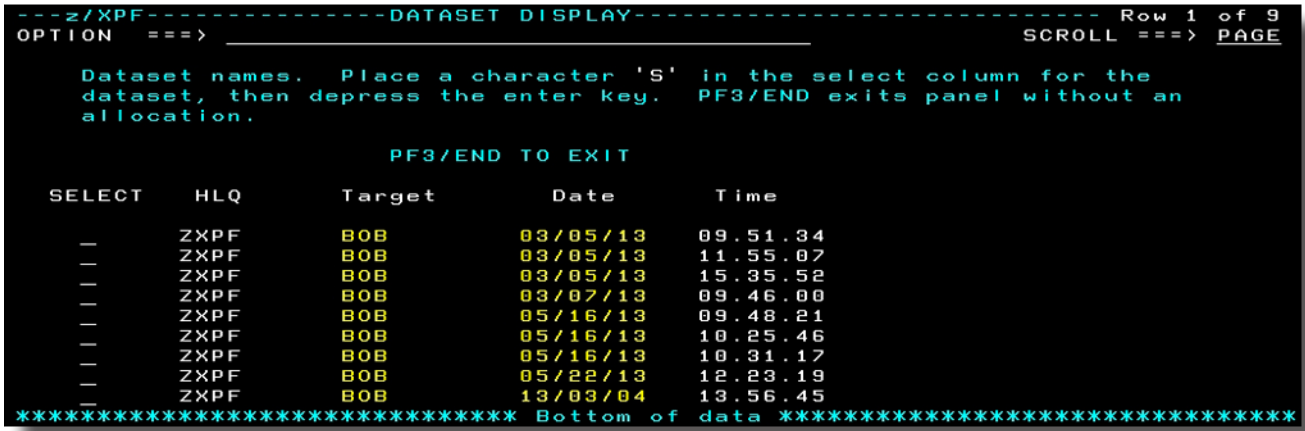
The panel in Figure 5-1 has two logical "partitions". You can either press Enter to manipulate the default/wildcard values for a z/XPF profile data set, OR you can specify a dataset name directly in the lower portion of the panel (The "DS1 ==>" field). If you specify a name here, then enclose the name in single quotation marks. You can't use both portions of the panel.

The general naming convention for data capture datasets is "ZXPF.<jobname or TSO userid>.D<MMDDYYYY>.T<HHMMSS>.PROFL". "D<MMDDYYYY>" is a date field, and the "T<HHMMSS>" field is the time of day, values which correspond to when the data capture dataset was created.

In the next field ("Map load modules during allocation process."), you can choose whether you want mapping done or not. "Yes" is the system default, and is a good idea to leave it as it is.

"Add non-CDE load module to capture dataset" is meant for the mapping of "user-defined" load modules. For more information on this, see “How to map modules that do not have a CDE-entry” later in this chapter.

Once you have filled in the Allocation panel in Figure 5-6-2, you’ll see the panel that appears in Figure 5-2 below:



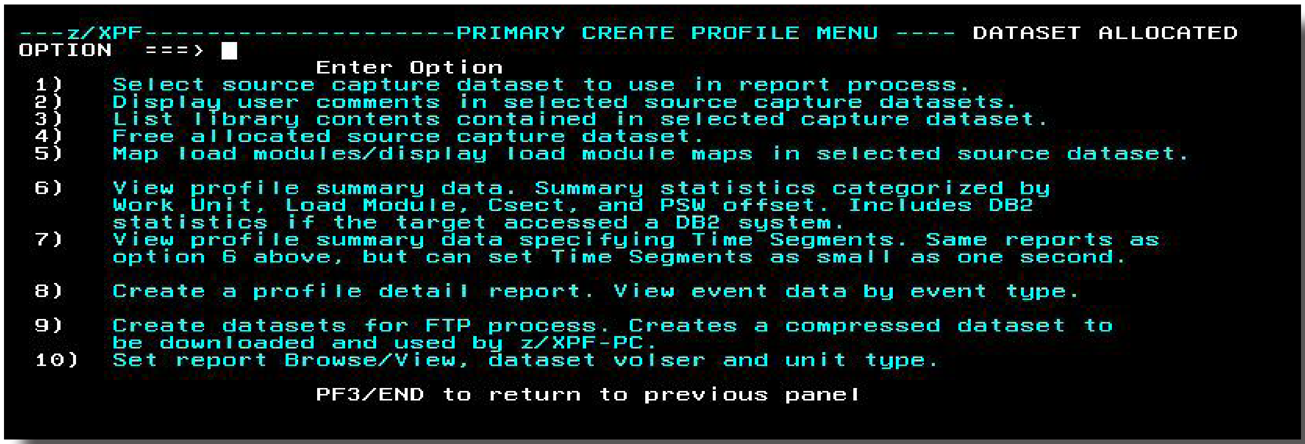
```
---z/XPF---------------DATASET DISPLAY---------------------------- Row 1 of 9
OPTION  ===> ____________________________________________  SCROLL ===> PAGE

     Dataset names.  Place a character 'S' in the select column for the
     dataset, then depress the enter key.  PF3/END exits panel without an
     allocation.

                         PF3/END TO EXIT

  SELECT     HLQ        Target         Date        Time

    _       ZXPF        BOB          03/05/13    09.51.34
    _       ZXPF        BOB          03/05/13    11.55.07
    _       ZXPF        BOB          03/05/13    15.35.52
    _       ZXPF        BOB          03/07/13    09.46.00
    _       ZXPF        BOB          05/16/13    09.48.21
    _       ZXPF        BOB          05/16/13    10.25.46
    _       ZXPF        BOB          05/16/13    10.31.17
    _       ZXPF        BOB          05/22/13    12.23.19
    _       ZXPF        BOB          13/03/04    13.56.45
******************************** Bottom of data ********************************
```

Figure 5-6-2.

In the “DATASET DISPLAY” panel you’ll see all the datasets that meet the naming criteria you specified previously. To select one from the list, tab down next to the entry you wish, enter an “S” and press the Enter key.

5-7 Beginning the Reporting Process

Once you have allocated your data capture dataset and pressed PF3 and you will see the panel entitled, “PRIMARY CREATE PROFILE MENU”. Refer to Figure 5-7-1 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. Same reports as
       option 6 above, but can set Time Segments 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 5-7-1.

Here is an explanation of the choices available on this panel:

Option 1 allocates a dataset, if one is not currently allocated.

Option 2 will display the user text comments in the currently allocated capture dataset, if any have been entered. If no comments exist, you'll see "NONE THIS DSN".

Option 3 allows the user to list the contents of libraries used during the report generation process. This panel allows you to learn which system libraries were accessed during your data capture session.

Option 4 frees the currently allocated source capture dataset. z/XPF can report on only one capture dataset at a time, so in order to create reports on another capture dataset you must free the currently allocated one. Use Option 4 to see a display of the data capture dataset you're currently looking at. Put an "F" next to the dataset and press Enter. This frees the capture dataset. Then, you may select another capture datset using Option 1.

Option 5 may be used to access Binder/link-edit maps to be used during the report generation process. Modules that are already mapped display the dataset used as source for the map. You can use "M" to map any un-mapped modules. You can use "D" to display the map. You can use "X" to delete a map.

Option 6 takes the user to the primary Summary Report generation panel. Here the user is able to create reports that contain statistical information on the performance of the target application. This is DYNAMIC ISPF reporting, a real time-saver!

Option 7 allows you to sub-divide z/XPF's reports into a series of one-second time segments. You can then use z/XPF's Summary Reporting to drill down within these time segments.

Option 8 is for Detail Reporting. Use this function to further investigate and understand the statistics contained in the Summary Reports. This is how you drill WAY down into your reports - right down to individual z/XPF-formated Trace records. There are a robust series of filters available to you in Detail reporting. The topic of Detail Reporting gets its own chapter further on in the book.

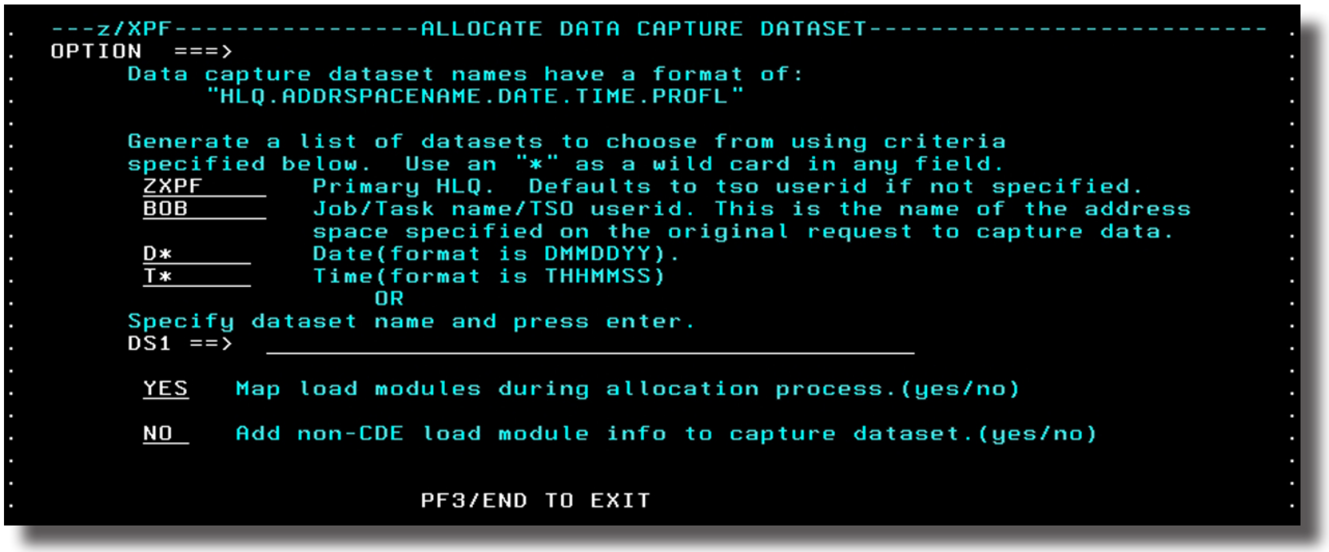
Option 9 is used to compress data capture dataset for FTP transfer to a desktop so that you may view your reports with z/XPF-PC our desktop GUI interface.

Option 10 is used to alter the allocation parameters for your data capture dataset. Here you can influence the Unit type, specify a VOLSER or specify whether you wish to use ISPF's View or Browse functions.

Throughout the following chapters of this book, we'll discuss these options in detail (except Option 10, which is fairly straightforward).

5-8 Option 1: Allocating A Data Capture Dataset

We'll repeat ourselves a bit here, because there are two ways to access the next panel. When Option one is selected from the PRIMARY CREATE PROFILE MENU panel, you will next see the panel entitled, "ALLOCATE DATA CAPTURE DATASETS" (just as we showed in Figure 5-1). See Figure 5-8-1 below:



```
---z/XPF----------------ALLOCATE DATA CAPTURE DATASET--------------------------
OPTION  ===>
     Data capture dataset names have a format of:
          "HLQ.ADDRSPACENAME.DATE.TIME.PROFL"

     Generate a list of datasets to choose from using criteria
     specified below.  Use an "*" as a wild card in any field.
       ZXPF        Primary HLQ.  Defaults to tso userid if not specified.
       BOB         Job/Task name/TSO userid. This is the name of the address
                   space specified on the original request to capture data.
       D*          Date(format is DMMDDYY).
       T*          Time(format is THHMMSS)
                           OR
     Specify dataset name and press enter.
     DS1  ==>

       YES    Map load modules during allocation process.(yes/no)

       NO     Add non-CDE load module info to capture dataset.(yes/no)


                           PF3/END TO EXIT
```

Figure 5-8-1.

The panel is divided into two logical parts. You may use the first four entry fields to select a dataset or you may specify the dataset explicitly in the fifth field (but not both at the same time).

Using the first four fields of the ALLOCATE DATA CAPTURE DATASETS panel:

This is where you can specify the individual elements of a capture dataset's name, specifying values for the High Level Qualifier, the Address Space, the Date, and Time. All datasets end with the Low Level Qualifier of "PROFL".

In the example above in Figure 5-4, the High-level qualifier is set to ZXPF - a system default. Your naming convention may be different. You may contact your installing systems programmer for this information, which is specified in the z/XPF input control statement "DATA_CAPTURE_DSN_HLQ=".

The next parameter is typically the target address space name that the data capture was run against. If that was a TSO userid, the TSO userid will appear. If it was a started task, then the started task's name will appear. If it is a batch job, then the job name will appear.

The next two fields allow you to specify (or wildcard) date and time values. You may use an asterisk("*") to end any of the fields as a wild card. You may expand or contract any of the variables to broaden or narrow the list returned.

In this example (we're still on Figure 5-4), we've used an asterisk to set the Date and Time variables to list all datasets for all dates/times.

Using the fifth field of the ALLOCATE DATA CAPTURE DATASETS panel:

You may either use the top portion of this panel to specify a data capture dataset name, OR you can name it directly (you can't do both). The "DS1 ==>" field allows you to explicitly name a dataset. Enclose this dataset name in single quotations.

The next field on the panel tells z/XPF whether or not to map load modules identified during the data capture process. The default value is "YES".

The last field allows z/XPF to map modules that were loaded by "non-standard" means, perhaps by way of a directed load. If you change this field to "YES" you'll be directed to another panel that asks you to fill in the name of the dataset which contains the statements which "map" the non-CDE module(s) (see "How To Map Modules That Do Not Have A CDE Entry" later in this chapter).

Using the sixth field of the ALLOCATE DATA CAPTURE DATASETS panel: "Map Load Modules during allocation process"

If z/XPF's default actions are taken, the user needs to do nothing. Maps will have been created during data capture, written to the data capture dataset, then read in during report initialization(after the dataset is allocated, opened, and prior to displaying the next panel).

If the user sets the value to NO in the panel in Figure 5-4 then no mapping of any kind will take place. If there are maps present either from the data capture, or from a previous report session where the user mapped load modules themselves, they are ignored.

If the value is set to YES in the, and there are no maps present in the data capture dataset, then the same mapping function that would have taken place during data capture takes place now. All load modules that were loaded into the target private area and identified during data capture will be run through the Binder, and maps will be created when the module is found. Note that these maps will then be written to the capture dataset to be used the next time the dataset is opened for report generation.

If the value is set to YES, and there are maps present for some - but not all - of the identified load modules, then those maps will be used. This means some load modules will have csect psw displays, and some will not. This can only happen if the user originally said NO to mapping, then manually mapped one/some of the modules, then closed and re-opened the dataset.

The Seventh field of the ALLOCATE DATA CAPTURE DATASETS panel: "Add non-CDE load module info to capture dataset. (yes/no)"

z/XPF has the ability to map modules that do NOT have a CDE entry. Such modules can be loaded via a “directed load” in a dynamic, as-needed basis. This technique is used by some software vendors, and z/XPF’s can map such modules.

In the “ALLOCATE DATA CAPTURE DATASETS” panel, you can specify “YES” to “Add non-CDE load module info to data capture dataset”. After the regular allocation panel, z/XPF puts up another panel that requests the name of a dataset that contains special statements that tell z/XPF where to find “non-CDE” modules. The title of this panel is “SPECIFY DATASET NAME”.

The dataset name that you enter into the “DSNAME” field follows TSO naming conventions: Use either a pure dataset name for a physical sequential file, or DSNAME(MEMBER) for a PDS. If the high level qualifier for this dataset is not your TSO userid, then surround the dataset name with single quotation marks.

In the dataset you specify in DSNAME, z/XPF expects to find one or more statements that refer to the module name and its beginning and ending addresses. These statements are in the format of:

LMOD=nnnnnnnn,BEGIN=XXXXXXXX,END=YYYYYYYY

Where:

LMOD=nnnnnnnn <== where nnnnnnnn is the name of the module, up to eight characters.
BEGIN=xxxxxxxÿ <== where xxxxxxxx is the virtual address of the start of the module.
END=yyyyyyyyÿ <== where yyyyyyyy is the virtual address of the end of the module.

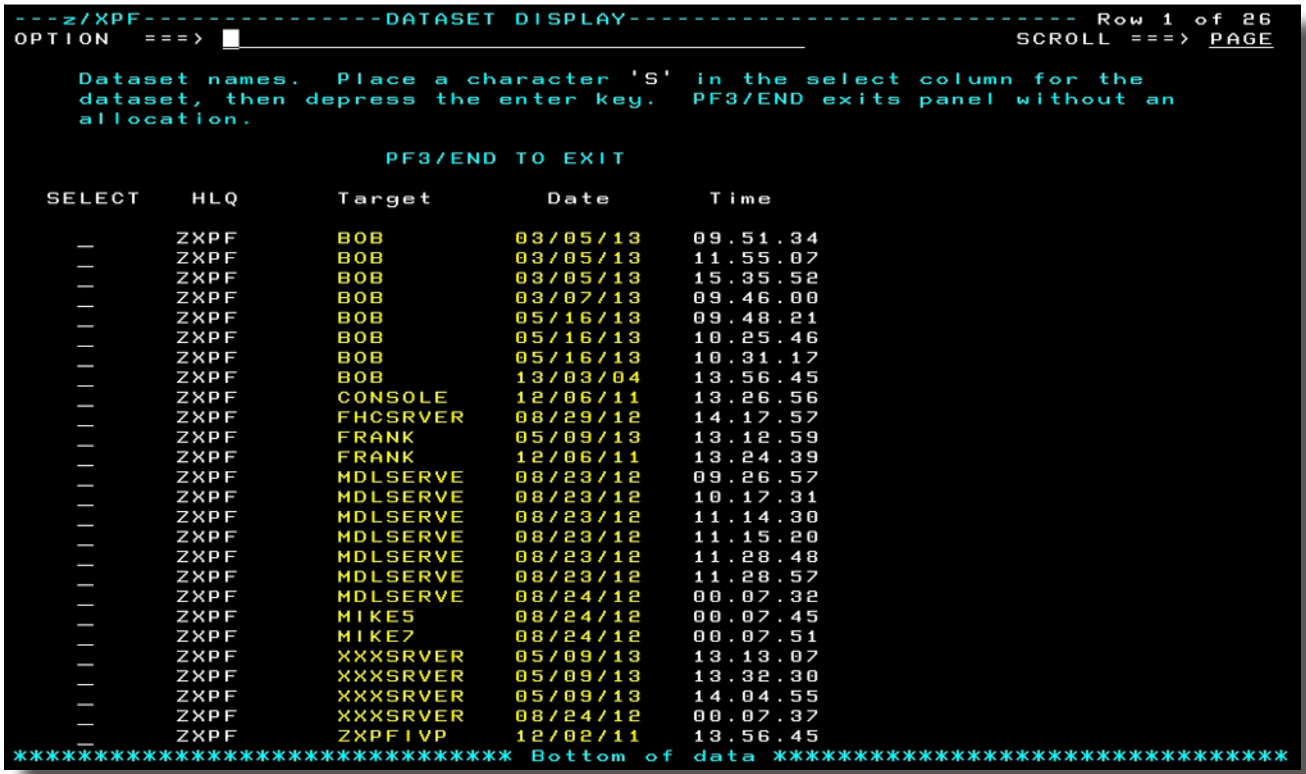
You may define one or more LMOD= statements in your dataset. If z/XPF detects an error in any LMOD= statement, then the incorrect statement(s) will be logged to the ISPF log dataset.

When z/XPF accesses the DSNAME you specify, it will search your STEPLIB or LINKLIST libraries and will map the load modules. Otherwise, you may manually map them using Option 5 from the PRIMARY PROFILE CREATE MENU display.

5-9 Option 2: Display user comments in selected source data capture datasets

When you schedule a data capture with z/XPF, you’re presented with the opportunity to add text comments that will be included in the capture dataset for later review. If you select Option 2 from the panel in Figure 5-3 you will see these comments if any have been created. Tect commenting is a handy way to jot down when/where/why you initiated this data capture.

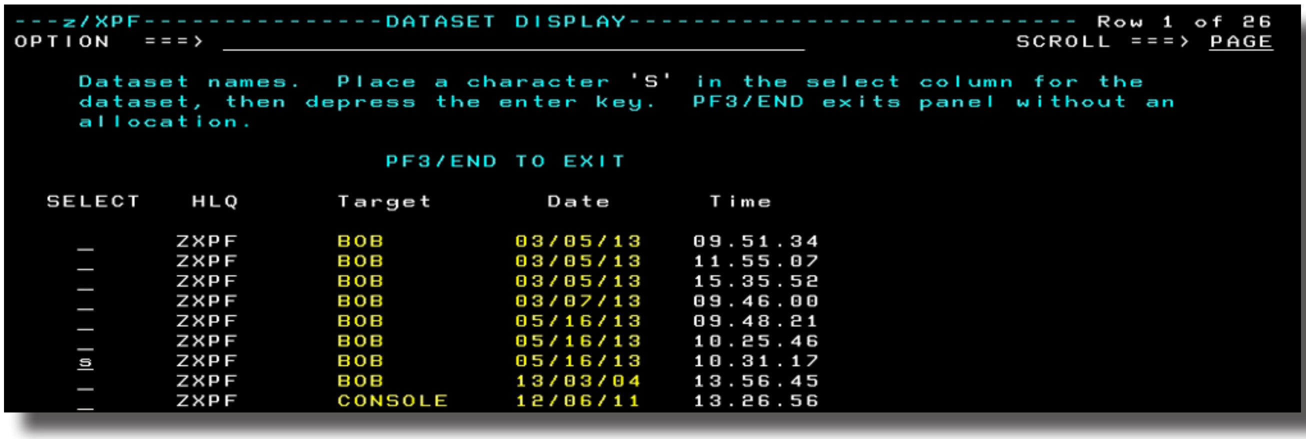
When you use the variables in the top portion of the panel shown in Figure 5-4, you will see the display entitled, "DATASET DISPLAY"   See Figure 5-9-1 below:



```
---z/XPF---------------DATASET DISPLAY--------------------------- Row 1 of 26
OPTION   ===>                                                  SCROLL ===> PAGE

    Dataset names.  Place a character 'S' in the select column for the
    dataset, then depress the enter key.  PF3/END exits panel without an
    allocation.

                         PF3/END TO EXIT

  SELECT    HLQ       Target        Date        Time

    _      ZXPF      BOB         03/05/13    09.51.34
    _      ZXPF      BOB         03/05/13    11.55.07
    _      ZXPF      BOB         03/05/13    15.35.52
    _      ZXPF      BOB         03/07/13    09.46.00
    _      ZXPF      BOB         05/16/13    09.48.21
    _      ZXPF      BOB         05/16/13    10.25.46
    _      ZXPF      BOB         05/16/13    10.31.17
    _      ZXPF      BOB         13/03/04    13.56.45
    _      ZXPF      CONSOLE     12/06/11    13.26.56
    _      ZXPF      FHCSRVER    08/29/12    14.17.57
    _      ZXPF      FRANK       05/09/13    13.12.59
    _      ZXPF      FRANK       12/06/11    13.24.39
    _      ZXPF      MDLSERVE    08/23/12    09.26.57
    _      ZXPF      MDLSERVE    08/23/12    10.17.31
    _      ZXPF      MDLSERVE    08/23/12    11.14.30
    _      ZXPF      MDLSERVE    08/23/12    11.15.20
    _      ZXPF      MDLSERVE    08/23/12    11.28.48
    _      ZXPF      MDLSERVE    08/23/12    11.28.57
    _      ZXPF      MDLSERVE    08/24/12    00.07.32
    _      ZXPF      MIKE5       08/24/12    00.07.45
    _      ZXPF      MIKE7       08/24/12    00.07.51
    _      ZXPF      XXXSRVER    05/09/13    13.13.07
    _      ZXPF      XXXSRVER    05/09/13    13.32.30
    _      ZXPF      XXXSRVER    05/09/13    14.04.55
    _      ZXPF      XXXSRVER    08/24/12    00.07.37
    _      ZXPF      ZXPFIVP     12/02/11    13.56.45
****************************** Bottom of data ********************************
```

Figure 5-9-1.

Use the Dataset Display panel to select a data set for reporting. Place an "S" next to the dataset you wish to create reports for, and press Enter. Figure 5-9-2 below shows that one dataset is going to be selected.

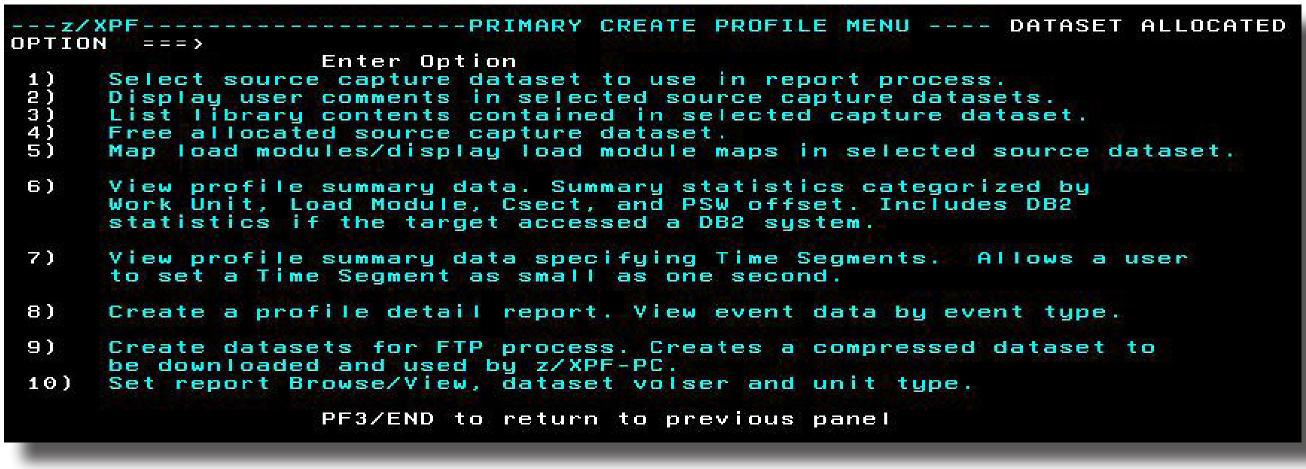


```
---z/XPF---------------DATASET DISPLAY--------------------------- Row 1 of 26
OPTION   ===>                                                  SCROLL ===> PAGE

    Dataset names.  Place a character 'S' in the select column for the
    dataset, then depress the enter key.  PF3/END exits panel without an
    allocation.

                         PF3/END TO EXIT

  SELECT    HLQ       Target        Date        Time

    _      ZXPF      BOB         03/05/13    09.51.34
    _      ZXPF      BOB         03/05/13    11.55.07
    _      ZXPF      BOB         03/05/13    15.35.52
    _      ZXPF      BOB         03/07/13    09.46.00
    _      ZXPF      BOB         05/16/13    09.48.21
    _      ZXPF      BOB         05/16/13    10.25.46
    s      ZXPF      BOB         05/16/13    10.31.17
    _      ZXPF      BOB         13/03/04    13.56.45
    _      ZXPF      CONSOLE     12/06/11    13.26.56
```

Figure 5-9-2.

Press PF3/END to exit this function, and complete the allocation of the dataset.

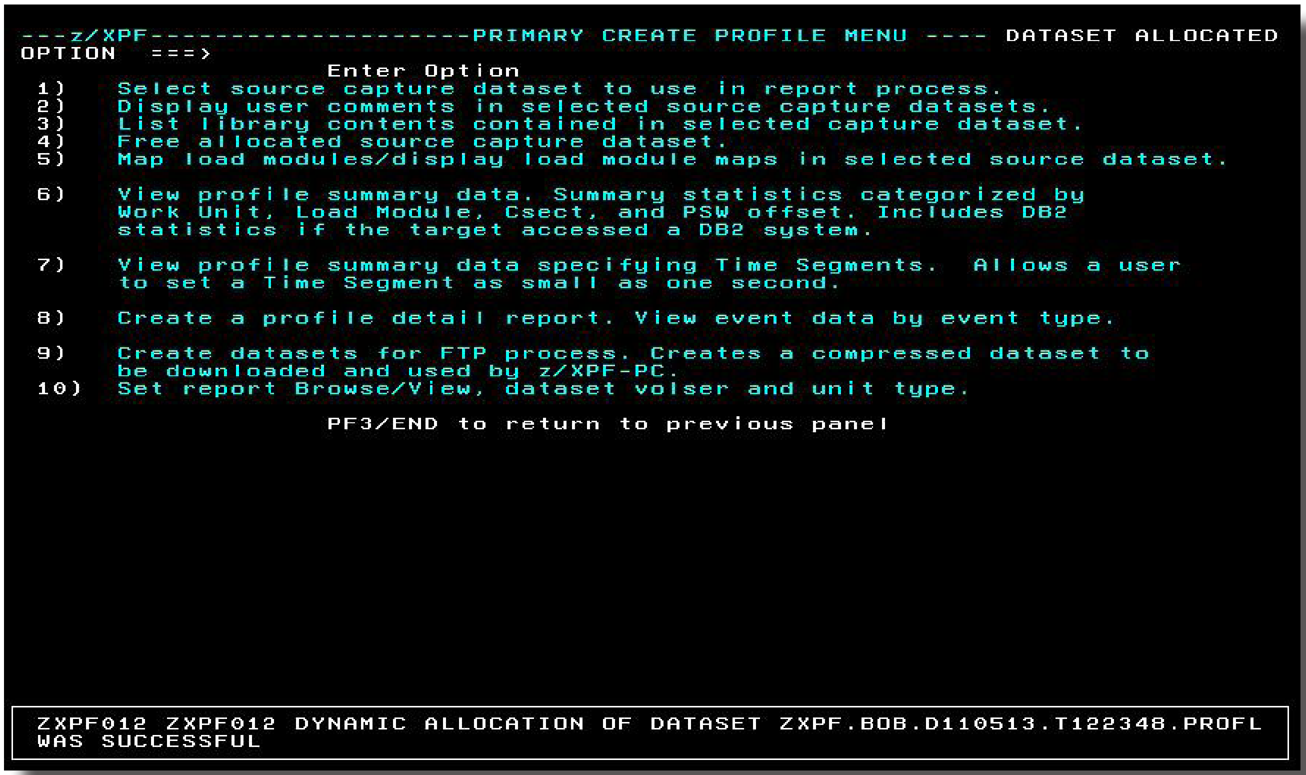
Figure 5-7 below shows the PRIMARY CREATE PROFILE MENU panel immediately after the allocation process completes successfully for the selected dataset. Note the "DATASET ALLOCATED" message in upper right portion of the panel :



```
---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 5-9-3.

If you want to see more of the text in any z/XPF message, you can press PF1 to reveal it, as below, in Figure 5-9-4:

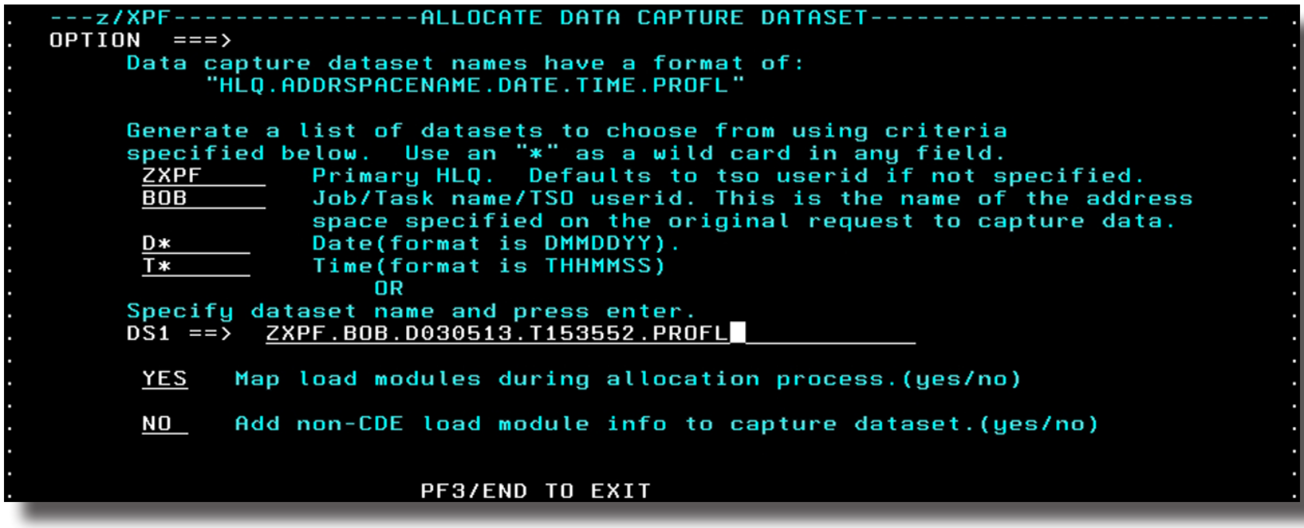


```
---z/XPF--------------------PRIMARY CREATE PROFILE MENU ---- DATASET ALLOCATED
OPTION  ===>
                       Enter Option
 1)   Select source capture dataset to use in report process.
 2)   Display user comments in selected source capture datasets.
 3)   List library contents contained in selected 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

 ZXPF012 ZXPF012 DYNAMIC ALLOCATION OF DATASET ZXPF.BOB.D110513.T122348.PROFL
 WAS SUCCESSFUL
```

Figure 5-9-4.

Figure 5-9-4 shows the effect of having pressed PF1, and the name of the allocated dataset has been displayed. If, for some reason, this was not the dataset you intended to select, proceed to Option 4 to free the allocation, and then re-do the allocation process. .

To allocate a source capture dataset without using the catalog search and list function, the DS1 variable must be filled in. Figure 5-9-5 shows the allocation panel with this variable filled in. When the enter key is depressed on this panel, and the DS1 variable is not blanks or spaces, the value in the variable is used to allocate the dataset, NOT the values in the top part of the panel.



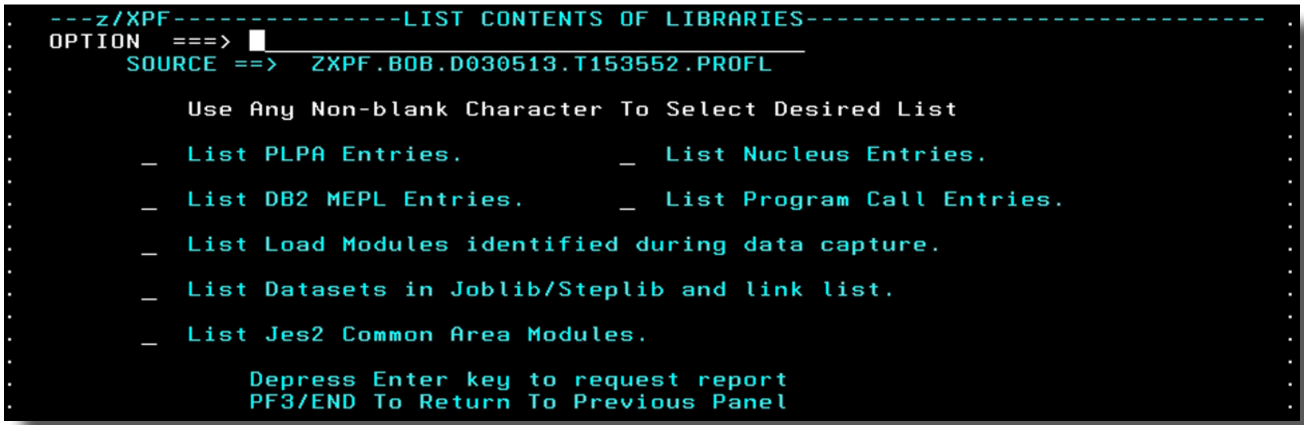
```
 ---z/XPF----------------ALLOCATE DATA CAPTURE DATASET--------------------------
 OPTION  ===>
      Data capture dataset names have a format of:
           "HLQ.ADDRSPACENAME.DATE.TIME.PROFL"

      Generate a list of datasets to choose from using criteria
      specified below.  Use an "*" as a wild card in any field.
        ZXPF         Primary HLQ.  Defaults to tso userid if not specified.
        BOB          Job/Task name/TSO userid. This is the name of the address
                     space specified on the original request to capture data.
        D*           Date(format is DMMDDYY).
        T*           Time(format is THHMMSS)
                             OR
      Specify dataset name and press enter.
      DS1  ==>   ZXPF.BOB.D030513.T153552.PROFL

        YES    Map load modules during allocation process.(yes/no)

        NO     Add non-CDE load module info to capture dataset.(yes/no)


                           PF3/END TO EXIT
```

Figure 5-9-5.

With the DS1 variable filled in, the catalog search and select logic is bypassed.

5-10 Option 3: Listing Library Contents

Option 3 is used to examine the contents of the z/OS environmental libraries that are used during Summary and Detail report processing to match an instruction address to a load module. See Figure 5-10-1 below:



```
 ---z/XPF---------------LIST CONTENTS OF LIBRARIES-----------------------------
 OPTION  ===>
      SOURCE ==>  ZXPF.BOB.D030513.T153552.PROFL

          Use Any Non-blank Character To Select Desired List

        _  List PLPA Entries.          _  List Nucleus Entries.

        _  List DB2 MEPL Entries.      _  List Program Call Entries.

        _  List Load Modules identified during data capture.

        _  List Datasets in Joblib/Steplib and link list.

        _  List Jes2 Common Area Modules.

              Depress Enter key to request report
              PF3/END To Return To Previous Panel
```

Figure 5-10-1.

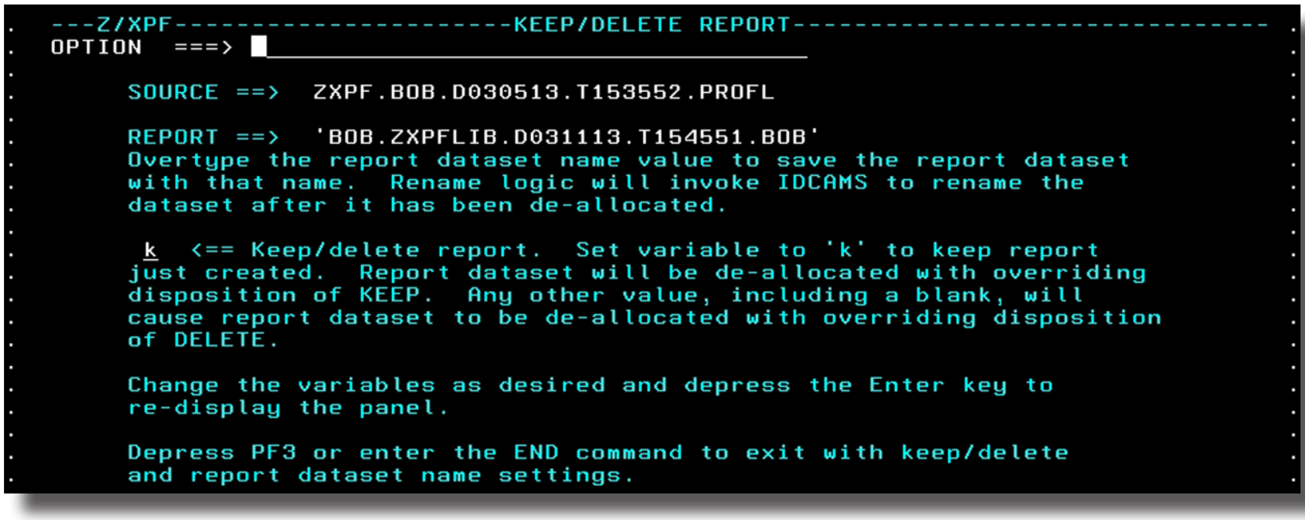
Here is where you can select items that you want included in the report.

[Of the choices above, “List PLPA Entries”, “List Nucleus Entries, “List DB2 MEPL Entries’ and “List JES2 Common Area Modules” will be the same for all data capture datasets created by the same instance of the z/XPF server address space. That’s because these items, when loaded at IPL time, don’t move. So you’ll get the same results for any address space.]

When z/XPF’s server address space initializes, lists are constructed for these entries from their respective sources. If DB2 systems are active on the z/OS image, each system is examined and a copy of the MEPL list is made. If the primary Job Entry Sub-system is JES2, the JES2 Common Area modules are identified, and a list is created.

z/XPF uses a table to keep track of the currently valid Program Call numbers the target application could use. At server initialization, a primary look-up table is built that contains one entry for each PC defined with a system LX plus one entry for each of PC owned by an active DB2 system. At that time, the server notes the location of the default system LX table that all address spaces use. This look-up table is the default table z/XPF uses to validate Program Call numbers. During data capture, z/XPF monitors the target profile application’s Program Call environment. When the application’s environment is altered so that it is no longer using the default system LX table to resolve PC numbers, a second table is built using the application’s current environment as input. This second table is then used by z/XPF to validate Program Call numbers.

All of the reports created using this function are directed to a report dataset. Depress the Enter key after making the selection(s). You’ll see your report. When you have finished with the report, z/XPF allows the chance to keep the report. You may alter the name by over-typing it, then put a “K” next to <== Keep/delete report. The file will be renamed, and saved for reuse. An example of this panel appears below, in Figure 5-10-2:



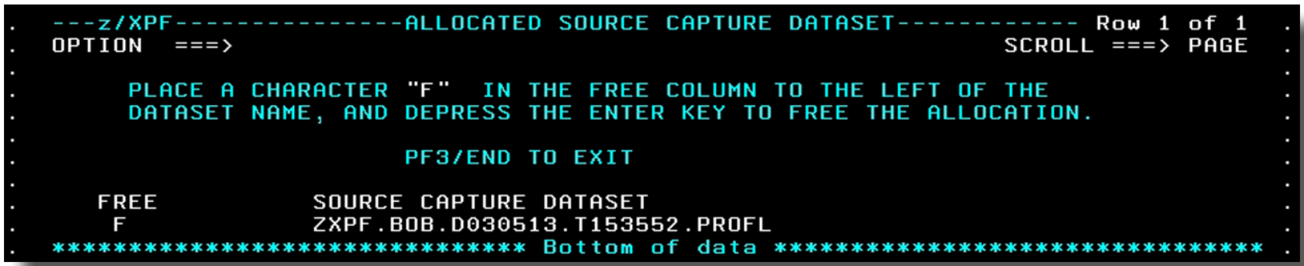
```
 ---Z/XPF---------------------KEEP/DELETE REPORT-------------------------------
 OPTION   ===> 

       SOURCE ==>  ZXPF.BOB.D030513.T153552.PROFL

       REPORT ==>  'BOB.ZXPFLIB.D031113.T154551.BOB'
       Overtype the report dataset name value to save the report dataset
       with that name.  Rename logic will invoke IDCAMS to rename the
       dataset after it has been de-allocated.

        k  <== Keep/delete report.  Set variable to 'k' to keep report
       just created.  Report dataset will be de-allocated with overriding
       disposition of KEEP.  Any other value, including a blank, will
       cause report dataset to be de-allocated with overriding disposition
       of DELETE.

       Change the variables as desired and depress the Enter key to
       re-display the panel.

       Depress PF3 or enter the END command to exit with keep/delete
       and report dataset name settings.
```

Figure 5-10-2.

5-11 Option 4: Freeing Allocated Source Capture Datasets

z/XPF can report on one data capture dataset at a time. So, when you're finished doing your reporting on a specific data capture dataset, you must free the existing dataset before allocating another. Alternatively, if you exit z/XPF's reporting structure, the dataset will be automatically freed.

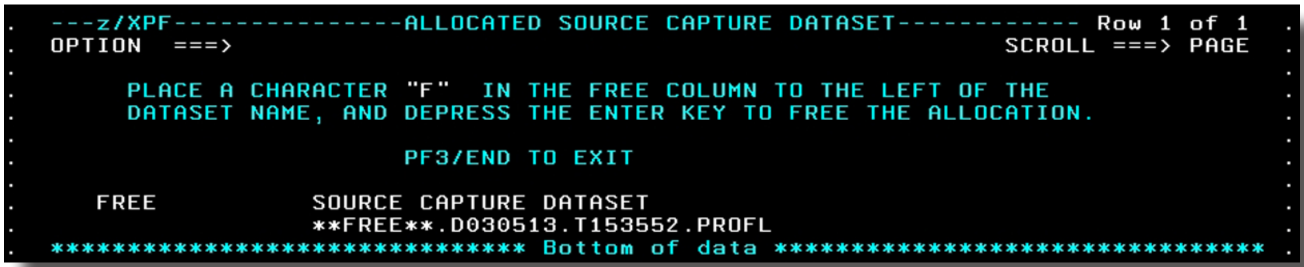
In the panel below, Place an "F" character next to the dataset you wish to free, and Press Enter.



```
.  ---z/XPF---------------ALLOCATED SOURCE CAPTURE DATASET------------ Row 1 of 1  .
.  OPTION  ===>                                                   SCROLL ===> PAGE  .
.                                                                                   .
.       PLACE A CHARACTER "F"  IN THE FREE COLUMN TO THE LEFT OF THE                .
.       DATASET NAME, AND DEPRESS THE ENTER KEY TO FREE THE ALLOCATION.             .
.                                                                                   .
.                         PF3/END TO EXIT                                           .
.                                                                                   .
.     FREE           SOURCE CAPTURE DATASET                                         .
.      F             ZXPF.BOB.D030513.T153552.PROFL                                 .
.  ******************************* Bottom of data ******************************** .
```

Figure 5-11-1.

After you press Enter you'll see the message "**FREE** appear in the dataset name, as in Figure :5-11-2 below:

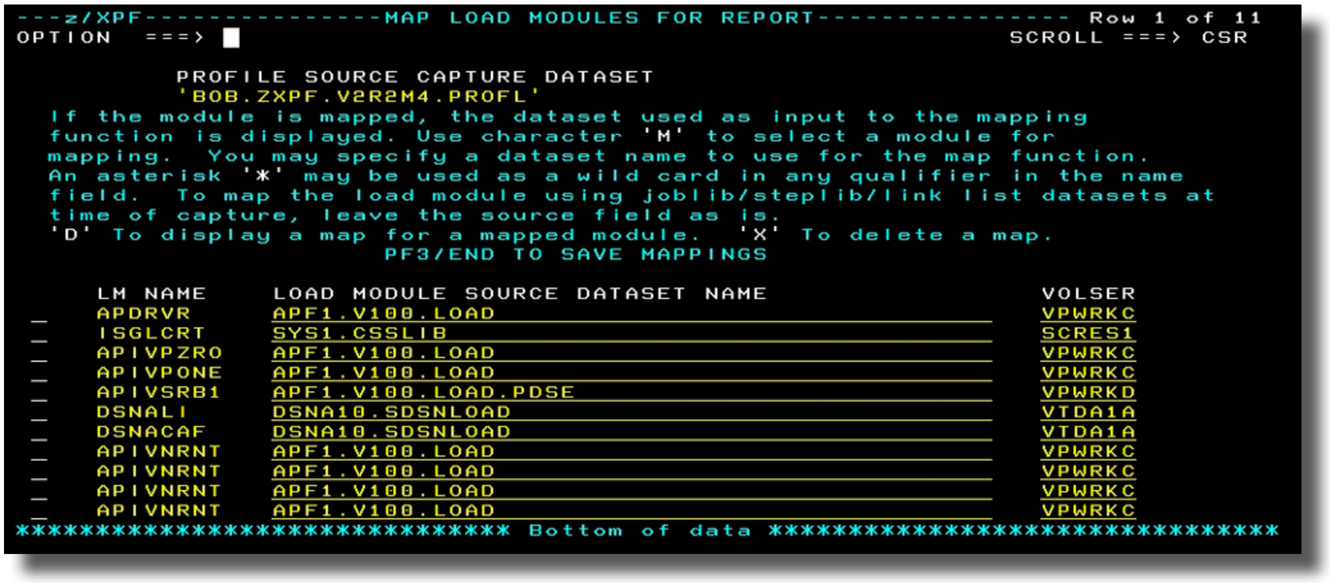


```
.  ---z/XPF---------------ALLOCATED SOURCE CAPTURE DATASET------------ Row 1 of 1  .
.  OPTION  ===>                                                   SCROLL ===> PAGE  .
.                                                                                   .
.       PLACE A CHARACTER "F"  IN THE FREE COLUMN TO THE LEFT OF THE                .
.       DATASET NAME, AND DEPRESS THE ENTER KEY TO FREE THE ALLOCATION.             .
.                                                                                   .
.                         PF3/END TO EXIT                                           .
.                                                                                   .
.     FREE           SOURCE CAPTURE DATASET                                         .
.                    **FREE**.D030513.T153552.PROFL                                 .
.  ******************************* Bottom of data ******************************** .
```

Figure 5-11-2.

5-12 Option 5: Map load modules/display load module maps

Use this function to map a load module that isn't mapped, to delete and re-map a load module, or to just delete the existing map for a load module. When you select Option 5 you will see a panel similar to Figure 5-12-1:



```
---z/XPF---------------MAP LOAD MODULES FOR REPORT---------------- Row 1 of 11
OPTION   ===>                                                SCROLL ===> CSR

          PROFILE SOURCE CAPTURE DATASET
          'BOB.ZXPF.V2R2M4.PROFL'
  If the module is mapped, the dataset used as input to the mapping
  function is displayed. Use character 'M' to select a module for
  mapping.  You may specify a dataset name to use for the map function.
  An asterisk '*' may be used as a wild card in any qualifier in the name
  field.  To map the load module using joblib/steplib/link list datasets at
  time of capture, leave the source field as is.
  'D' To display a map for a mapped module.  'X' To delete a map.
                        PF3/END TO SAVE MAPPINGS

     LM NAME    LOAD MODULE SOURCE DATASET NAME                 VOLSER
 _   APDRVR     APF1.V100.LOAD                                  VPWRKC
 _   ISGLCRT    SYS1.CSSLIB                                     SCRES1
 _   APIVPZRO   APF1.V100.LOAD                                  VPWRKC
 _   APIVPONE   APF1.V100.LOAD                                  VPWRKC
 _   APIVSRB1   APF1.V100.LOAD.PDSE                             VPWRKD
 _   DSNALI     DSNA10.SDSNLOAD                                 VTDA1A
 _   DSNACAF    DSNA10.SDSNLOAD                                 VTDA1A
 _   APIVNRNT   APF1.V100.LOAD                                  VPWRKC
 _   APIVNRNT   APF1.V100.LOAD                                  VPWRKC
 _   APIVNRNT   APF1.V100.LOAD                                  VPWRKC
 _   APIVNRNT   APF1.V100.LOAD                                  VPWRKC
******************************* Bottom of data ********************************
```

Figure 5-12-1.

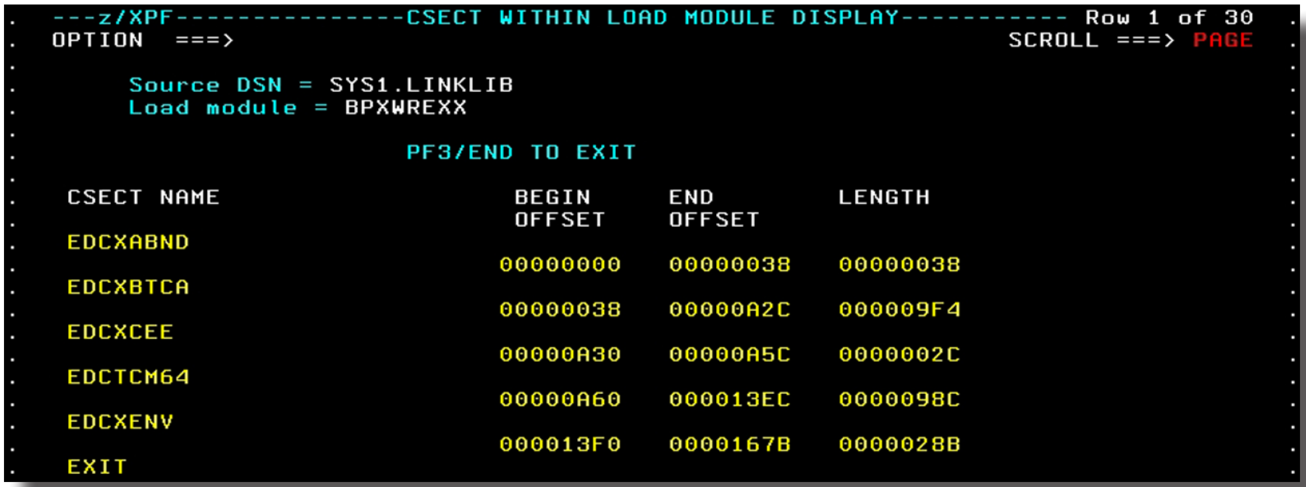
The field to the left of the load module name will accept one of three values. “M”, “D”, or “X”.

“M” will map a load module.
“D” will display a module map.
“X” will delete an existing map.

To map a Load Module, place a character “M” in the input variable, and press the Enter key. Joblib/Steplib and Linklist datasets will be searched for the load module, and when found, the Binder will be used to create the csect map for the load module.

If you need to map the load module using a library not in the Joblib/Steplib and Linklist datasets, then you may enter a dataset name in the variable to the right of the load module name. Overtype the “NOT MAPPED” characters with the dataset name, specified in TSO format. If the dataset you wish to map from does not have your TSO userid as the high-level-qualifier, you'll need to enclose the dataset name in single quotes. You may also have to specify the VOLSER to be searched for the map.

Let's display a csect map. In this example, I've put a "D" next to BMXWREXX (out of SYS1.LINKLIB) and have displayed the csect information within it. See Figure 5-12-2 below:

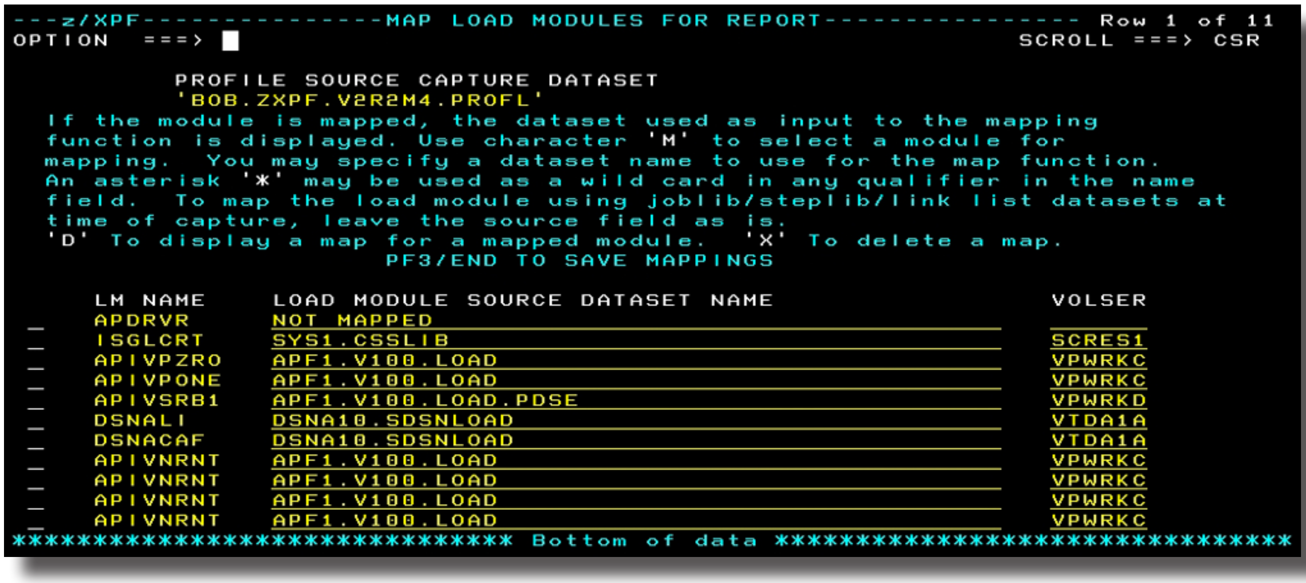


```
.   ---z/XPF---------------CSECT WITHIN LOAD MODULE DISPLAY----------- Row 1 of 30 .
.   OPTION  ===>                                                  SCROLL ===> PAGE .
.                                                                                  .
.        Source DSN = SYS1.LINKLIB                                                 .
.        Load module = BPXWREXX                                                    .
.                                                                                  .
.                                PF3/END TO EXIT                                   .
.                                                                                  .
.    CSECT NAME                         BEGIN       END         LENGTH             .
.                                       OFFSET      OFFSET                         .
.    EDCXABND                                                                      .
.                                      00000000    00000038    00000038            .
.    EDCXBTCA                                                                      .
.                                      00000038    00000A2C    000009F4            .
.    EDCXCEE                                                                       .
.                                      00000A30    00000A5C    0000002C            .
.    EDCTCM64                                                                      .
.                                      00000A60    000013EC    0000098C            .
.    EDCXENV                                                                       .
.                                      000013F0    0000167B    0000028B            .
.    EXIT                                                                          .
```

Figure 5-12-2.

I've only shown a part of the display that resulted, because the csect list for this module goes on for a while. Sharp-eyed readers will note the "vertical offset" between the name of a csect and its extent information. This is because z/XPF has to allow for csect names that approach 63 characters in length. So, while the display looks bit odd, there's a good reason for the design.

To delete an existing map, place a character "X" in the input variable to the left of the Load Module name, and depress the enter key. See Figure 5-12-3 below:



```
---z/XPF---------------MAP LOAD MODULES FOR REPORT--------------- Row 1 of 11
OPTION   ===> █                                                 SCROLL ===> CSR

          PROFILE SOURCE CAPTURE DATASET
          'BOB.ZXPF.V2R2M4.PROFL'
  If the module is mapped, the dataset used as input to the mapping
  function is displayed. Use character 'M' to select a module for
  mapping.  You may specify a dataset name to use for the map function.
  An asterisk '*' may be used as a wild card in any qualifier in the name
  field.  To map the load module using joblib/steplib/link list datasets at
  time of capture, leave the source field as is.
  'D' To display a map for a mapped module.   'X' To delete a map.
                              PF3/END TO SAVE MAPPINGS

     LM NAME     LOAD MODULE SOURCE DATASET NAME                  VOLSER
 _   APDRVR      NOT MAPPED
 _   ISGLCRT     SYS1.CSSLIB                                      SCRES1
 _   APIVPZRO    APF1.V100.LOAD                                   VPWRKC
 _   APIVPONE    APF1.V100.LOAD                                   VPWRKC
 _   APIVSRB1    APF1.V100.LOAD.PDSE                              VPWRKD
 _   DSNALI      DSNA10.SDSNLOAD                                  VTDA1A
 _   DSNACAF     DSNA10.SDSNLOAD                                  VTDA1A
 _   APIVNRNT    APF1.V100.LOAD                                   VPWRKC
 _   APIVNRNT    APF1.V100.LOAD                                   VPWRKC
 _   APIVNRNT    APF1.V100.LOAD                                   VPWRKC
 _   APIVNRNT    APF1.V100.LOAD                                   VPWRKC
****************************** Bottom of data ********************************
```

Figure 5-12-3.

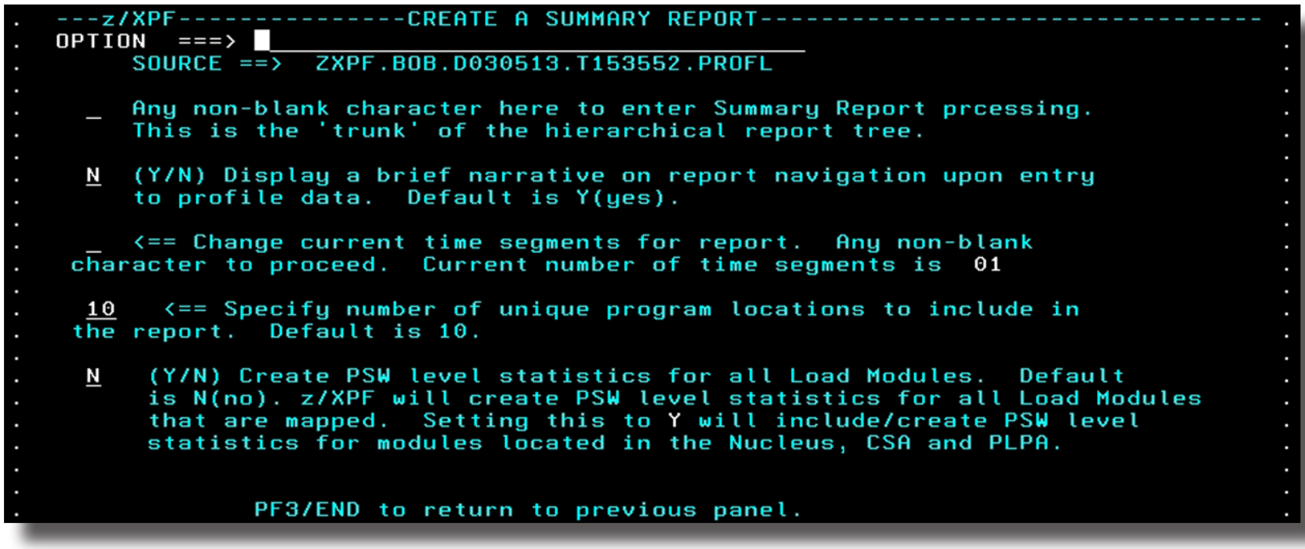
You can see that the status for Load Module APDRVR is now "NOT MAPPED".

5-13 Summary Reporting

5-14 Option 6: Create Profile Summary Reports - zXPF's Dynamic ISPF panels

[Tutorial digression: Because z/XPF captures each and every Trace Record for the span of time in which you do a data capture you can end up with HUGE VOLUMES of data. So, while other profilers may leave you wishing you had a bit more of a statistical sample, with z/XPF you'll wish you didn't have SO MUCH to look at. That's why it's important to understand how to limit the amount of information you have to look at with z/XPF. Some of this work is done in the next panel, and more is done transparently in z/XPF's dynamic ISPF reporting. Later on, we'll discuss Detail Reporting, and then using filters wisely will become CRUCIAL to you.]

Select Option 6 from the PRIMARY PROFILE CREATE MENU panel to begin Summary reporting. You'll get the panel shown in Figure 5-14-1 below:



```
---z/XPF---------------CREATE A SUMMARY REPORT---------------------------------
OPTION  ===> 
     SOURCE ==>  ZXPF.BOB.D030513.T153552.PROFL

  _  Any non-blank character here to enter Summary Report prcessing.
     This is the 'trunk' of the hierarchical report tree.

  N  (Y/N) Display a brief narrative on report navigation upon entry
     to profile data.  Default is Y(yes).

  _  <== Change current time segments for report.  Any non-blank
 character to proceed.  Current number of time segments is  01

  10   <== Specify number of unique program locations to include in
 the report.  Default is 10.

  N   (Y/N) Create PSW level statistics for all Load Modules.  Default
      is N(no). z/XPF will create PSW level statistics for all Load Modules
      that are mapped.  Setting this to Y will include/create PSW level
      statistics for modules located in the Nucleus, CSA and PLPA.


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

Figure 5-14-1.

From this panel you may make some initial choices before drilling down into your report.

To accept z/XPF's defaults, tab to the first field in the panel, insert an "S" and press Enter. That's all you really need to do.

The next field in this panel allows you to display a short tutorial on how to navigate in the reporting environment when the report has been processed and is ready for you to look at.

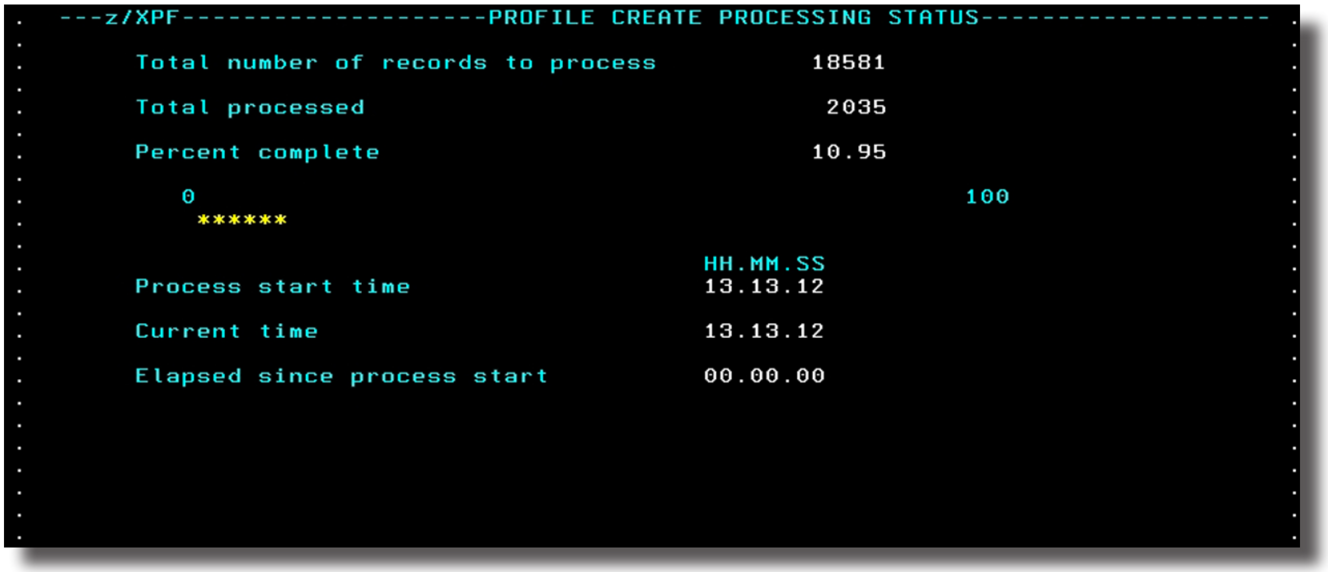
If you select "<=== Change current time segments for report., etc." a subsequent panel will appear that allows you to break the report into "time segments". You can specify up to ten evenly divided time segments. If you change this value, another panel will display and you may choose to include any time segments by putting an "I" next to them or exclude them by putting an "X" next to them. This allows you to isolate the time segment(s) that is/are most relevant to you, and ignore the rest.

> [Please note: In Summary Reporting *you now have a second opportunity to influence the number of Time Segments in your report.* That is coming up soon, under "Option 7" from the Summary Report Panel (when we get there).]

In the fourth field in Figure 5-14-1, you can alter the number of unique program locations from the default value of "10" to up to 99. You can consider this the "depth", or the "scope" of a report. The Unique Program Locations setting influences the scope by showing you only the most significant "nn" levels in any report. The default setting is "10", which means you'll see only the top ten Work Units, or load modules for any report. Since you're probably interested only in the "heavy hitters", the default of ten will be best for most situations.

In the fifth, or last field in this panel you can choose to have z/XPF create PSW statistics for load modules that are in system areas (the Nucleus, the CSA and PLPA) rather than just the load modules in the Private Area of the target address space. The default is "N", for "No". You can over-ride this if you need to investigate more deeply into operating system services, accepting the proviso that your report may be a good deal larger, and take more time to generate.

OK, we're done with the panel in Figure 5-14-1. Let's put a value in the first field and press Enter. You'll see an intermediate panel like the one below, in Figure 5-14-2:

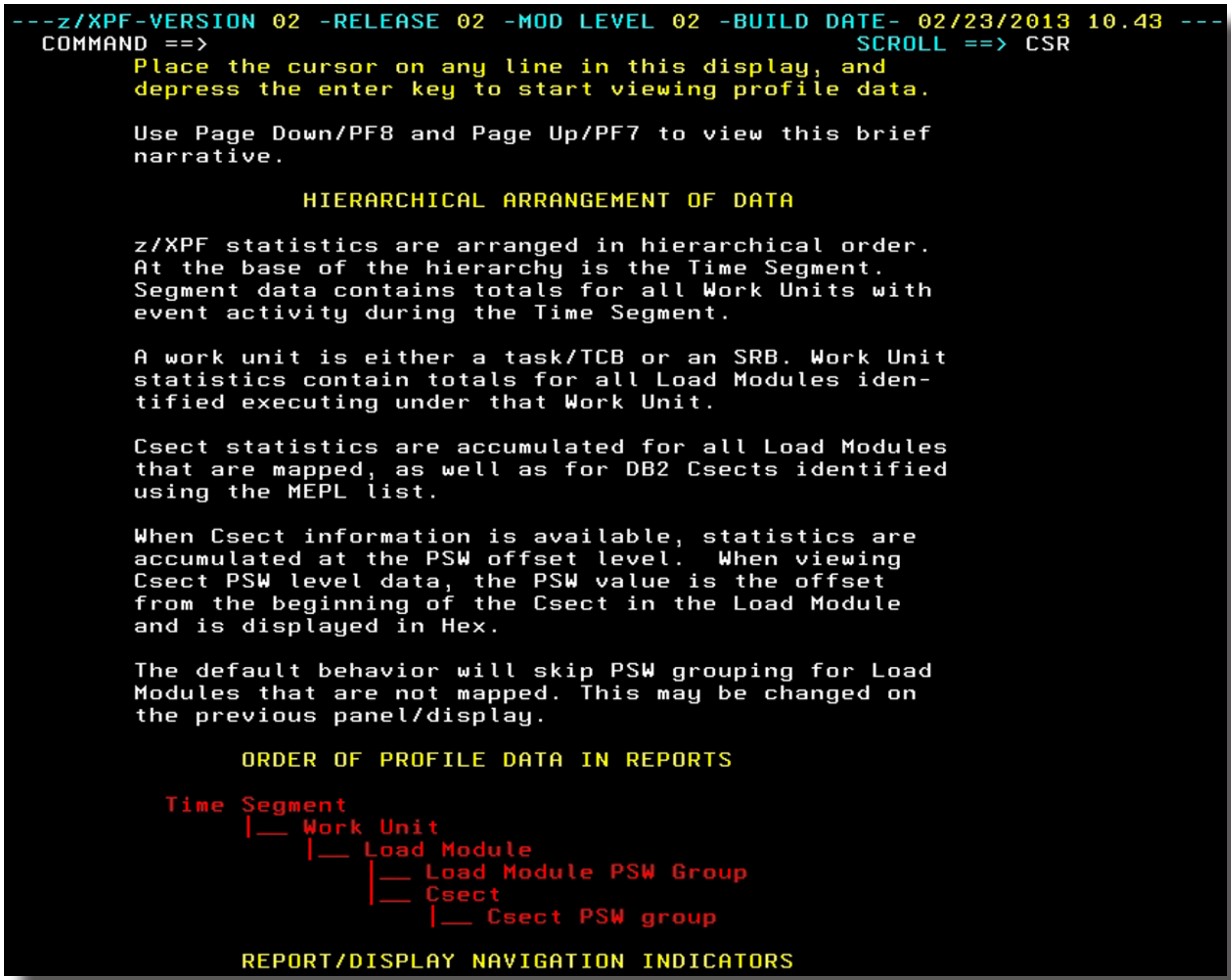


```
---z/XPF--------------------PROFILE CREATE PROCESSING STATUS-------------------

      Total number of records to process            18581

      Total processed                                2035

      Percent complete                              10.95

         0                                                        100
          ******

                                             HH.MM.SS
      Process start time                     13.13.12

      Current time                           13.13.12

      Elapsed since process start            00.00.00
```

Figure 5-14-2.

When you create a report for the first time z/XPF parses your data capture dataset into various categories. As it proceeds, the line of yellow asterisks will gradually extend across the page from left to right until the report is parsed. For very big reports, this could take some time. Find something else to do.

If you asked for a brief narrative on z/XPF's report navigation (the second field in Figure 5-18 above), you will see the, panel in Figure 5-14-3 below:



```
---z/XPF-VERSION 02 -RELEASE 02 -MOD LEVEL 02 -BUILD DATE- 02/23/2013 10.43 ---
  COMMAND ==>                                                  SCROLL ==> CSR
        Place the cursor on any line in this display, and
        depress the enter key to start viewing profile data.

        Use Page Down/PF8 and Page Up/PF7 to view this brief
        narrative.

                   HIERARCHICAL ARRANGEMENT OF DATA

        z/XPF statistics are arranged in hierarchical order.
        At the base of the hierarchy is the Time Segment.
        Segment data contains totals for all Work Units with
        event activity during the Time Segment.

        A work unit is either a task/TCB or an SRB. Work Unit
        statistics contain totals for all Load Modules iden-
        tified executing under that Work Unit.

        Csect statistics are accumulated for all Load Modules
        that are mapped, as well as for DB2 Csects identified
        using the MEPL list.

        When Csect information is available, statistics are
        accumulated at the PSW offset level.  When viewing
        Csect PSW level data, the PSW value is the offset
        from the beginning of the Csect in the Load Module
        and is displayed in Hex.

        The default behavior will skip PSW grouping for Load
        Modules that are not mapped. This may be changed on
        the previous panel/display.

               ORDER OF PROFILE DATA IN REPORTS

          Time Segment
               |__ Work Unit
                   |__ Load Module
                       |__ Load Module PSW Group
                       |__ Csect
                           |__ Csect PSW group

               REPORT/DISPLAY NAVIGATION INDICATORS
```

Figure 5-14-3.

This panel goes on for one more page-down operation, and it is worth your time to read, at least once.

With z/XPF's dynamic ISPF reporting you can quickly drill down through each level to see where your program is spending most of its resources, whether within your program itself or within a system service called by your program.

5-15 Navigating in Dynamic ISPF

Within each panel you can choose to select a report category and drill down. In some cases you can drill down and expand at the same time (depending on the context of your report).

As you proceed, z/XPF keeps track of your "place" in the yellow text at the top of each report panel. So, if you drill from Time Segment to the Work Unit Level to the Load Module level, z/XPF shows you that. If you drill "up", z/XPF's panel reacts to that action as well.

To "drill down within the same category" operation, position your cursor in any line that has this character string next to it: "Dril", then press Enter. You may also enter a "D" next to that report category, and that will perform a drill operation as well. The Enter key and the D character are interchangeable.

If you see the characters "DrEx", you have three choices. You can put the cursor in that input area and enter a "D" (for "drill") or press Enter (both actions perform a "Drill" operation) or you can enter an "X" to expand.

- **A Drill operation proceeds downward in the report hierarchy.**
- **An eXpand operation drills down and breaks out components laterally within the report hierarchy.**

Here's an example: If I wish to investigate Total Elapsed SVC Time, by Work Unit (see the Figure on the following page), I tab to the input field there and press Enter. Now, I'm looking at a panel that shows the total SVC time within the time segment, and the greatest consumers of SVC elapsed time are shown, sorted from highest consumer to lowest. Here, I have a choice to either drill down (by pressing Enter or using a "D" character") OR I can "X" for expand.

- If I drill here, I'll see the SVC elapsed time for the next level down (in this case I'll have gone from the Time Segment to the Work Unit level). I'll see which Work Unit used the most SVC elapsed time.
- If I expand here, I'll see elapsed SVC time at the Work Unit level, but now the report drills to the Work Unit level AND breaks the statistics out by the actual SVC numbers for that Work Unit.

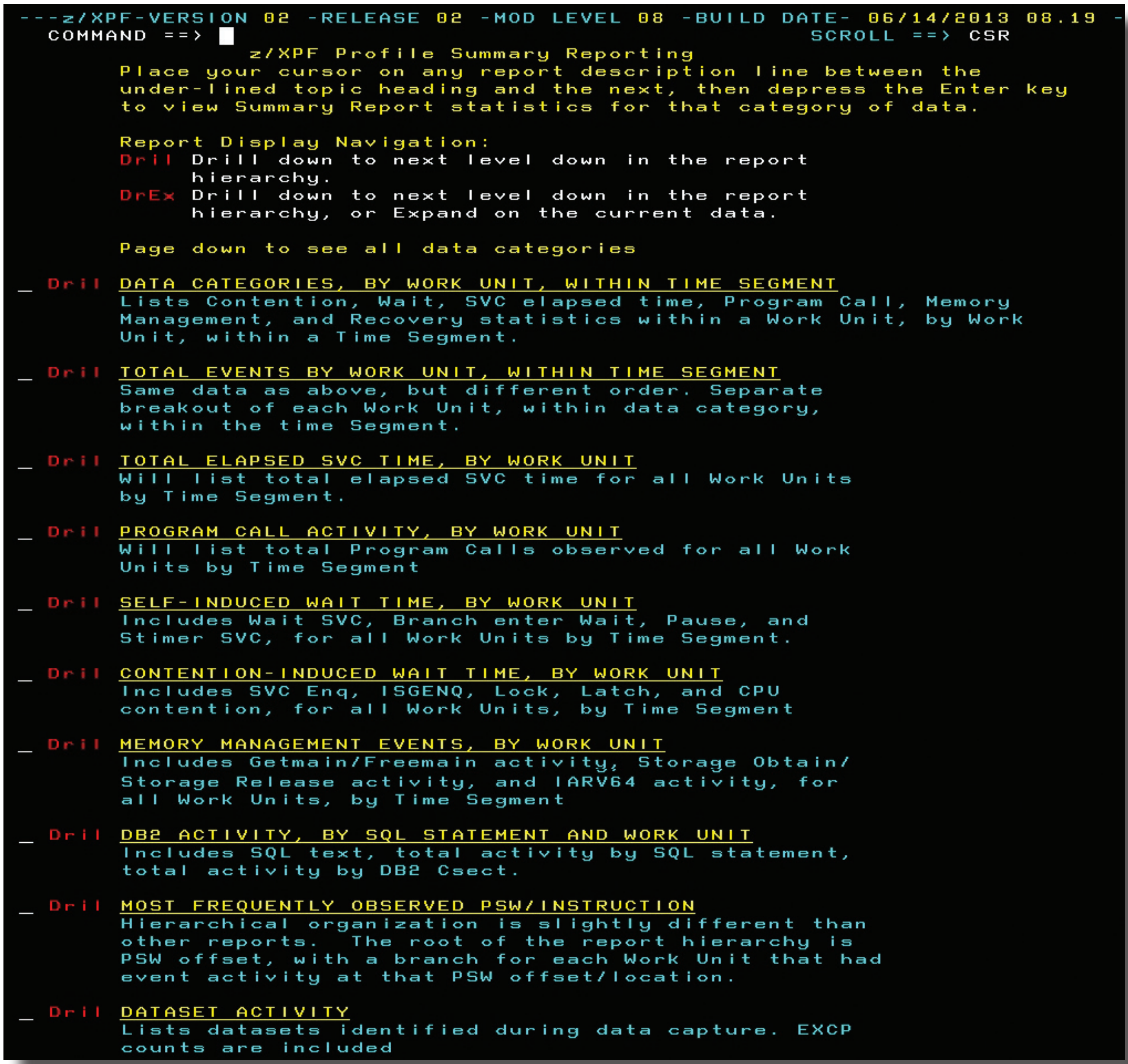
[Anecdote: The "Dril" and "DrEx" keywords take some getting used to. We went over and over 3270 screen attributes and character string combinations searching for a way to make these actions look more intuitive and less wierd. Do YOU have a better suggestion for these actions? We're listening...]

z/XPF's FIND Command

When you are in z/XPF's report panels, ISPF's FIND and RFIND commands are available to you. These are especially handy when you're looking at very large reports and need to locate something quickly. These commands are case-sensitive, so it's important to type precisely what you're looking for. The commands work as you'd expect them to.

Let's get to the main report panel.

Once you have scanned the educational panels, you can put your cursor anywhere on the page and press Enter. Now, you see the top panel of the report, as in Figure 5-15-1 below. I've artificially created the entire panel so you can see all that is available here:



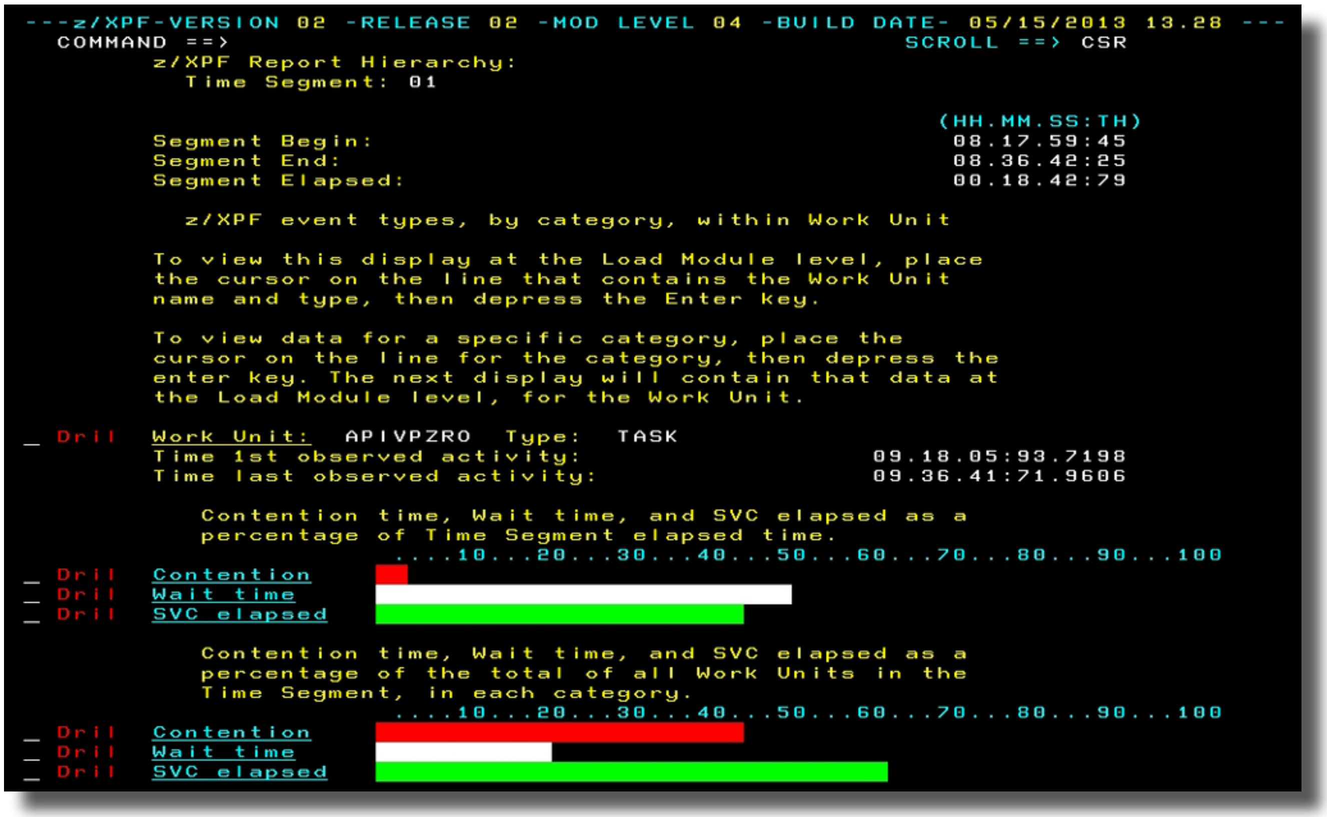
```
---z/XPF-VERSION 02 -RELEASE 02 -MOD LEVEL 08 -BUILD DATE- 06/14/2013 08.19 -
  COMMAND ==>                                                  SCROLL ==> CSR
                      z/XPF Profile Summary Reporting
           Place your cursor on any report description line between the
           under-lined 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  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  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  DATASET ACTIVITY
           Lists datasets identified during data capture. EXCP
           counts are included
```

Figure 5-15-1.

These categories are reasonably self-explanatory. Let's try a few.

5-16 Data Categories, by Work Unit, Within Time Segment

Tabbing down next to the first sub-topic in Figure 5-15-1 and pressing enter performs a "drill-down" into "Data Categories, by Work Unit, Within Time Segment". See Figure 5-16-1 below:



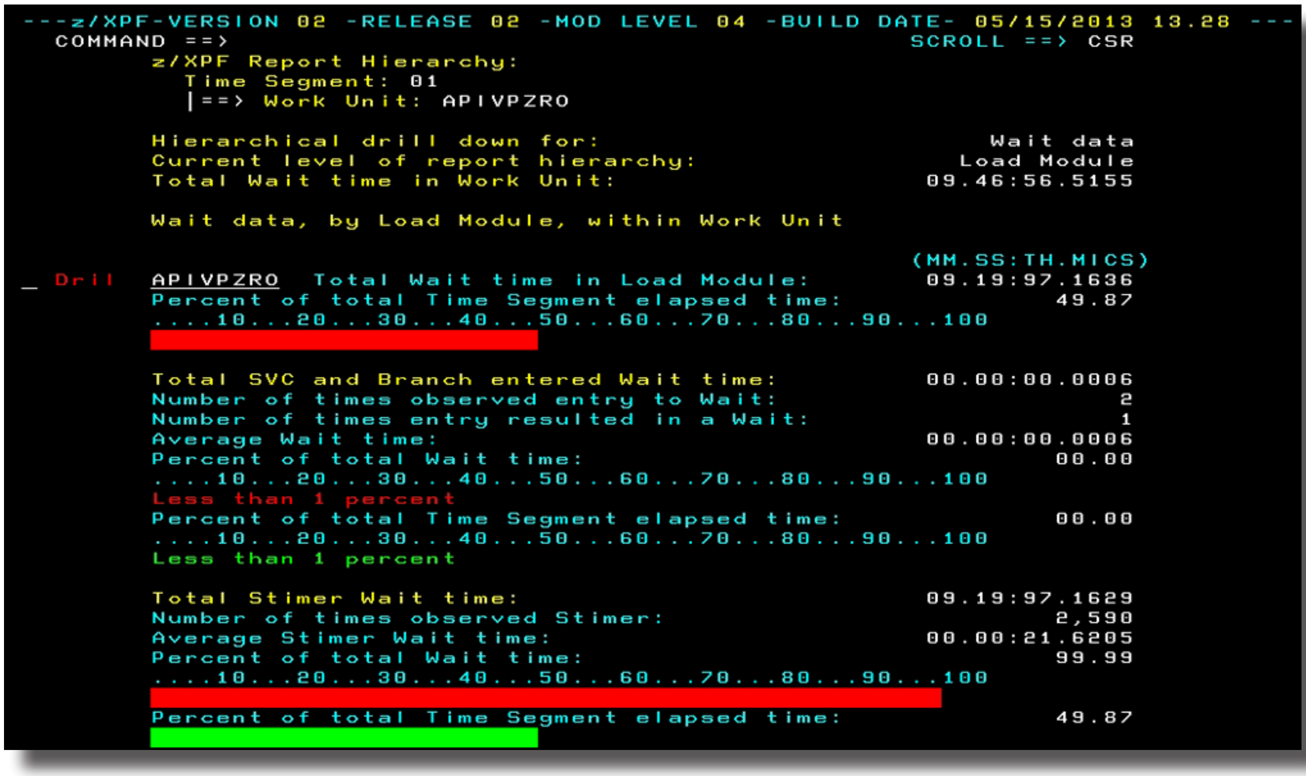
```
---z/XPF-VERSION 02 -RELEASE 02 -MOD LEVEL 04 -BUILD DATE- 05/15/2013 13.28 ---
  COMMAND ==>                                                  SCROLL ==> CSR
        z/XPF Report Hierarchy:
          Time Segment: 01

                                                                (HH.MM.SS:TH)
        Segment Begin:                                           08.17.59:45
        Segment End:                                             08.36.42:25
        Segment Elapsed:                                         00.18.42:79

          z/XPF event types, by category, within Work Unit

        To view this display at the Load Module level, place
        the cursor on the line that contains the Work Unit
        name and type, then depress the Enter key.

        To view data for a specific category, place the
        cursor on the line for the category, then depress the
        enter key. The next display will contain that data at
        the Load Module level, for the Work Unit.

_ Drill Work Unit:   APIVPZRO   Type:   TASK
        Time 1st observed activity:                         09.18.05:93.7198
        Time last observed activity:                        09.36.41:71.9606

           Contention time, Wait time, and SVC elapsed as a
           percentage of Time Segment elapsed time.
                       ....10...20...30...40...50...60...70...80...90...100
_ Drill Contention
_ Drill Wait time
_ Drill SVC elapsed

           Contention time, Wait time, and SVC elapsed as a
           percentage of the total of all Work Units in the
           Time Segment, in each category.
                       ....10...20...30...40...50...60...70...80...90...100
_ Drill Contention
_ Drill Wait time
_ Drill SVC elapsed
```

Figure 5-16-1.

z/XPF always tries to orient you by showing you the top paragraph of text (in yellow). There is only a single Time Segment for the report. Begin, end and elapsed times are given. Then z/XPF tells you what you're looking at. The greatest consumer of resources (Work Unit APIVPZRO) is sorted to the top of the report, and you can see three sub-categories: Contention, Wait Time and SVC Elapsed (time).

Clearly, Work Unit APIVPZRO is spending about 50% of its time in a Wait state. Let's drill down. I'll put my cursor next to the Wait Time line and press Enter. I see Figure 5-16-2, on the next page:

[The next few pages in this book are going to look a little wierd because of the size of each display. I'm making the assumption that you're reading "glass" here. If you decide to print this book then I suppose we'll kill a few more trees...]

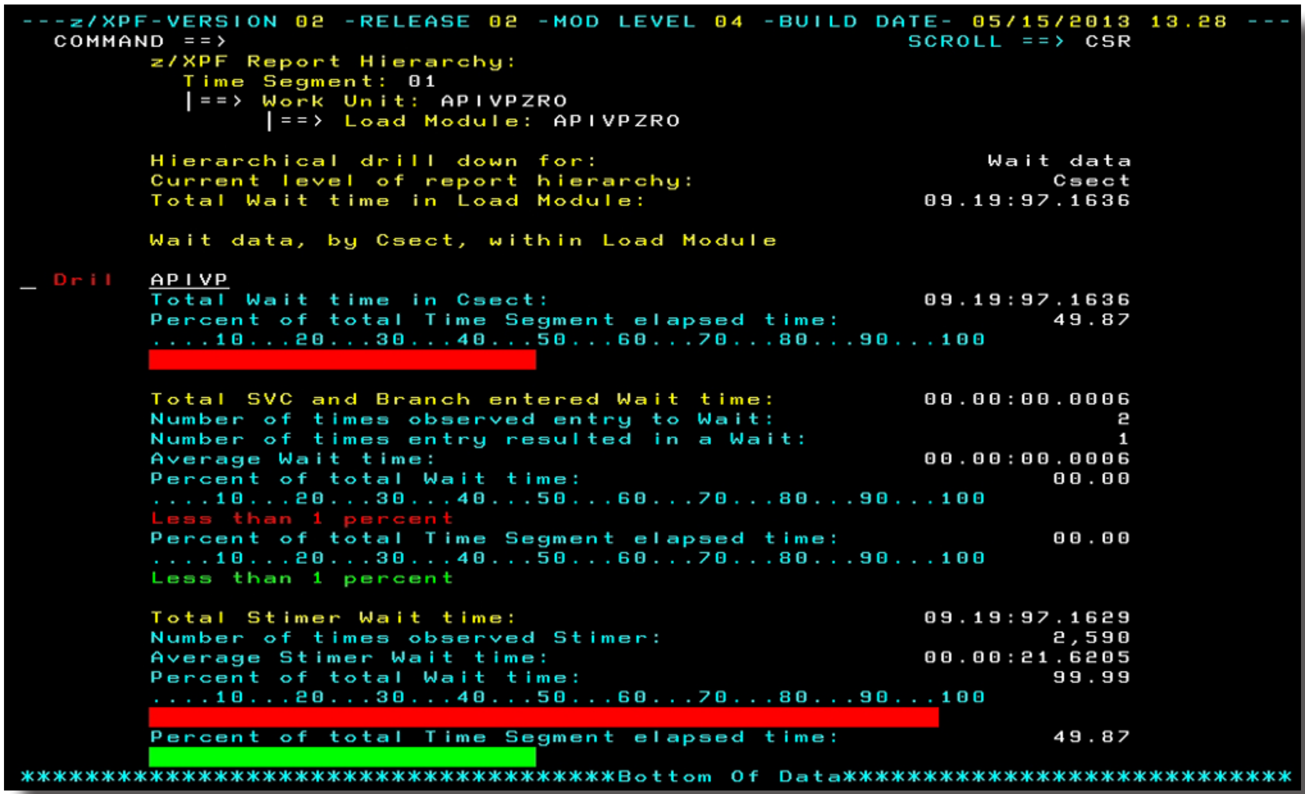


```
---z/XPF-VERSION 02 -RELEASE 02 -MOD LEVEL 04 -BUILD DATE- 05/15/2013 13.28 ---
 COMMAND ==>                                                SCROLL ==> CSR
       z/XPF Report Hierarchy:
         Time Segment: 01
         |==> Work Unit: APIVPZRO

       Hierarchical drill down for:                                Wait data
       Current level of report hierarchy:                        Load Module
       Total Wait time in Work Unit:                            09.46:56.5155

       Wait data, by Load Module, within Work Unit

                                                            (MM.SS:TH.MICS)
_ Dril APIVPZRO  Total Wait time in Load Module:            09.19:97.1636
       Percent of total Time Segment elapsed time:                   49.87
       ....10...20...30...40...50...60...70...80...90...100


       Total SVC and Branch entered Wait time:              00.00:00.0006
       Number of times observed entry to Wait:                          2
       Number of times entry resulted in a Wait:                        1
       Average Wait time:                                   00.00:00.0006
       Percent of total Wait time:                                  00.00
       ....10...20...30...40...50...60...70...80...90...100
       Less than 1 percent
       Percent of total Time Segment elapsed time:                  00.00
       ....10...20...30...40...50...60...70...80...90...100
       Less than 1 percent

       Total Stimer Wait time:                              09.19:97.1629
       Number of times observed Stimer:                             2,590
       Average Stimer Wait time:                            00.00:21.6205
       Percent of total Wait time:                                  99.99
       ....10...20...30...40...50...60...70...80...90...100

       Percent of total Time Segment elapsed time:                  49.87
```

Figure 5-16-2.

Now I can see that within the Work Unit APIVPZRO, load module APIVPZRO accounted for almost 50% of Wait time. SVC and branch-entered Wait time was less than 1% of this aggregate, and most of the Wait time is due to STIMER Wait. We'll drill down again into load module APIVPZRO to isolate Wait time at the csect level.

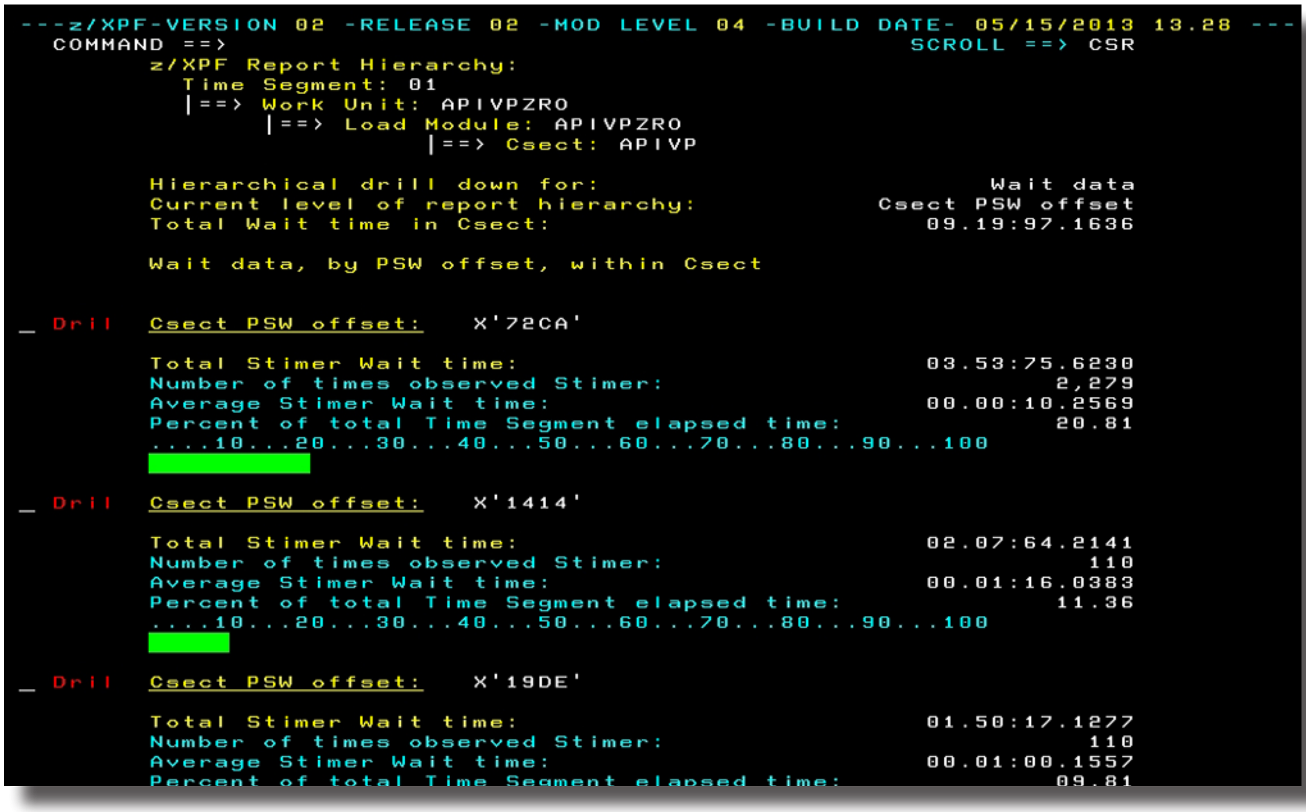
See Figure 5-16-3 below:



```
 ---z/XPF-VERSION 02 -RELEASE 02 -MOD LEVEL 04 -BUILD DATE- 05/15/2013 13.28 ---
   COMMAND ==>                                                 SCROLL ==> CSR
         z/XPF Report Hierarchy:
           Time Segment: 01
           |==> Work Unit: APIVPZRO
                |==> Load Module: APIVPZRO

         Hierarchical drill down for:                              Wait data
         Current level of report hierarchy:                            Csect
         Total Wait time in Load Module:                       09.19:97.1636

         Wait data, by Csect, within Load Module

 _ Dril  APIVP
         Total Wait time in Csect:                             09.19:97.1636
         Percent of total Time Segment elapsed time:                   49.87
         ....10...20...30...40...50...60...70...80...90...100


         Total SVC and Branch entered Wait time:               00.00:00.0006
         Number of times observed entry to Wait:                           2
         Number of times entry resulted in a Wait:                         1
         Average Wait time:                                    00.00:00.0006
         Percent of total Wait time:                                   00.00
         ....10...20...30...40...50...60...70...80...90...100
         Less than 1 percent
         Percent of total Time Segment elapsed time:                   00.00
         ....10...20...30...40...50...60...70...80...90...100
         Less than 1 percent

         Total Stimer Wait time:                               09.19:97.1629
         Number of times observed Stimer:                              2,590
         Average Stimer Wait time:                             00.00:21.6205
         Percent of total Wait time:                                   99.99
         ....10...20...30...40...50...60...70...80...90...100

         Percent of total Time Segment elapsed time:                   49.87

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

Figure 5-16-3.

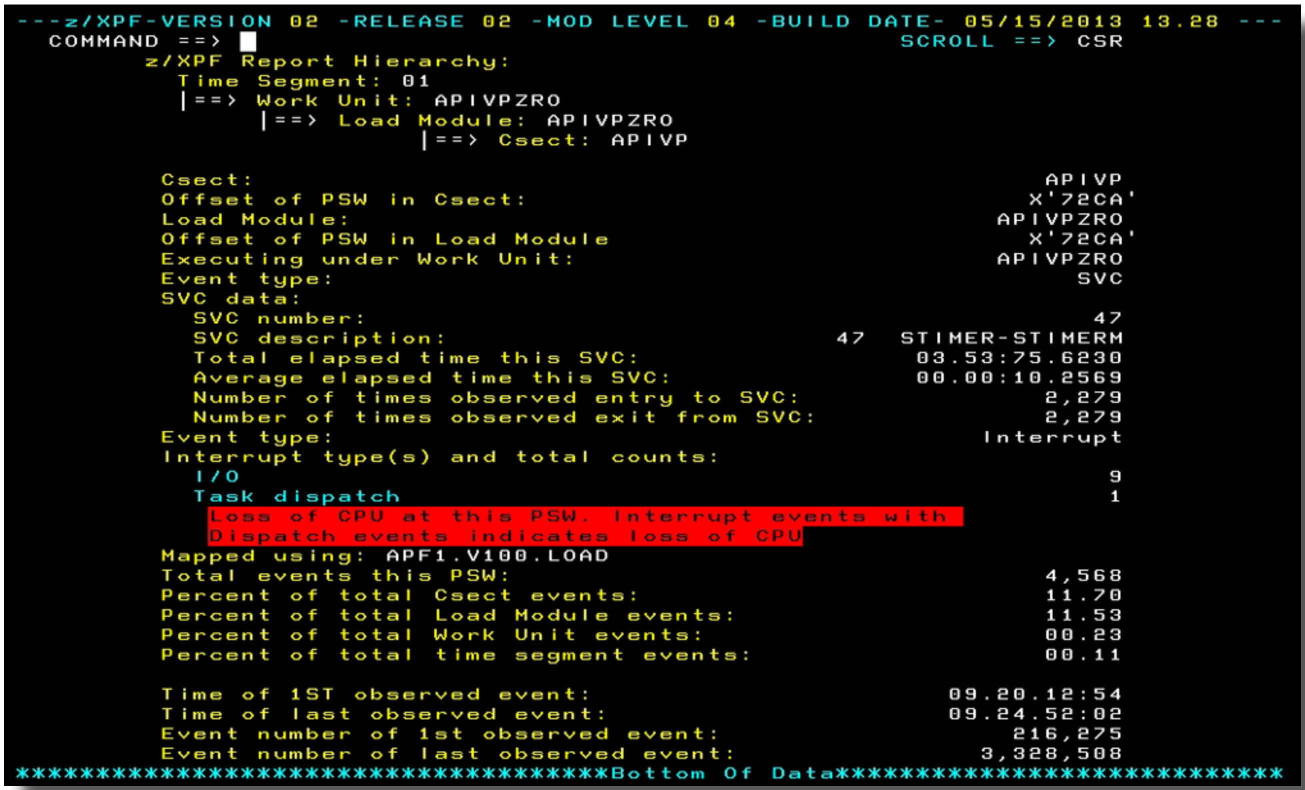
It appears that csect APIVP is doing most of the waiting, with 49.87% of the entire Time Segment. We can drill down again. See Figure 5-15-5 below:



```
---z/XPF-VERSION 02 -RELEASE 02 -MOD LEVEL 04 -BUILD DATE- 05/15/2013 13.28 ---
   COMMAND ==>                                                  SCROLL ==> CSR
         z/XPF Report Hierarchy:
           Time Segment: 01
           |==> Work Unit: APIVPZRO
                |==> Load Module: APIVPZRO
                          |==> Csect: APIVP

         Hierarchical drill down for:                              Wait data
         Current level of report hierarchy:                 Csect PSW offset
         Total Wait time in Csect:                             09.19:97.1636

         Wait data, by PSW offset, within Csect


_  Dril  Csect PSW offset:   X'72CA'

         Total Stimer Wait time:                               03.53:75.6230
         Number of times observed Stimer:                              2,279
         Average Stimer Wait time:                             00.00:10.2569
         Percent of total Time Segment elapsed time:                   20.81
         ....10...20...30...40...50...60...70...80...90...100


_  Dril  Csect PSW offset:   X'1414'

         Total Stimer Wait time:                               02.07:64.2141
         Number of times observed Stimer:                                110
         Average Stimer Wait time:                             00.01:16.0383
         Percent of total Time Segment elapsed time:                   11.36
         ....10...20...30...40...50...60...70...80...90...100


_  Dril  Csect PSW offset:   X'19DE'

         Total Stimer Wait time:                               01.50:17.1277
         Number of times observed Stimer:                                110
         Average Stimer Wait time:                             00.01:00.1557
         Percent of total Time Segment elapsed time:                   09.81
```

Figure 5-16-4.

It looks like the “culprit” here is at offset X’72CA’. Let’s drill again, shall we?
We do the drill operation and get this panel. See Figure 5-16-5 below:



```
---z/XPF-VERSION 02 -RELEASE 02 -MOD LEVEL 04 -BUILD DATE- 05/15/2013 13.28 ---
  COMMAND ==>                                              SCROLL ==> CSR
        z/XPF Report Hierarchy:
          Time Segment: 01
          |==> Work Unit: APIVPZRO
               |==> Load Module: APIVPZRO
                         |==> Csect: APIVP

         Csect:                                                       APIVP
         Offset of PSW in Csect:                                     X'72CA'
         Load Module:                                              APIVPZRO
         Offset of PSW in Load Module                                X'72CA'
         Executing under Work Unit:                                APIVPZRO
         Event type:                                                    SVC
         SVC data:
           SVC number:                                                   47
           SVC description:                            47   STIMER-STIMERM
           Total elapsed time this SVC:                      03.53:75.6230
           Average elapsed time this SVC:                    00.00:10.2569
           Number of times observed entry to SVC:                    2,279
           Number of times observed exit from SVC:                   2,279
         Event type:                                             Interrupt
         Interrupt type(s) and total counts:
           I/O                                                           9
           Task dispatch                                                 1
            Loss of CPU at this PSW.  Interrupt events with
            Dispatch events indicates loss of CPU
         Mapped using: APF1.V100.LOAD
         Total events this PSW:                                      4,568
         Percent of total Csect events:                              11.70
         Percent of total Load Module events:                        11.53
         Percent of total Work Unit events:                          00.23
         Percent of total time segment events:                       00.11

         Time of 1ST observed event:                           09.20.12:54
         Time of last observed event:                          09.24.52:02
         Event number of 1st observed event:                       216,275
         Event number of last observed event:                    3,328,508
*************************************Bottom Of Data*****************************
```

Figure 5-16-5.

Having drilled to the “bottom” level, we see that within the single Time Segment that makes up this report, within the Work Unit APIVPZRO, within the load module APIVPZRO within the csect APIVP at offset X’72CA’ the program is issuing a STIMER SVC (SVC 47). We’ve arrived at this information quickly, just by tabbing and pressing Enter at specific places.

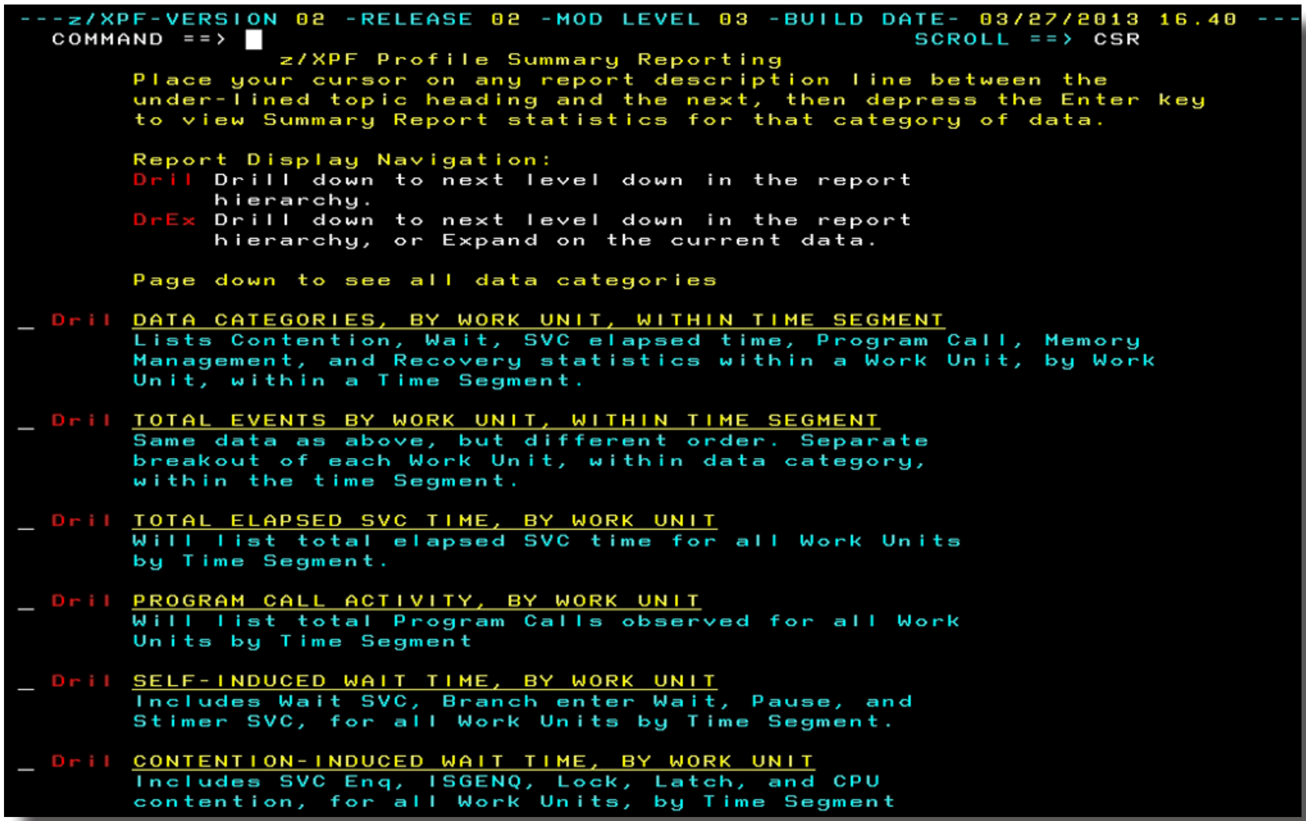
You can “back out” of any level of the report by pressing PF3.

> [Worth repeating: You can FIND whatever you need to in a report by typing “FIND”, and then typing the thing you want to see (case-sensitive). This is useful for quickly orienting the report to what you’re looking for (a load module name, a csect, etc.). The “RFIND” command is also supported.]

That was just one example of using z/XPF’s new dynamic ISPF reporting.

Let’s back out, and try a different approach. My last drilling sequence was into the “Data Categories, by Work Unit, Within Time Segment. In that enquiry, I quickly drilled down into APIVPZRO’s Wait Time report path. This time, I’m going to drill down to the PSW level in by Work Unit, to load module, to csect and finally to the PSW offset.

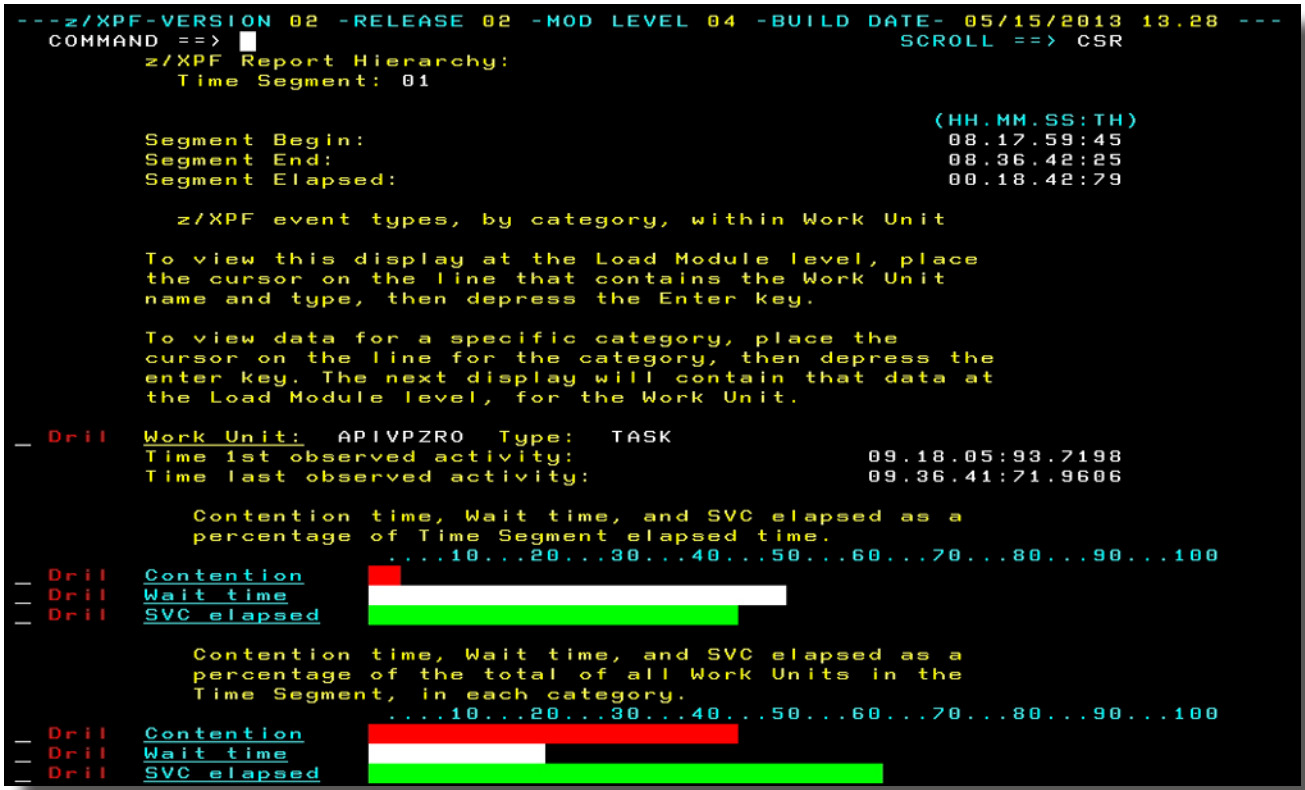
Here’s the first screen-full of z/XPF’s Profile Summary Reporting panel in Figure 5-16-8 below:



```
---z/XPF-VERSION 02 -RELEASE 02 -MOD LEVEL 03 -BUILD DATE- 03/27/2013 16.40 ---
   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 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
```

Figure 5-16-8.

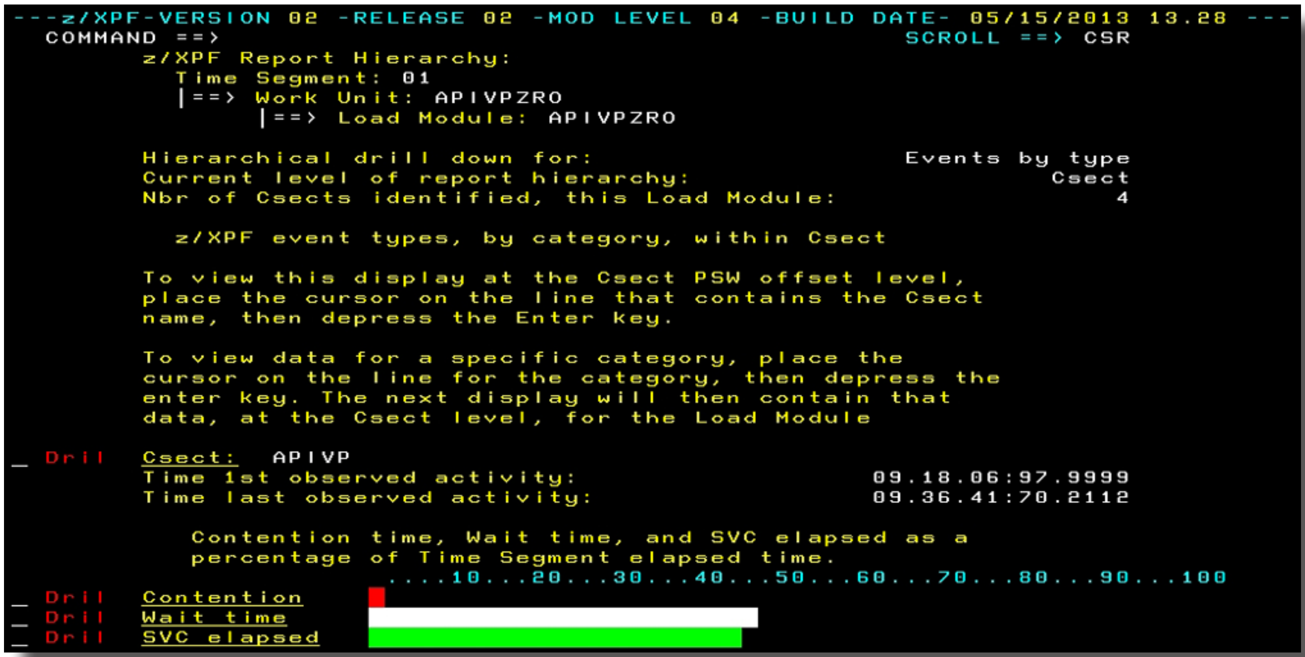
Again, I drill down into Data Categories, by Work Unit, Within Time Segment, and I see Figure 5-16-9 below:



```
---z/XPF-VERSION 02 -RELEASE 02 -MOD LEVEL 04 -BUILD DATE- 05/15/2013 13.28 ---
   COMMAND ==>                                                   SCROLL ==> CSR
         z/XPF Report Hierarchy:
           Time Segment: 01

                                                                  (HH.MM.SS:TH)
         Segment Begin:                                            08.17.59:45
         Segment End:                                              08.36.42:25
         Segment Elapsed:                                          00.18.42:79

           z/XPF event types, by category, within Work Unit

         To view this display at the Load Module level, place
         the cursor on the line that contains the Work Unit
         name and type, then depress the Enter key.

         To view data for a specific category, place the
         cursor on the line for the category, then depress the
         enter key. The next display will contain that data at
         the Load Module level, for the Work Unit.

_  Drill Work Unit:  APIVPZRO  Type:  TASK
         Time 1st observed activity:                          09.18.05:93.7198
         Time last observed activity:                         09.36.41:71.9606

            Contention time, Wait time, and SVC elapsed as a
            percentage of Time Segment elapsed time.
                        ....10...20...30...40...50...60...70...80...90...100
_  Drill Contention
_  Drill Wait time
_  Drill SVC elapsed

            Contention time, Wait time, and SVC elapsed as a
            percentage of the total of all Work Units in the
            Time Segment, in each category.
                        ....10...20...30...40...50...60...70...80...90...100
_  Drill Contention
_  Drill Wait time
_  Drill SVC elapsed
```

Figure 5-16-9.

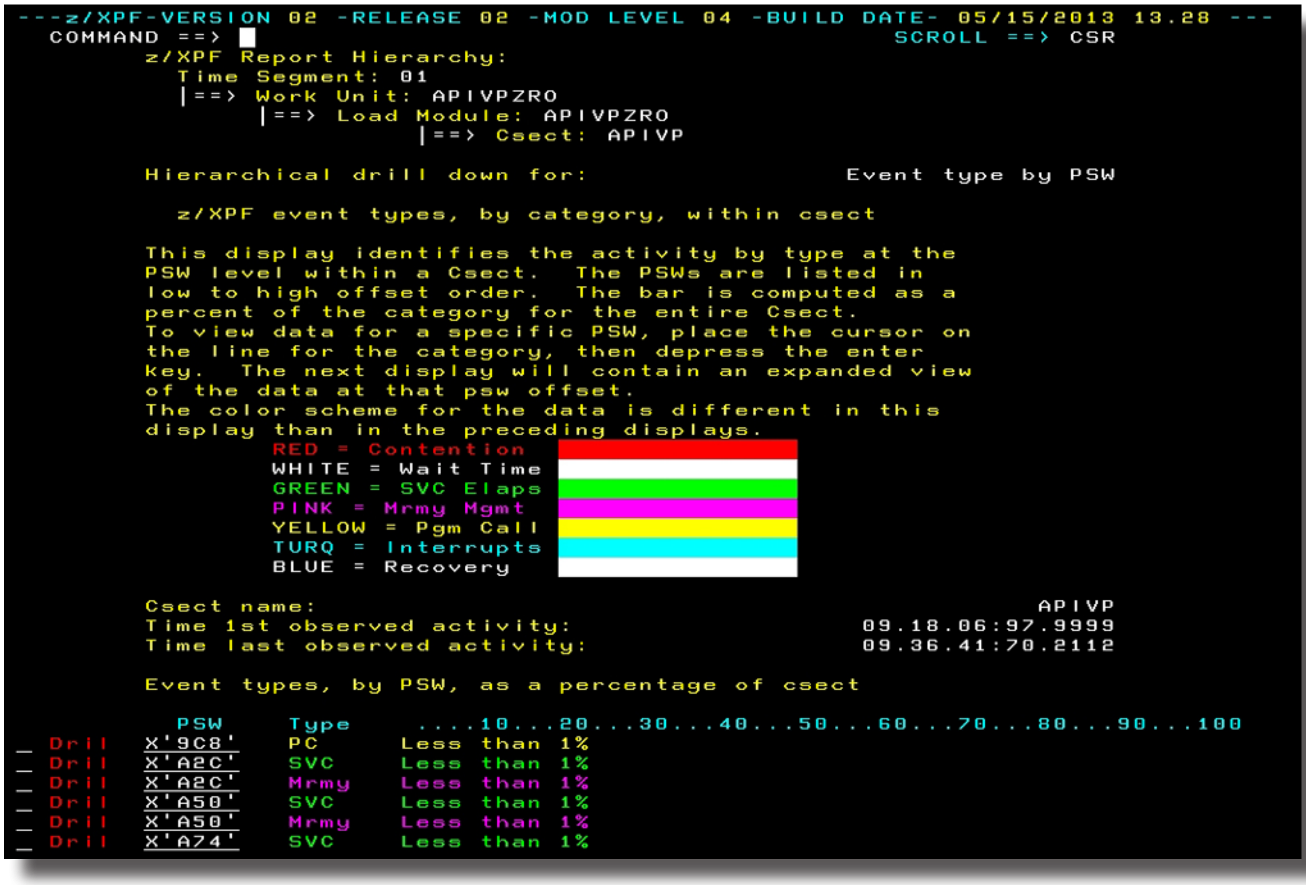
This time, instead of investigating Wait Time, I'll tab to the Work Unit APIVPZRO and drill down. That gets me from the Work Unit level to the load module level, and I want to drill into the APIVPZRO load module. I drill once more and arrive at Figure 5-16-10 below:



```
---z/XPF-VERSION 02 -RELEASE 02 -MOD LEVEL 04 -BUILD DATE- 05/15/2013 13.28 ---
   COMMAND ==>                                                   SCROLL ==> CSR
         z/XPF Report Hierarchy:
           Time Segment: 01
           |==> Work Unit: APIVPZRO
                |==> Load Module: APIVPZRO

         Hierarchical drill down for:                          Events by type
         Current level of report hierarchy:                             Csect
         Nbr of Csects identified, this Load Module:                        4

           z/XPF event types, by category, within Csect

         To view this display at the Csect PSW offset level,
         place the cursor on the line that contains the Csect
         name, then depress the Enter key.

         To view data for a specific category, place the
         cursor on the line for the category, then depress the
         enter key. The next display will then contain that
         data, at the Csect level, for the Load Module

_  Drill Csect:  APIVP
         Time 1st observed activity:                          09.18.06:97.9999
         Time last observed activity:                         09.36.41:70.2112

            Contention time, Wait time, and SVC elapsed as a
            percentage of Time Segment elapsed time.
                        ....10...20...30...40...50...60...70...80...90...100
_  Drill Contention
_  Drill Wait time
_  Drill SVC elapsed
```

Figure 5-16-10.

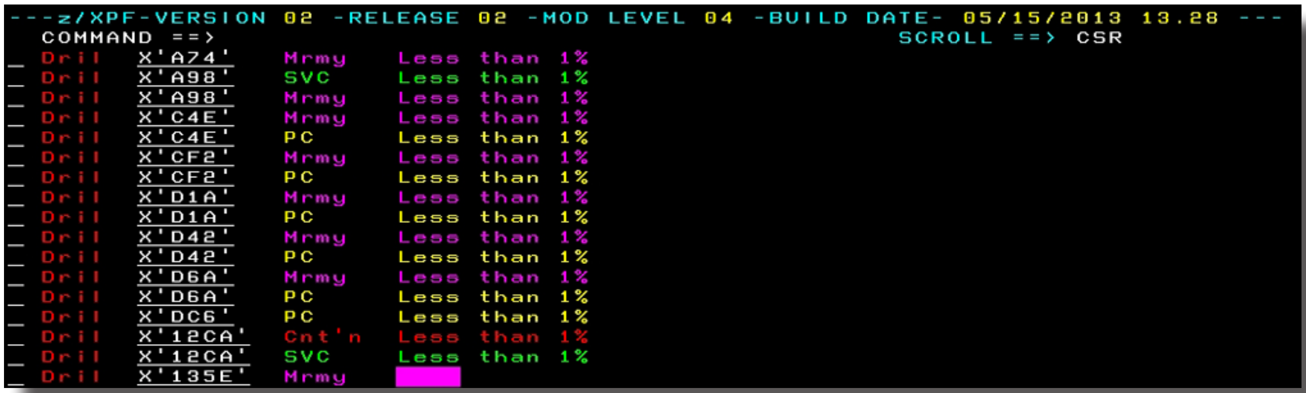
Now I see a list of csects, and the first one in the list is the APIVP csect. I drill down again. See Figure 5-16-11 below:



```
---z/XPF-VERSION 02 -RELEASE 02 -MOD LEVEL 04 -BUILD DATE- 05/15/2013 13.28 ---
   COMMAND ==>                                                   SCROLL ==> CSR
            z/XPF Report Hierarchy:
              Time Segment: 01
              |==> Work Unit: APIVPZRO
                   |==> Load Module: APIVPZRO
                             |==> Csect: APIVP

            Hierarchical drill down for:                  Event type by PSW

              z/XPF event types, by category, within csect

            This display identifies the activity by type at the
            PSW level within a Csect.  The PSWs are listed in
            low to high offset order.  The bar is computed as a
            percent of the category for the entire Csect.
            To view data for a specific PSW, place the cursor on
            the line for the category, then depress the enter
            key.  The next display will contain an expanded view
            of the data at that psw offset.
            The color scheme for the data is different in this
            display than in the preceding displays.
                    RED = Contention
                    WHITE = Wait Time
                    GREEN = SVC Elaps
                    PINK = Mrmy Mgmt
                    YELLOW = Pgm Call
                    TURQ = Interrupts
                    BLUE = Recovery

            Csect name:                                                   APIVP
            Time 1st observed activity:                      09.18.06:97.9999
            Time last observed activity:                     09.36.41:70.2112

            Event types, by PSW, as a percentage of csect

              PSW      Type     ....10...20...30...40...50...60...70...80...90...100
_ Drill   X'9C8'    PC       Less than 1%
_ Drill   X'A2C'    SVC      Less than 1%
_ Drill   X'A2C'    Mrmy     Less than 1%
_ Drill   X'A50'    SVC      Less than 1%
_ Drill   X'A50'    Mrmy     Less than 1%
_ Drill   X'A74'    SVC      Less than 1%
```

Figure 5-16-11.

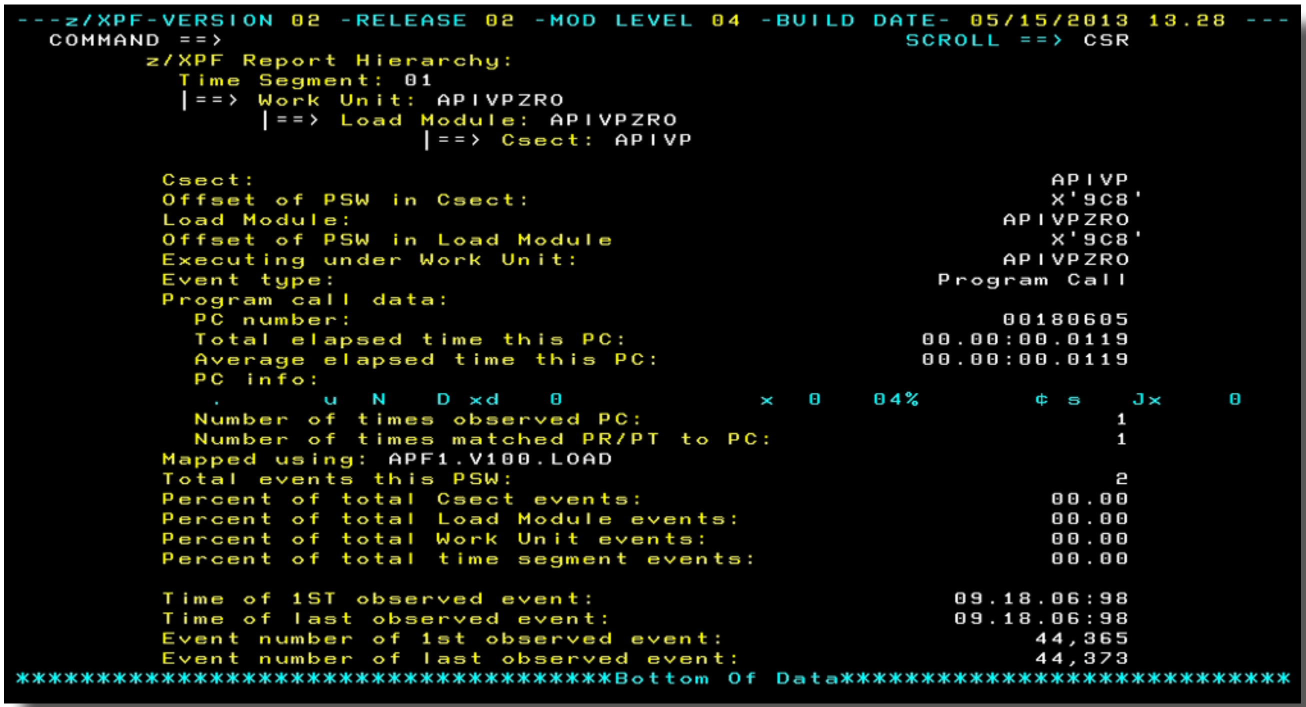
Now I get a MAP of all PSW offsets in my program with a color code to denote the kind of actvity, and the percentage of resources spent at that offset! WAY COOL. I'll perform a Page Down so you can see more of the report. See Figure 5-16-12 below:



```
---z/XPF-VERSION 02 -RELEASE 02 -MOD LEVEL 04 -BUILD DATE- 05/15/2013 13.28 ---
   COMMAND ==>                                                   SCROLL ==> CSR
_ Drill   X'A74'    Mrmy     Less than 1%
_ Drill   X'A98'    SVC      Less than 1%
_ Drill   X'A98'    Mrmy     Less than 1%
_ Drill   X'C4E'    Mrmy     Less than 1%
_ Drill   X'C4E'    PC       Less than 1%
_ Drill   X'CF2'    Mrmy     Less than 1%
_ Drill   X'CF2'    PC       Less than 1%
_ Drill   X'D1A'    Mrmy     Less than 1%
_ Drill   X'D1A'    PC       Less than 1%
_ Drill   X'D42'    Mrmy     Less than 1%
_ Drill   X'D42'    PC       Less than 1%
_ Drill   X'D6A'    Mrmy     Less than 1%
_ Drill   X'D6A'    PC       Less than 1%
_ Drill   X'DC6'    PC       Less than 1%
_ Drill   X'12CA'   Cnt'n    Less than 1%
_ Drill   X'12CA'   SVC      Less than 1%
_ Drill   X'135E'   Mrmy
```

Figure 5-16-12.

I'll Page Up to offset '9C8' and drill down one more time so you can see the final level of the report. See Figure 5-16-13 below:



```
---z/XPF-VERSION 02 -RELEASE 02 -MOD LEVEL 04 -BUILD DATE- 05/15/2013 13.28 ---
  COMMAND ==>                                                  SCROLL ==> CSR
        z/XPF Report Hierarchy:
          Time Segment: 01
          |==> Work Unit: APIVPZRO
               |==> Load Module: APIVPZRO
                         |==> Csect: APIVP

         Csect:                                                          APIVP
         Offset of PSW in Csect:                                        X'9C8'
         Load Module:                                                 APIVPZRO
         Offset of PSW in Load Module                                   X'9C8'
         Executing under Work Unit:                                   APIVPZRO
         Event type:                                              Program Call
         Program call data:
           PC number:                                                 00180605
           Total elapsed time this PC:                           00.00:00.0119
           Average elapsed time this PC:                         00.00:00.0119
           PC info:
            .      u  N   D xd   0             x  0   04%        ¢ s   Jx    0
           Number of times observed PC:                                      1
           Number of times matched PR/PT to PC:                              1
         Mapped using: APF1.V100.LOAD
         Total events this PSW:                                              2
         Percent of total Csect events:                                  00.00
         Percent of total Load Module events:                            00.00
         Percent of total Work Unit events:                              00.00
         Percent of total time segment events:                           00.00

         Time of 1ST observed event:                               09.18.06:98
         Time of last observed event:                              09.18.06:98
         Event number of 1st observed event:                            44,365
         Event number of last observed event:                           44,373
*************************************Bottom Of Data*****************************
```

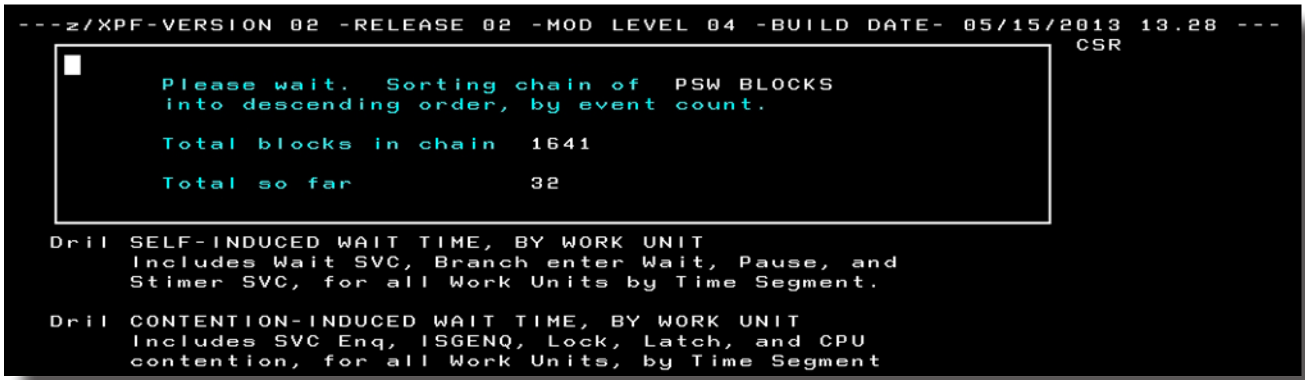
Figure 5-16-13.

We have arrived at the final depth of the report. At offset 9C8 within csect APIVP within load module APIVPZRO running under Work Unit APIVPZRO within the single Time Segment in this data capture report, a program call has been made to PC number 180605. It took an elapsed time of 119 micro-seconds to complete. The "PC Info" field contains the first 60 or so bytes of the Program Call as an "eye-catcher". There is more on the screen for you to look over. Stare at it for a few seconds. This is GOOD STUFF.

z/XPF's dynamic ISPF Summary Reporting is a nice advance over the previous "tabular-only" approach. We think it will greatly speed you in your work with the product.

5-17 Most Frequently Observed PSW/Instruction Report

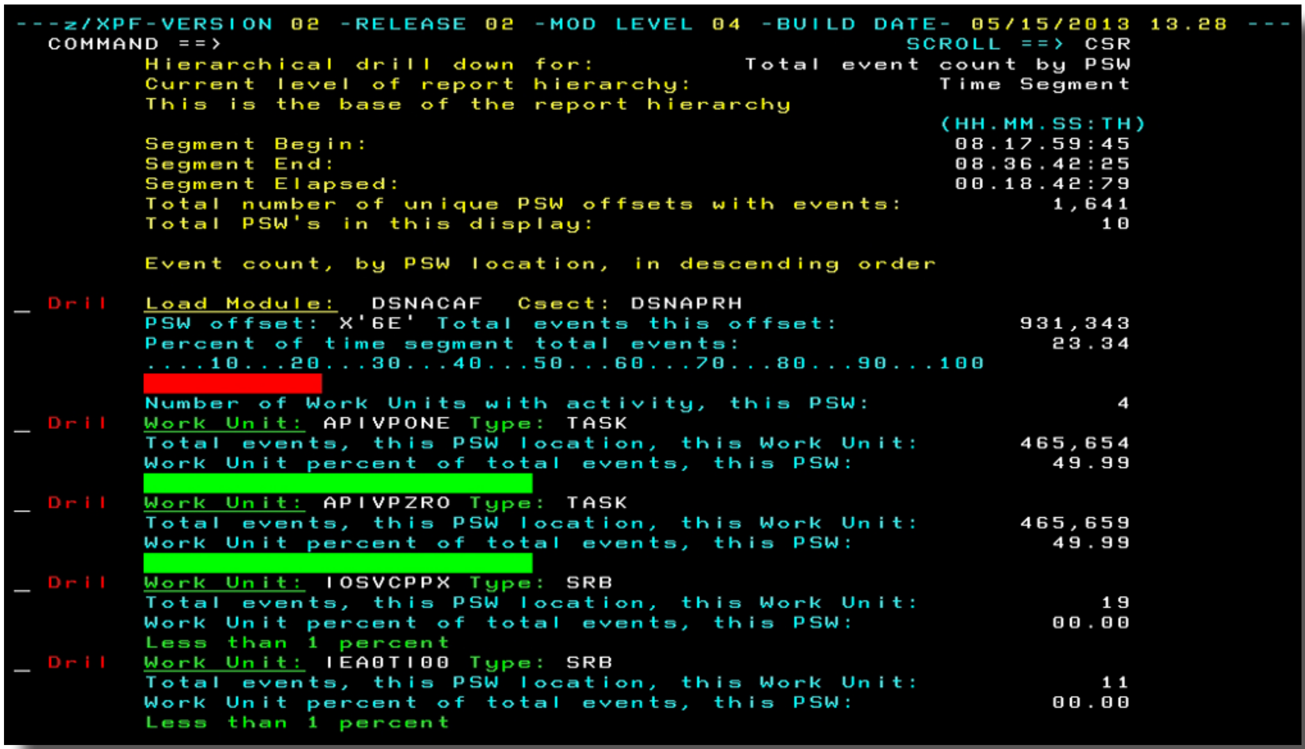
My favorite z/XPF report is the “MOST FREQUENTLY OBSERVED PSW/INSTRUCTION”, Report, which occurs on the next page of the panel (I am lobbying to have it moved to the top!). This report is a great initial way to find WHERE your programs are spending most of their time. We expect that anyone is going to want to see this first. I’ll put the cursor next to Most Frequently Observed PSW/Instruction and press Enter. z/XPF performs a sort operation for you that looks a bit like figure 5-32 below:



```
---z/XPF-VERSION 02 -RELEASE 02 -MOD LEVEL 04 -BUILD DATE- 05/15/2013 13.28 ---
                                                                       CSR
           Please wait.  Sorting chain of  PSW BLOCKS
           into descending order, by event count.

           Total blocks in chain    1641

           Total so far             32

  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
```

Figure 5-17-1.

z/XPF is now sorting the PSW blocks it has identified, into a sequence of “most active” to least active”. When the sort is completed, you see something like Figure 5-17-2 below:

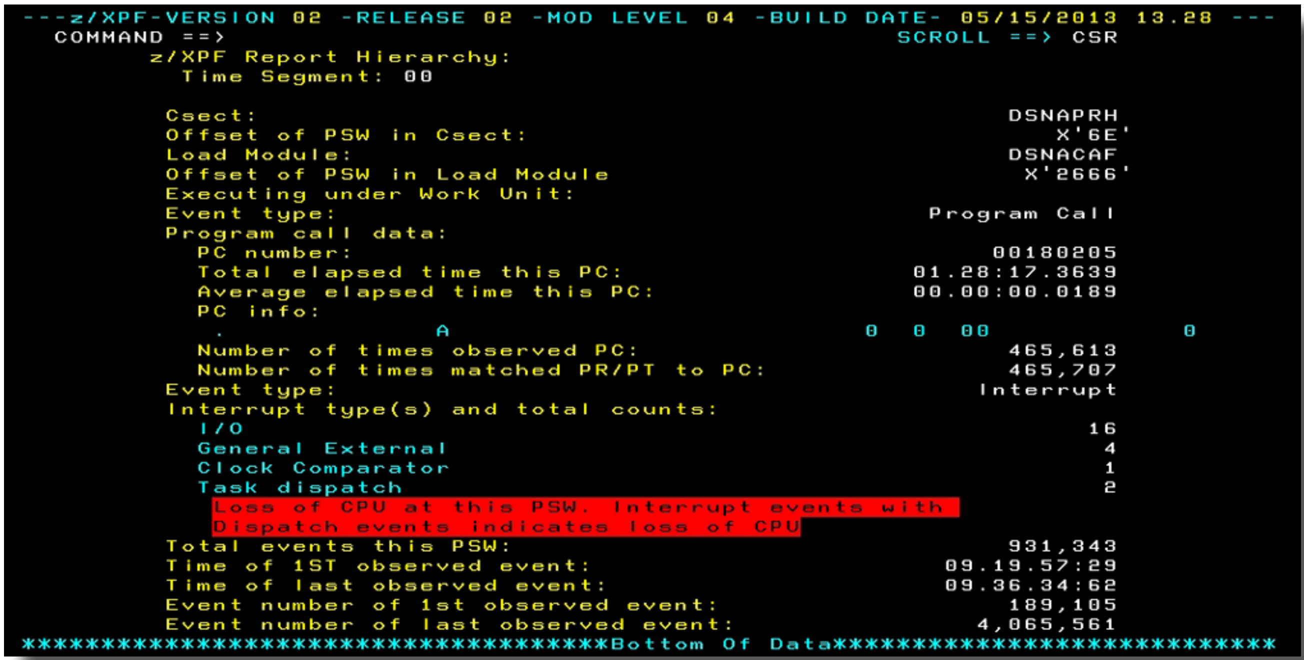


```
---z/XPF-VERSION 02 -RELEASE 02 -MOD LEVEL 04 -BUILD DATE- 05/15/2013 13.28 ---
  COMMAND ==>                                              SCROLL ==> CSR
        Hierarchical drill down for:          Total event count by PSW
        Current level of report hierarchy:                Time Segment
        This is the base of the report hierarchy
                                                           (HH.MM.SS:TH)
        Segment Begin:                                      08.17.59:45
        Segment End:                                        08.36.42:25
        Segment Elapsed:                                    00.18.42:79
        Total number of unique PSW offsets with events:          1,641
        Total PSW's in this display:                                10

        Event count, by PSW location, in descending order

_ Dril  Load Module:  DSNACAF   Csect: DSNAPRH
        PSW offset: X'6E' Total events this offset:            931,343
        Percent of time segment total events:                    23.34
        ....10...20...30...40...50...60...70...80...90...100
        🟥🟥🟥🟥🟥
        Number of Work Units with activity, this PSW:                4
_ Dril  Work Unit: APIVPONE Type: TASK
        Total events, this PSW location, this Work Unit:       465,654
        Work Unit percent of total events, this PSW:             49.99
        🟩🟩🟩🟩🟩🟩🟩🟩🟩🟩
_ Dril  Work Unit: APIVPZRO Type: TASK
        Total events, this PSW location, this Work Unit:       465,659
        Work Unit percent of total events, this PSW:             49.99
        🟩🟩🟩🟩🟩🟩🟩🟩🟩🟩
_ Dril  Work Unit: IOSVCPPX Type: SRB
        Total events, this PSW location, this Work Unit:            19
        Work Unit percent of total events, this PSW:             00.00
        Less than 1 percent
_ Dril  Work Unit: IEA0TI00 Type: SRB
        Total events, this PSW location, this Work Unit:            11
        Work Unit percent of total events, this PSW:             00.00
        Less than 1 percent
```

Figure 5-17-2.

Understanding this report takes just a bit of orientation. The MOST FREQUENTLY OBSERVED PSW/INSTRUCTION Report inverts the normal hierarchy because it's oriented towards the individual PSW. So, this report lists the most active PSW addresses for this data capture. You can see that in load module DSNACAF's csect DSNAPRH had one offset at x'6E' that accounted for 23.34 percent of the entire run. You can also see that FOUR Work Units (APIVPONE, APIVPZRO, IOSVCPPX and IEA0T1000) hit that offset, and each one is listed.

So, let's drill down into load module DSNACAF. I'll put the cursor on the "Load Module" field next to DSNACAF and press Enter. Next, I see Figure 5-17-3 below:



```
---z/XPF-VERSION 02 -RELEASE 02 -MOD LEVEL 04 -BUILD DATE- 05/15/2013 13.28 ---
  COMMAND ==>                                                    SCROLL ==> CSR
        z/XPF Report Hierarchy:
          Time Segment: 00

         Csect:                                                         DSNAPRH
         Offset of PSW in Csect:                                          X'6E'
         Load Module:                                                   DSNACAF
         Offset of PSW in Load Module                                   X'2666'
         Executing under Work Unit:
         Event type:                                               Program Call
         Program call data:
           PC number:                                                  00180205
           Total elapsed time this PC:                            01.28:17.3639
           Average elapsed time this PC:                          00.00:00.0189
           PC info:
            .              A                            0  0  00               0
           Number of times observed PC:                                 465,613
           Number of times matched PR/PT to PC:                         465,707
         Event type:                                                  Interrupt
         Interrupt type(s) and total counts:
           I/O                                                               16
           General External                                                   4
           Clock Comparator                                                   1
           Task dispatch                                                      2
            Loss of CPU at this PSW. Interrupt events with
            Dispatch events indicates loss of CPU
         Total events this PSW:                                         931,343
         Time of 1ST observed event:                                09.19.57:29
         Time of last observed event:                               09.36.34:62
         Event number of 1st observed event:                            189,105
         Event number of last observed event:                         4,065,561
*************************************Bottom Of Data*****************************
```

Figure 5-17-3.

Here we see a comprehensive display of csect DSNAPRH with the most active PSW offset at X'6E', the load module (DSNACAF) that contains it, the location of offset X'6E' within the parent module and a WHOLE lot of other pertinent information having to do with it (including the "PC info" field, the first sixty characters code as a sort of "eye-catcher"). Looks like PC number 180205 got called here, and this happened 465,613 times. We also see a red field that indicates that there was a loss of the CPU at this point, because there was an interrupt, and a dispatch.

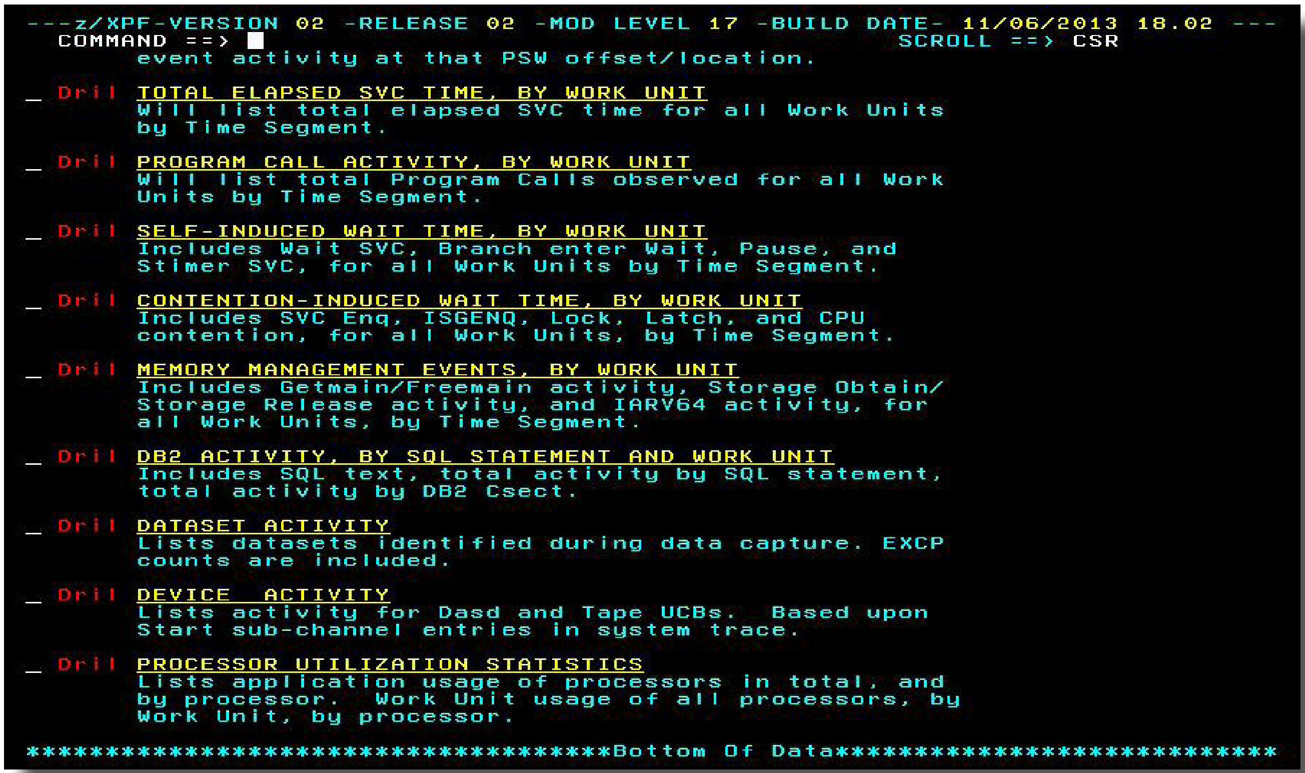
[Waxing philosophic for a moment: z/XPF generally either confirms an investigator's suspicions about resource consumption or it SURPRISES that person. It's the surprises that often lead folks to tune their code, and realize greater efficiencies. That's the pay-off.]

5-18 Processor Utilization Statistics

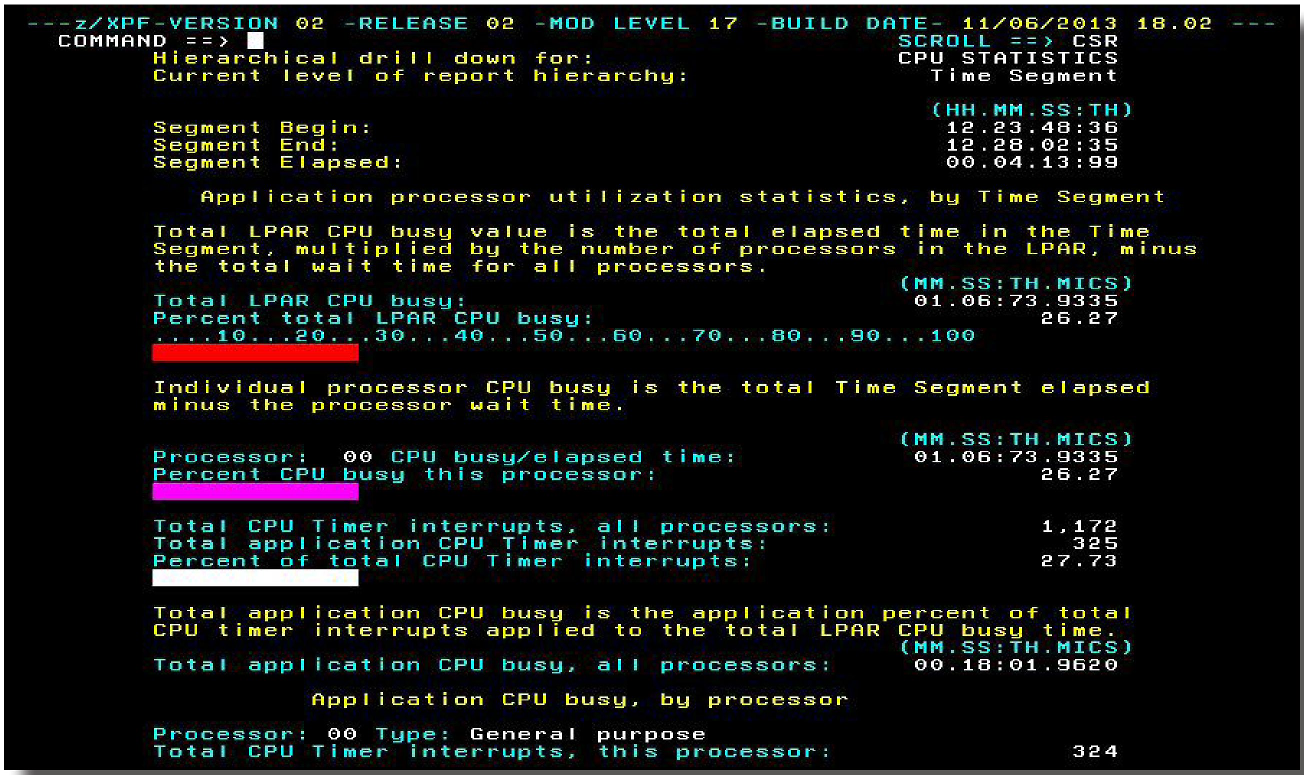
Dave Day has found what we believe to be a TRUE measurement of CPU consumption. Other profilers measure the frequency with which particular PSWs appear during sampling, and label this "CPU Consumption". That's not really so. It's merely a report of PSWs,which fails to account for downstream calls from 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 t 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 5-18-1:


```
---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 5-18-1.

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 5-18-2.

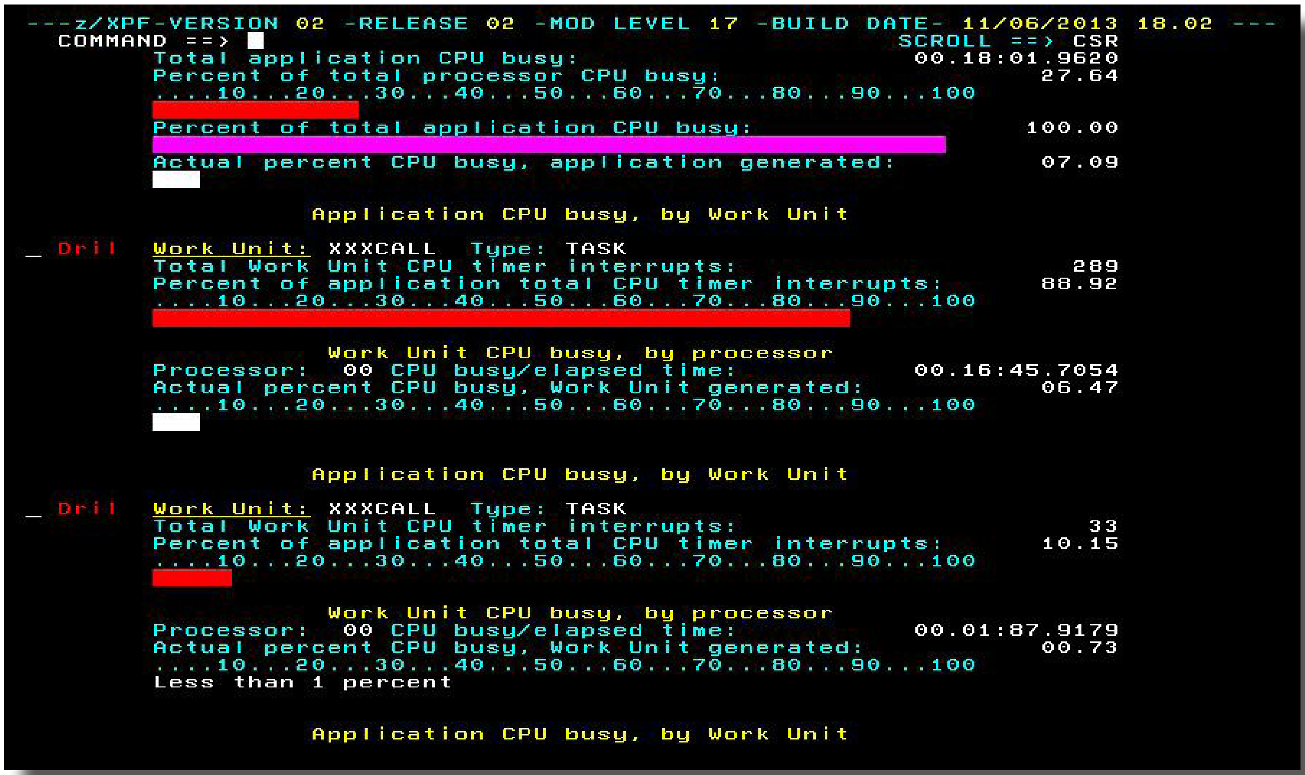


```
---z/XPF-VERSION 02 -RELEASE 02 -MOD LEVEL 17 -BUILD DATE- 11/06/2013 18.02 ---
   COMMAND ==>                                           SCROLL ==> CSR
         Hierarchical drill down for:                    CPU STATISTICS
         Current level of report hierarchy:                Time Segment

                                                           (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 5-18-2.

Figure 5-36 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.

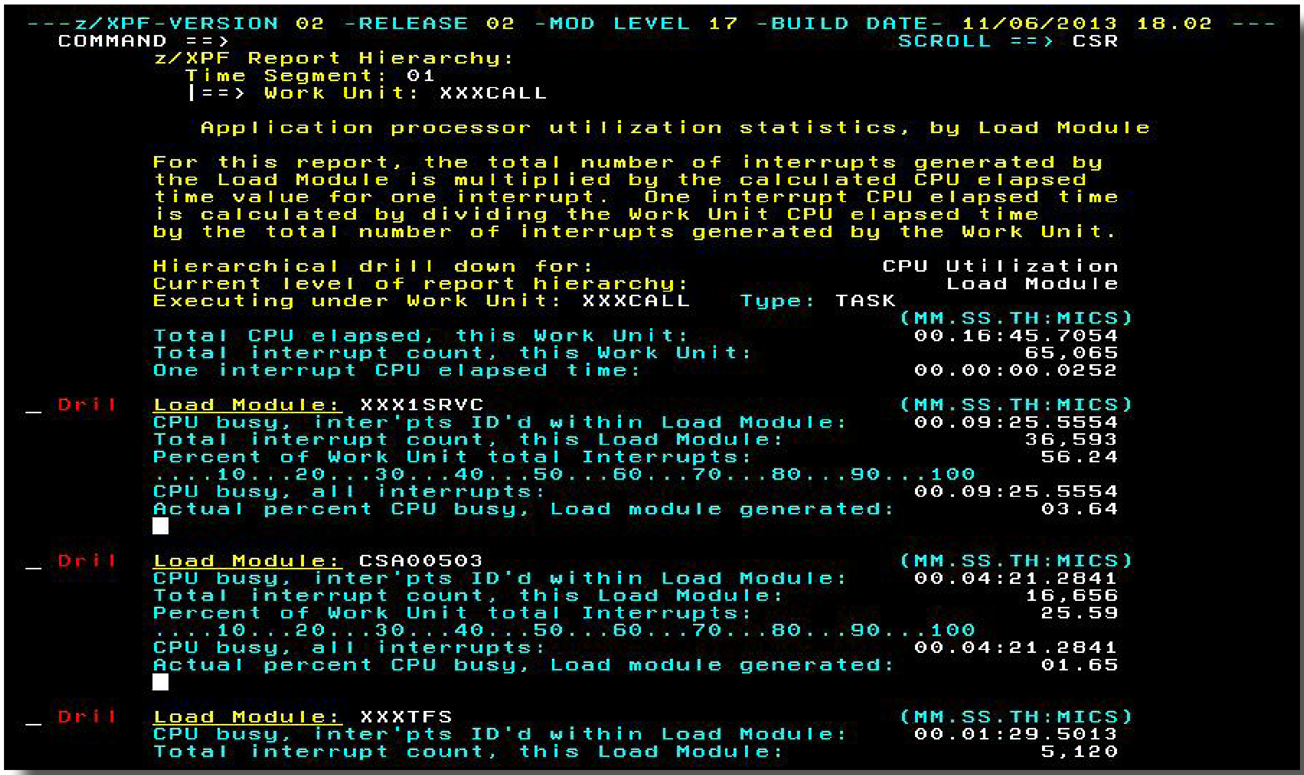
If I page down from here, I'll see the display shown in Figure 5-18-3 below:



```
---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 5-18-3.

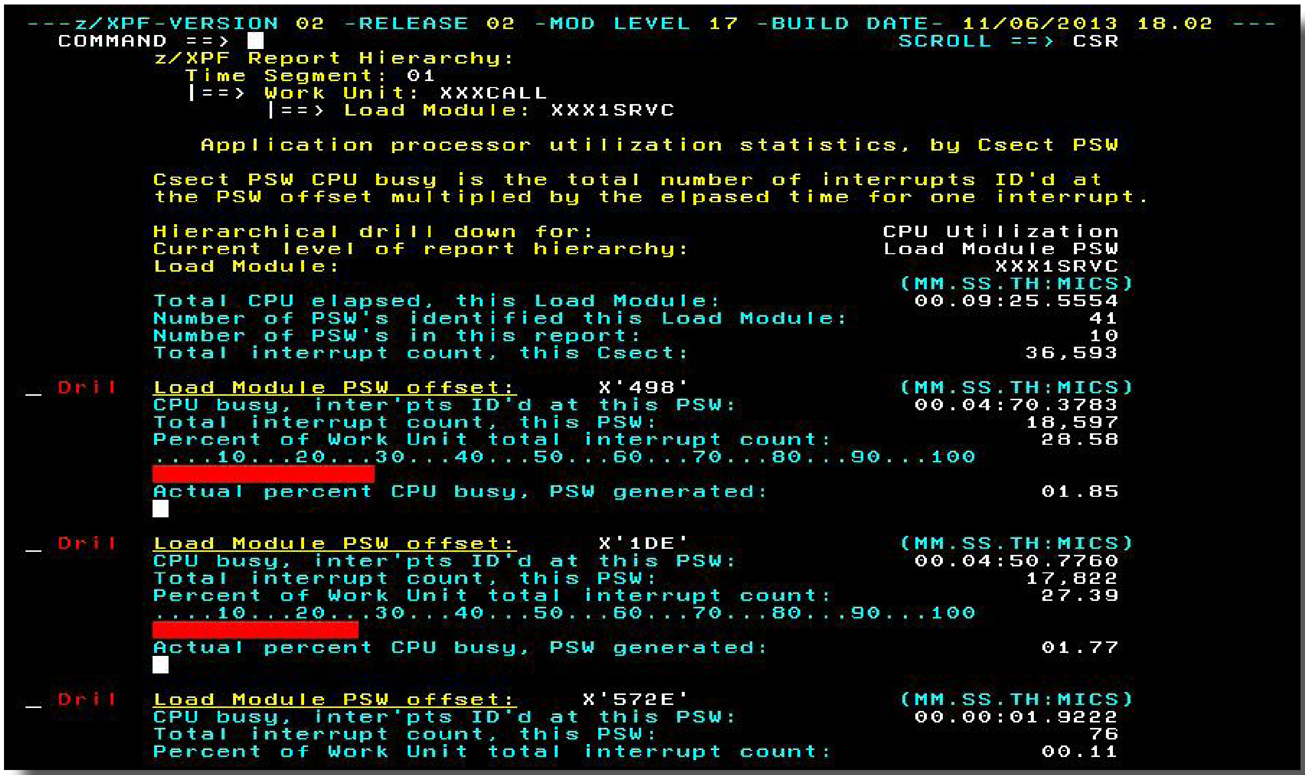
In Figure 5-18-3 I have a continuation of the “overview” information. Below that I see Work Units for the Time Segment with the biggest CPU Consumer sorted to the top. In this case, it is XXXCALL, a Task that used 289 of the 324 Timer Interrupts observed in the Time Segment. Let’s drill down again, and we’ll see Figure 5-18-4, 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 5-18-4.

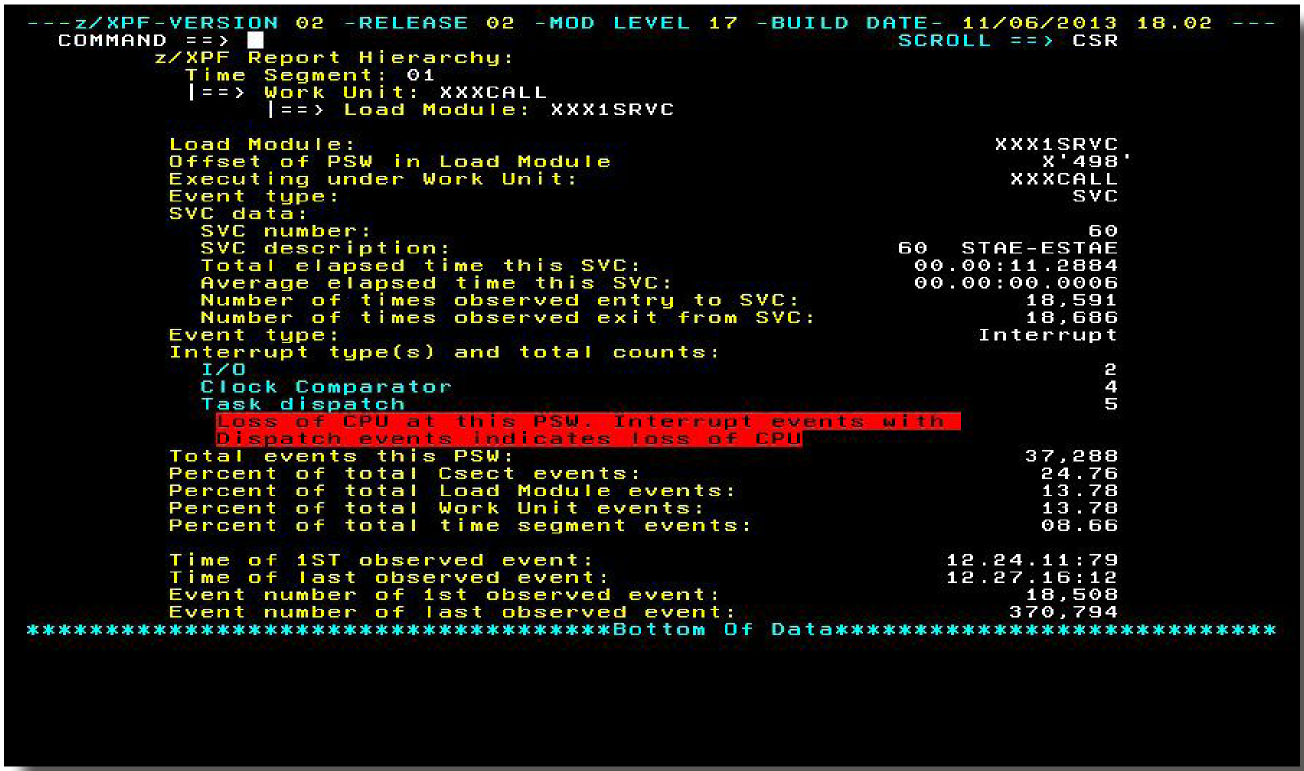
Now z/XPF has dropped me from the Work Unit level to the Load Module Level (just as it does in all the other Summary Reports. Here I see that Load Module XXX1SRVC generated 36,593 Interrupts (now we're tracking ALL interrupts not just Timer Interrupts), or 56.24% of the Work Unit. Let's drill again, and we'll see Figure 5-18-5, 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 5-18-5.

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 5-18-6:



```
---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 5-18-6.

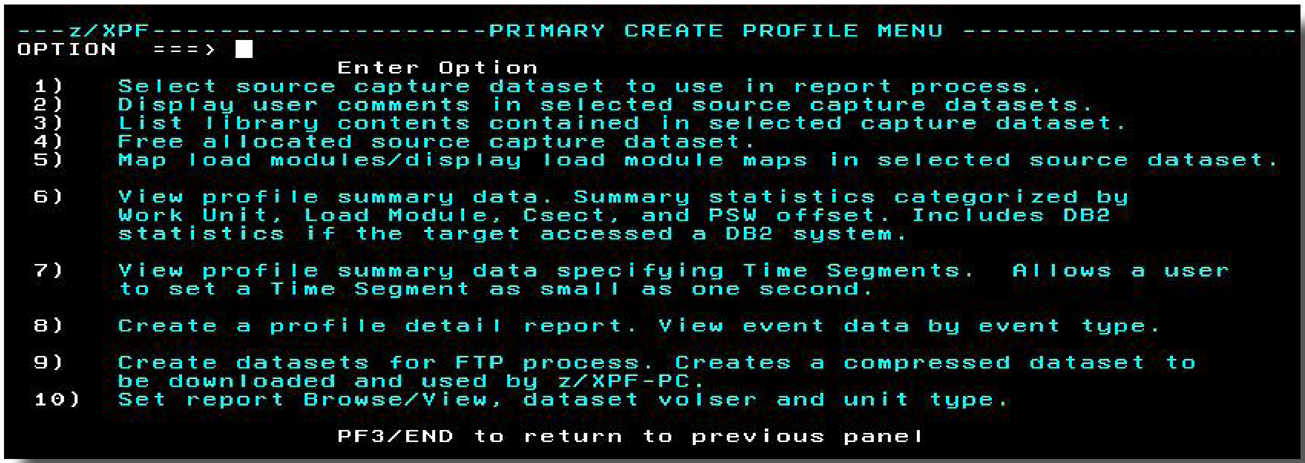
We've arrived at the final level of the report, and we can see that at this PSW address a call has been to SVC 60, the STAE-ESTAE Supervisor call.

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

5-19 Option 7: Creating much smaller Time Segments in your report

z/XPF no longer limits the user to only ten Time Segments within a report. Now, using Option 7 you can divide your report into Time Segments with a duration of from 60 seconds down to one-second durations. Then, you can use z/XPF's Summary Reporting to isolate by Time Segment and Drill/Expand within these small Time Segments.

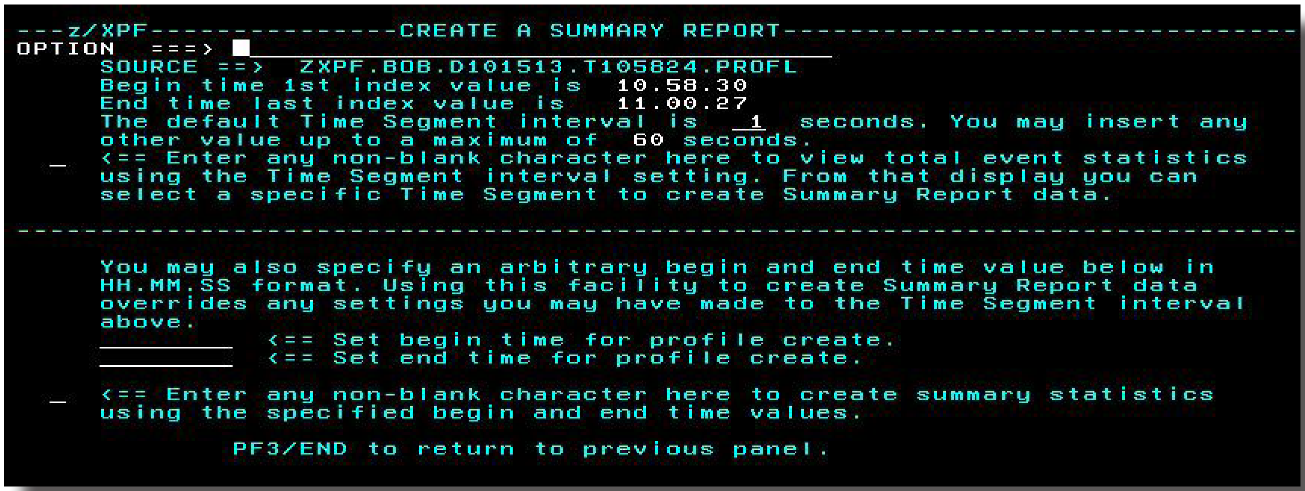
When you select Option 7, you'll see the display below in figure 5-19-1.



```
---z/XPF---------------------PRIMARY CREATE PROFILE MENU --------------------
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 5-19-1.

If you select Option 7, you'll see the panel below, in figure 5-19-2:



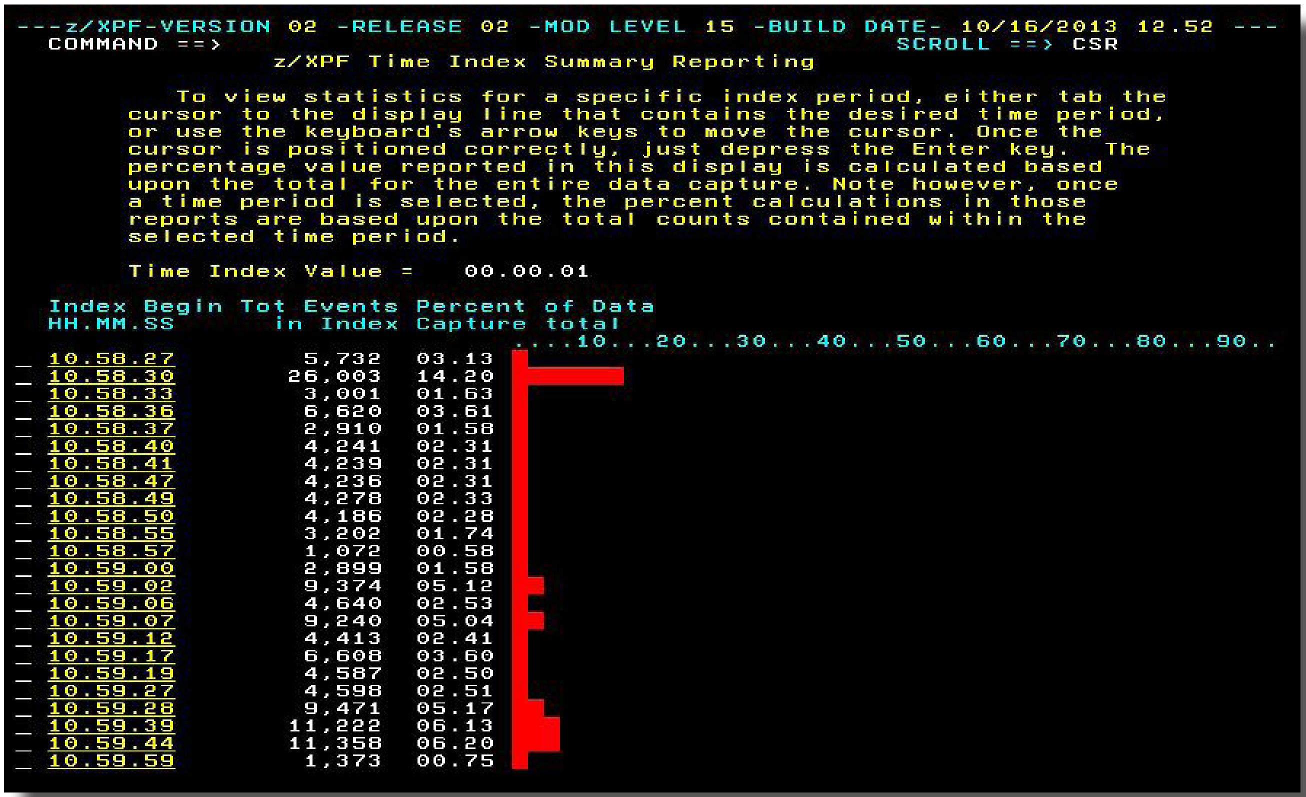
```
---z/XPF----------------CREATE A SUMMARY REPORT------------------------------
OPTION  ===> 
     SOURCE ==>  ZXPF.BOB.D101513.T105824.PROFL
     Begin time 1st index value is   10.58.30
     End time last index value is    11.00.27
     The default Time Segment interval is  _1  seconds. You may insert 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 5-19-2.

This screen is divided into two logical parts: You can either alter the Time Segment values from 1 second to up to 60 seconds, or you can specify any 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).

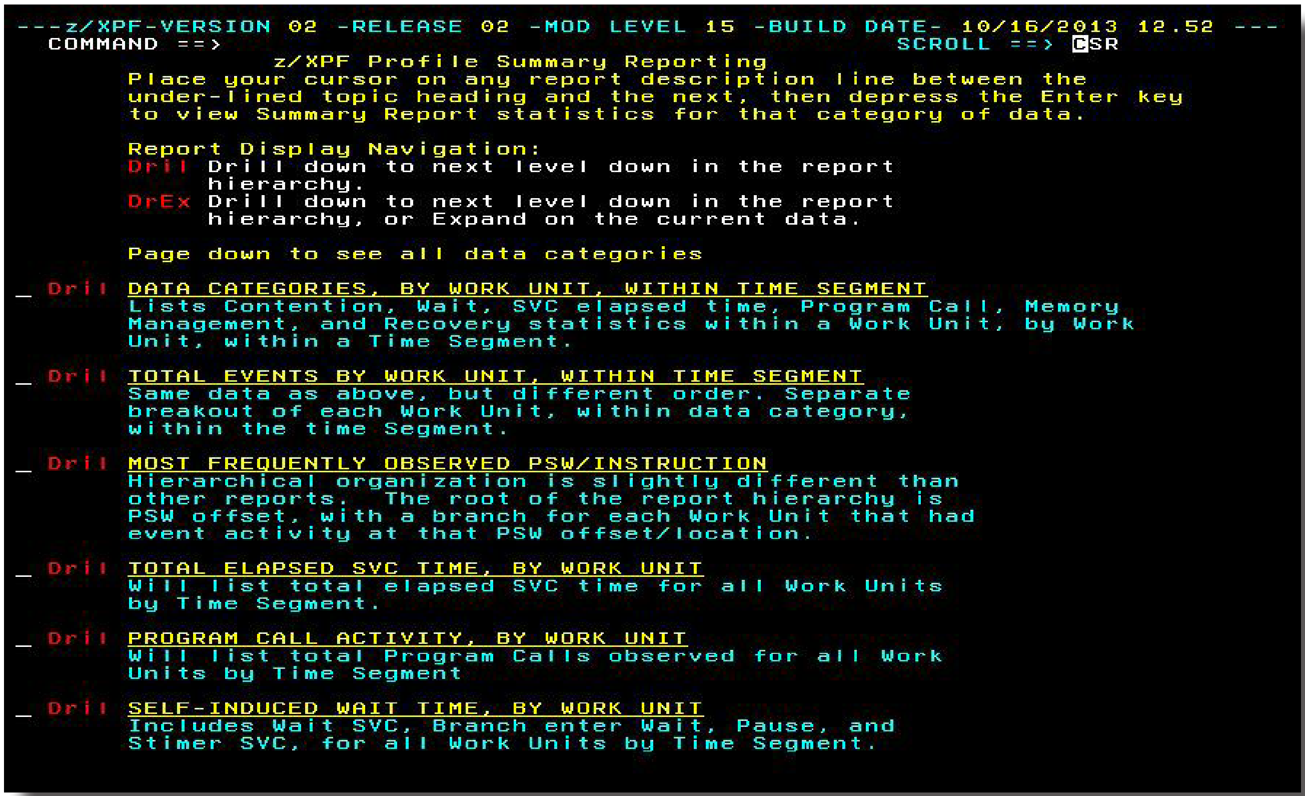
I'll allow a one-second time interval for a z/XPF data capture, and put "S" into the input field on the top portion of the screen. When I do so, z/XPF parses out my report and shows me the panel below, in Figure 5-19-3:



```
---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 5-19-3.

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 5-19-4:



```
---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 5-19-4.

From this point on, I can drill down or expand as I care to. It's all right there.

Folks: The rest of these reports function in much the same way, with opportunities for you to drill or expand categories along the way. So, in an attempt at brevity, I'm going leave Summary Reporting's description as it is (or risk doubling the size of this book!). As always questions and comments are welcome!

5-20 Source Statement Support

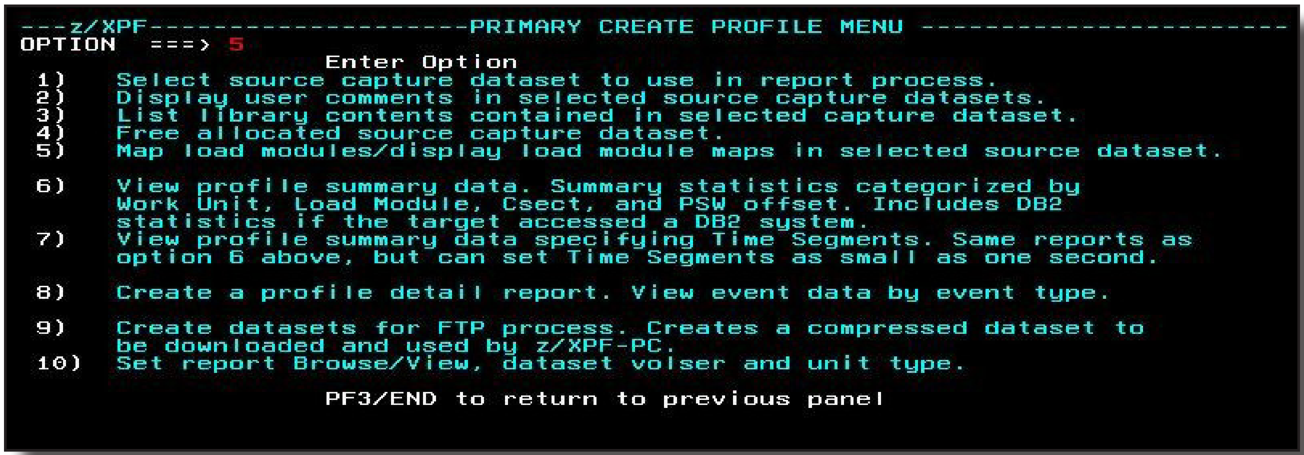
As of Release V2R2M24, z/XPF has the ability to show you source statement displays in its Summary Reporting. Very basically, z/XPF accesses the listing for the selected program(s), and allows you to see that source code when you are at the PSW level in a Summary Report. When you do get to the PSW level, you issue the “SOURCE” command from the command line, or use the “S” command next to any PSW in the display. z/XPF then retrieves the listing and places it into the report.

Note:

- z/XPF can show source statements ONLY for modules that have Binder maps;
- The association within z/XPF for a source listing is at the csect level.

The first step is to perform your data capture in the normal manner. When you have stopped the data capture, you then allocate the resulting VSAM dataset, also in the normal fashion.

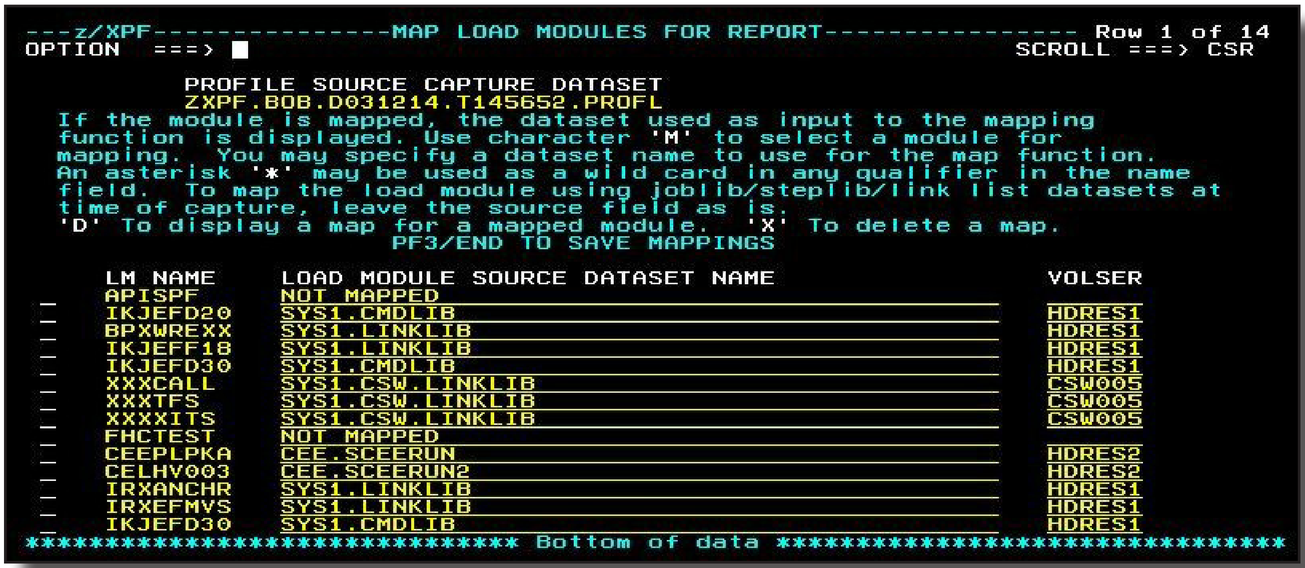
At this point, it is a good idea to make sure that the modules you are interested in were mapped during data capture. This step isn’t used to access the source listing. Instead, it shows whether or not the Binder was successful in mapping your module. Use Option 5 from the Primary Create Profile Menu to see if the z/XPF mapped the module or not. Here’s an example of the panel in Figure 5-20-1:



```
---z/XPF---------------------PRIMARY CREATE PROFILE MENU -----------------------
OPTION  ===> 5
                   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. Same reports as
       option 6 above, but can set Time Segments 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 5-20-1.

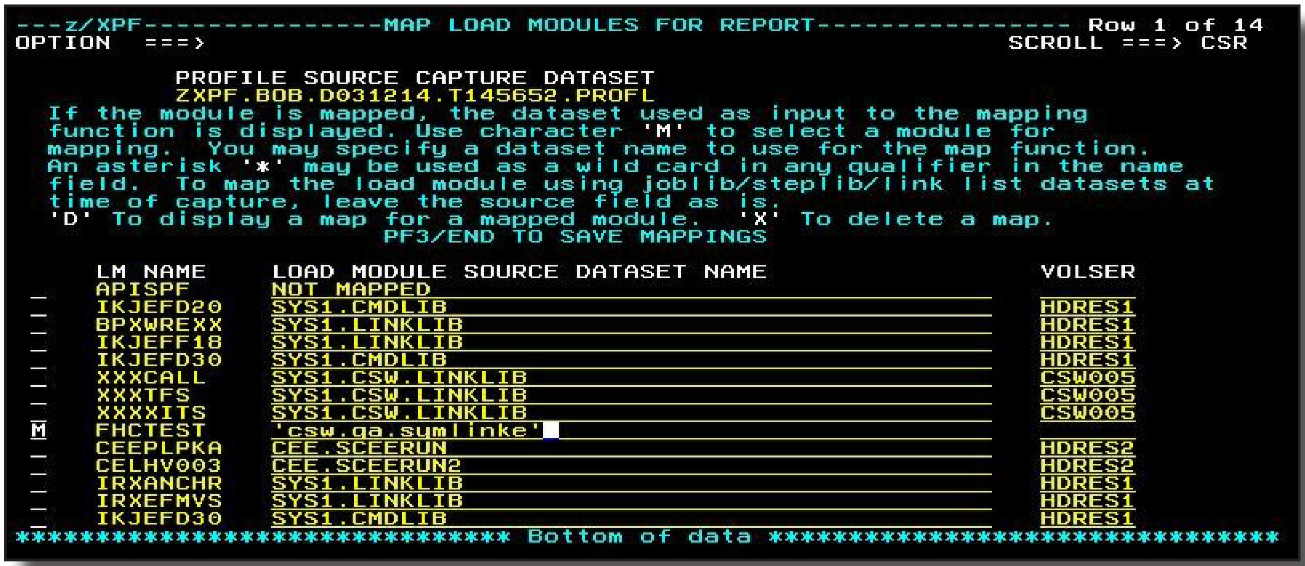
to look at the source code for our test program FHCTEST. You can see that it has not been mapped, due to the message "NOT MAPPED". See Figure 5-20-2 below:



```
---z/XPF----------------MAP LOAD MODULES FOR REPORT---------------- Row 1 of 14
OPTION  ===>                                                    SCROLL ===> CSR

          PROFILE SOURCE CAPTURE DATASET
          ZXPF.BOB.D031214.T145652.PROFL
  If the module is mapped, the dataset used as input to the mapping
  function is displayed. Use character 'M' to select a module for
  mapping.  You may specify a dataset name to use for the map function.
  An asterisk '*' may be used as a wild card in any qualifier in the name
  field.  To map the load module using joblib/steplib/link list datasets at
  time of capture, leave the source field as is.
  'D' To display a map for a mapped module.   'X' To delete a map.
                         PF3/END TO SAVE MAPPINGS

     LM NAME    LOAD MODULE SOURCE DATASET NAME                   VOLSER
     APISPF     NOT MAPPED
 _   IKJEFD20   SYS1.CMDLIB                                       HDRES1
 _   BPXWREXX   SYS1.LINKLIB                                      HDRES1
 _   IKJEFF18   SYS1.LINKLIB                                      HDRES1
 _   IKJEFD30   SYS1.CMDLIB                                       HDRES1
 _   XXXCALL    SYS1.CSW.LINKLIB                                  CSW005
 _   XXXTFS     SYS1.CSW.LINKLIB                                  CSW005
 _   XXXXITS    SYS1.CSW.LINKLIB                                  CSW005
 _   FHCTEST    NOT MAPPED
 _   CEEPLPKA   CEE.SCEERUN                                       HDRES2
 _   CELHV003   CEE.SCEERUN2                                      HDRES2
 _   IRXANCHR   SYS1.LINKLIB                                      HDRES1
 _   IRXEFMVS   SYS1.LINKLIB                                      HDRES1
 _   IKJEFD30   SYS1.CMDLIB                                       HDRES1
******************************* Bottom of data ********************************
```

Figure 5-20-2.

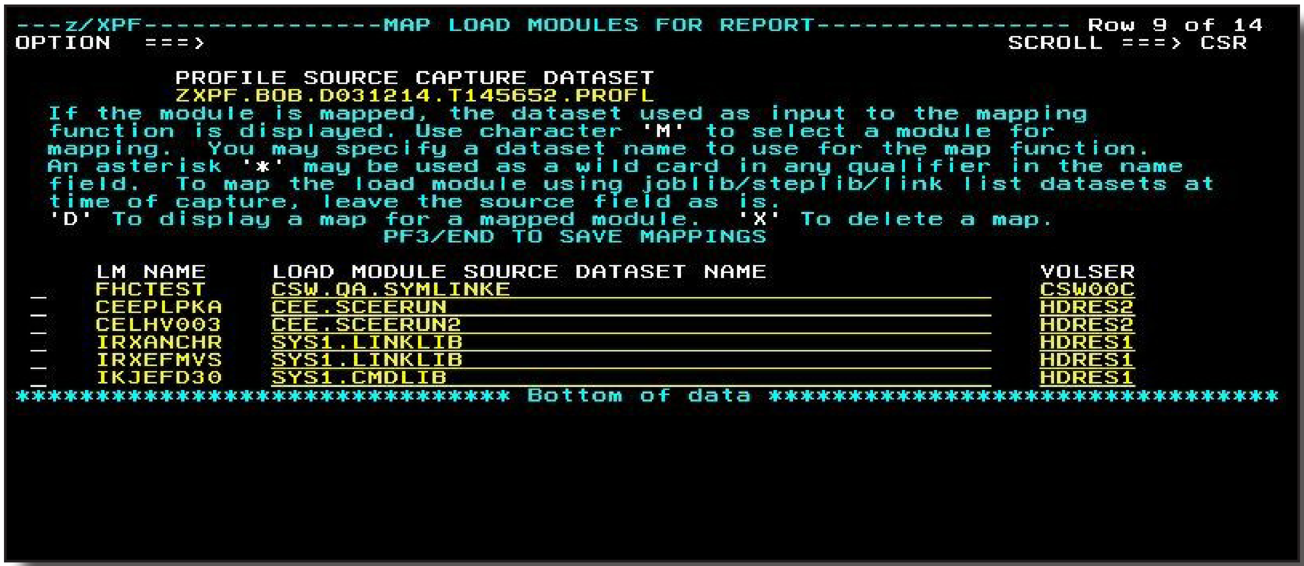
To map the module manually, enter the library that contains the load module, surrounded by single quotation marks, then press the Enter key. You can see that I've entered the dataset name below, in Figure 5-20-3:



```
---z/XPF----------------MAP LOAD MODULES FOR REPORT---------------- Row 1 of 14
OPTION  ===>                                                    SCROLL ===> CSR

          PROFILE SOURCE CAPTURE DATASET
          ZXPF.BOB.D031214.T145652.PROFL
  If the module is mapped, the dataset used as input to the mapping
  function is displayed. Use character 'M' to select a module for
  mapping.  You may specify a dataset name to use for the map function.
  An asterisk '*' may be used as a wild card in any qualifier in the name
  field.  To map the load module using joblib/steplib/link list datasets at
  time of capture, leave the source field as is.
  'D' To display a map for a mapped module.   'X' To delete a map.
                         PF3/END TO SAVE MAPPINGS

     LM NAME    LOAD MODULE SOURCE DATASET NAME                   VOLSER
     APISPF     NOT MAPPED
 _   IKJEFD20   SYS1.CMDLIB                                       HDRES1
 _   BPXWREXX   SYS1.LINKLIB                                      HDRES1
 _   IKJEFF18   SYS1.LINKLIB                                      HDRES1
 _   IKJEFD30   SYS1.CMDLIB                                       HDRES1
 _   XXXCALL    SYS1.CSW.LINKLIB                                  CSW005
 _   XXXTFS     SYS1.CSW.LINKLIB                                  CSW005
 _   XXXXITS    SYS1.CSW.LINKLIB                                  CSW005
 M   FHCTEST    'csw.qa.symlinke'
 _   CEEPLPKA   CEE.SCEERUN                                       HDRES2
 _   CELHV003   CEE.SCEERUN2                                      HDRES2
 _   IRXANCHR   SYS1.LINKLIB                                      HDRES1
 _   IRXEFMVS   SYS1.LINKLIB                                      HDRES1
 _   IKJEFD30   SYS1.CMDLIB                                       HDRES1
******************************* Bottom of data ********************************
```

Figure 5-20-3.

Now, the panel refreshes and you can see that z/XPF accepted the library name in Figure 5-20-4 below:

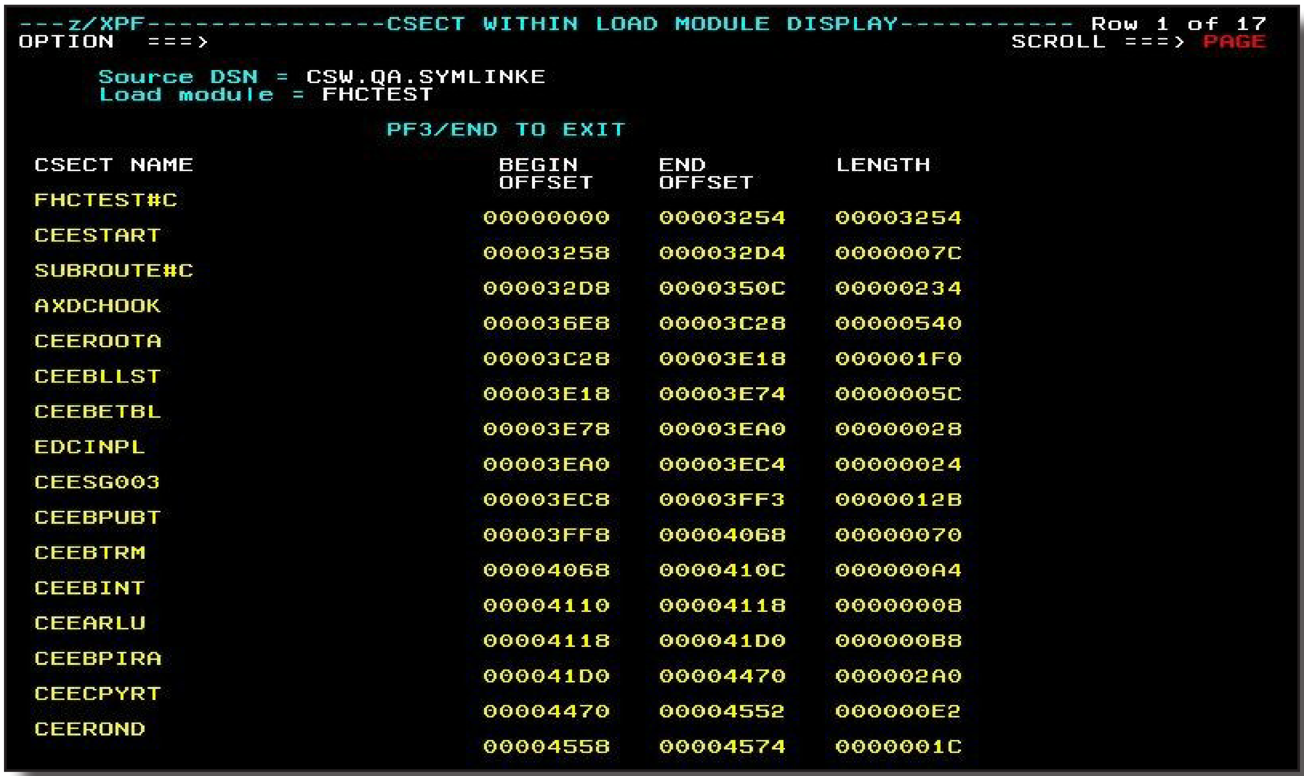


```
---z/XPF----------------MAP LOAD MODULES FOR REPORT---------------- Row 9 of 14
OPTION   ===>                                                  SCROLL ===> CSR

          PROFILE SOURCE CAPTURE DATASET
          ZXPF.BOB.D031214.T145652.PROFL
  If the module is mapped, the dataset used as input to the mapping
  function is displayed. Use character 'M' to select a module for
  mapping.  You may specify a dataset name to use for the map function.
  An asterisk '*' may be used as a wild card in any qualifier in the name
  field.  To map the load module using joblib/steplib/link list datasets at
  time of capture, leave the source field as is.
  'D' To display a map for a mapped module.   'X' To delete a map.
                          PF3/END TO SAVE MAPPINGS

     LM NAME     LOAD MODULE SOURCE DATASET NAME                   VOLSER
 _   FHCTEST     CSW.QA.SYMLINKE                                   CSW00C
 _   CEEPLPKA    CEE.SCEERUN                                       HDRES2
 _   CELHV003    CEE.SCEERUN2                                      HDRES2
 _   IRXANCHR    SYS1.LINKLIB                                      HDRES1
 _   IRXEFMVS    SYS1.LINKLIB                                      HDRES1
 _   IKJEFD30    SYS1.CMDLIB                                       HDRES1
******************************* Bottom of data ********************************
```

Figure 5-20-4.

Of course, if your module has already been mapped you can skip this step, but it's a good idea to check, just in case it is not.

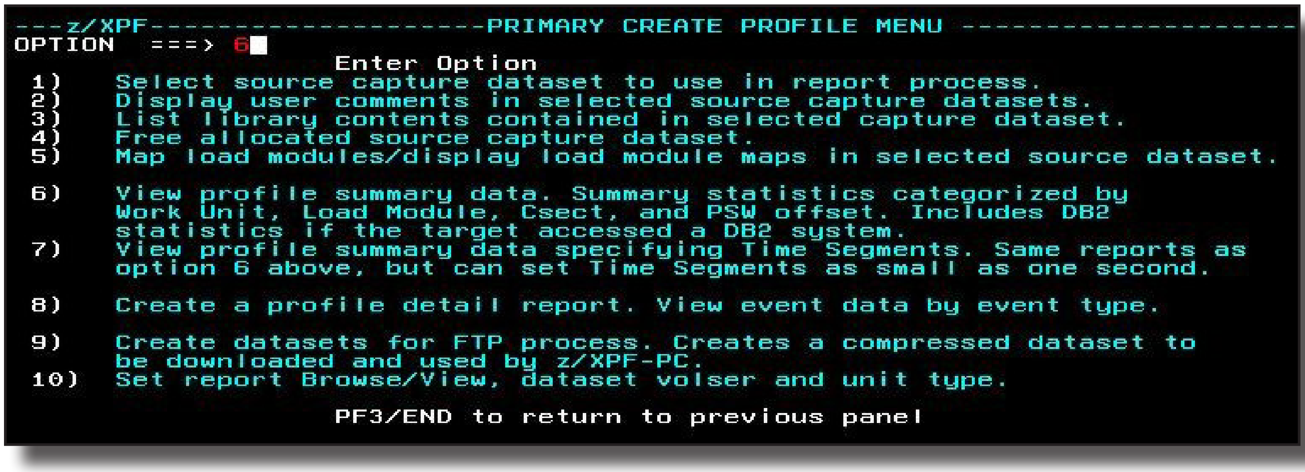
If you care to, you can enter a "D" next to your module to see the csects, offset addresses and lengths within it. Here's an example of the output of the "D" command, in FIgure 5-20-5 below:



```
---z/XPF---------------CSECT WITHIN LOAD MODULE DISPLAY----------- Row 1 of 17
OPTION   ===>                                                  SCROLL ===> PAGE

     Source DSN = CSW.QA.SYMLINKE
     Load module = FHCTEST

                          PF3/END TO EXIT

 CSECT NAME                  BEGIN      END        LENGTH
                             OFFSET     OFFSET
 FHCTEST#C
                            00000000   00003254   00003254
 CEESTART
                            00003258   000032D4   0000007C
 SUBROUTE#C
                            000032D8   0000350C   00000234
 AXDCHOOK
                            000036E8   00003C28   00000540
 CEEROOTA
                            00003C28   00003E18   000001F0
 CEEBLLST
                            00003E18   00003E74   0000005C
 CEEBETBL
                            00003E78   00003EA0   00000028
 EDCINPL
                            00003EA0   00003EC4   00000024
 CEESG003
                            00003EC8   00003FF3   0000012B
 CEEBPUBT
                            00003FF8   00004068   00000070
 CEEBTRM
                            00004068   0000410C   000000A4
 CEEBINT
                            00004110   00004118   00000008
 CEEARLU
                            00004118   000041D0   000000B8
 CEEBPIRA
                            000041D0   00004470   000002A0
 CEECPYRT
                            00004470   00004552   000000E2
 CEEROND
                            00004558   00004574   0000001C
```

Figure 5-20-5.

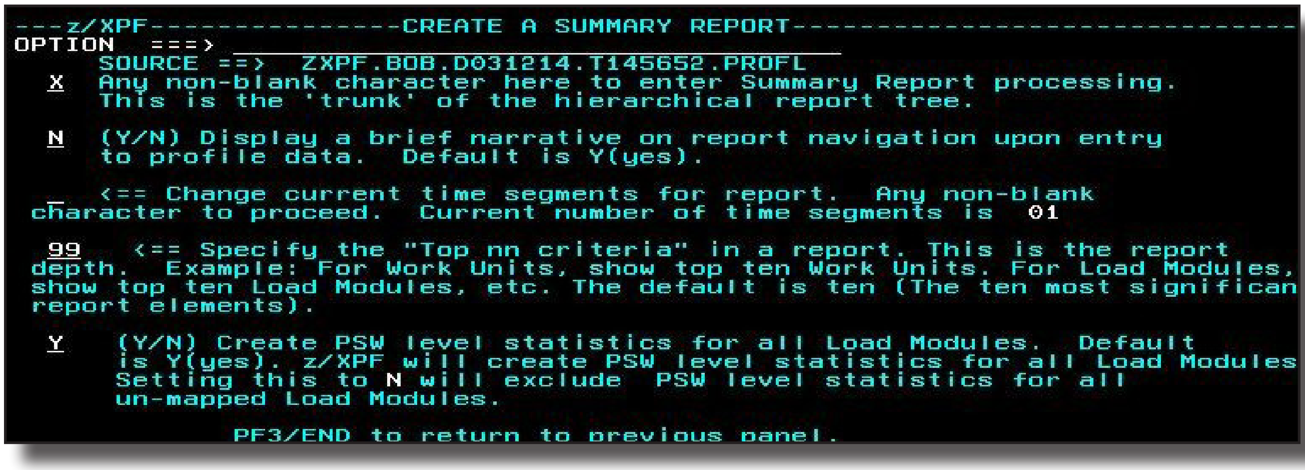
Now I'll use PF3 to navigate back to the Primary Create Profile Menu. I'll pick Option 6 to look at Summary Reports. See Figure 5-20-6 below:



```
---z/XPF---------------------PRIMARY CREATE PROFILE MENU --------------------
OPTION   ===> 6
                          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. Same reports as
       option 6 above, but can set Time Segments 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 5-20-6.

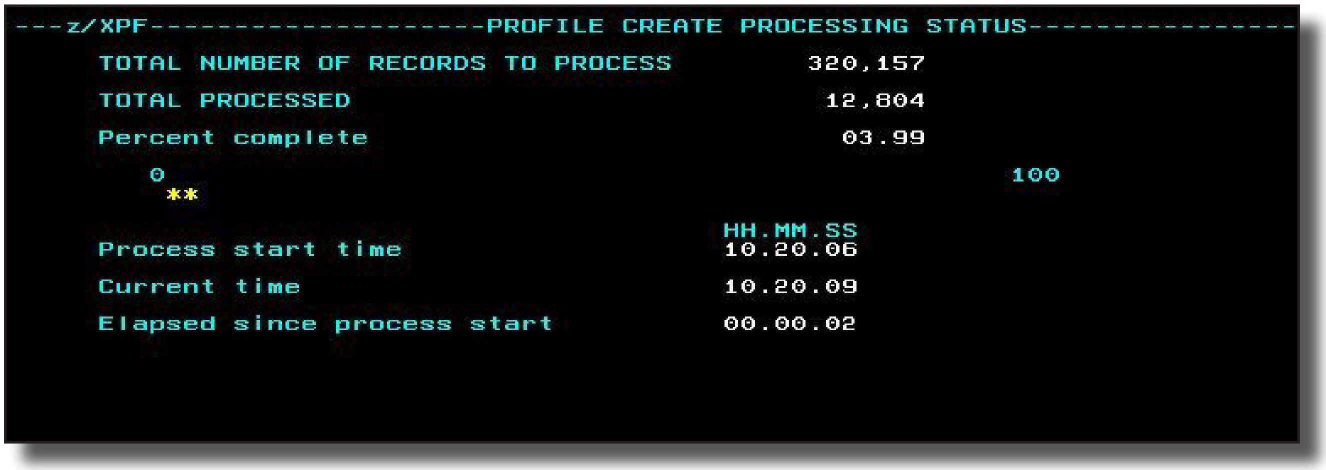
I enter any old character in the top field below the command line. Today I used "X" for the heck of it. See Figure 5-20-7 below:



```
---z/XPF----------------CREATE A SUMMARY REPORT------------------------------
OPTION   ===> ____________________________________
     SOURCE ==>  ZXPF.BOB.D031214.T145652.PROFL
  X  Any non-blank character here to enter Summary Report processing.
     This is the 'trunk' of the hierarchical report tree.

  N  (Y/N) Display a brief narrative on report navigation upon entry
     to profile data.  Default is Y(yes).

  _  <== Change current time segments for report.  Any non-blank
 character to proceed.  Current number of time segments is  01

 99   <== Specify the "Top nn criteria" in a report. This is the report
 depth.  Example: For Work Units, show top ten Work Units. For Load Modules,
 show top ten Load Modules, etc. The default is ten (The ten most significan
 report elements).

  Y    (Y/N) Create PSW level statistics for all Load Modules.  Default
       is Y(yes). z/XPF will create PSW level statistics for all Load Modules
       Setting this to N will exclude  PSW level statistics for all
       un-mapped Load Modules.

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

Figure 5-20-7.

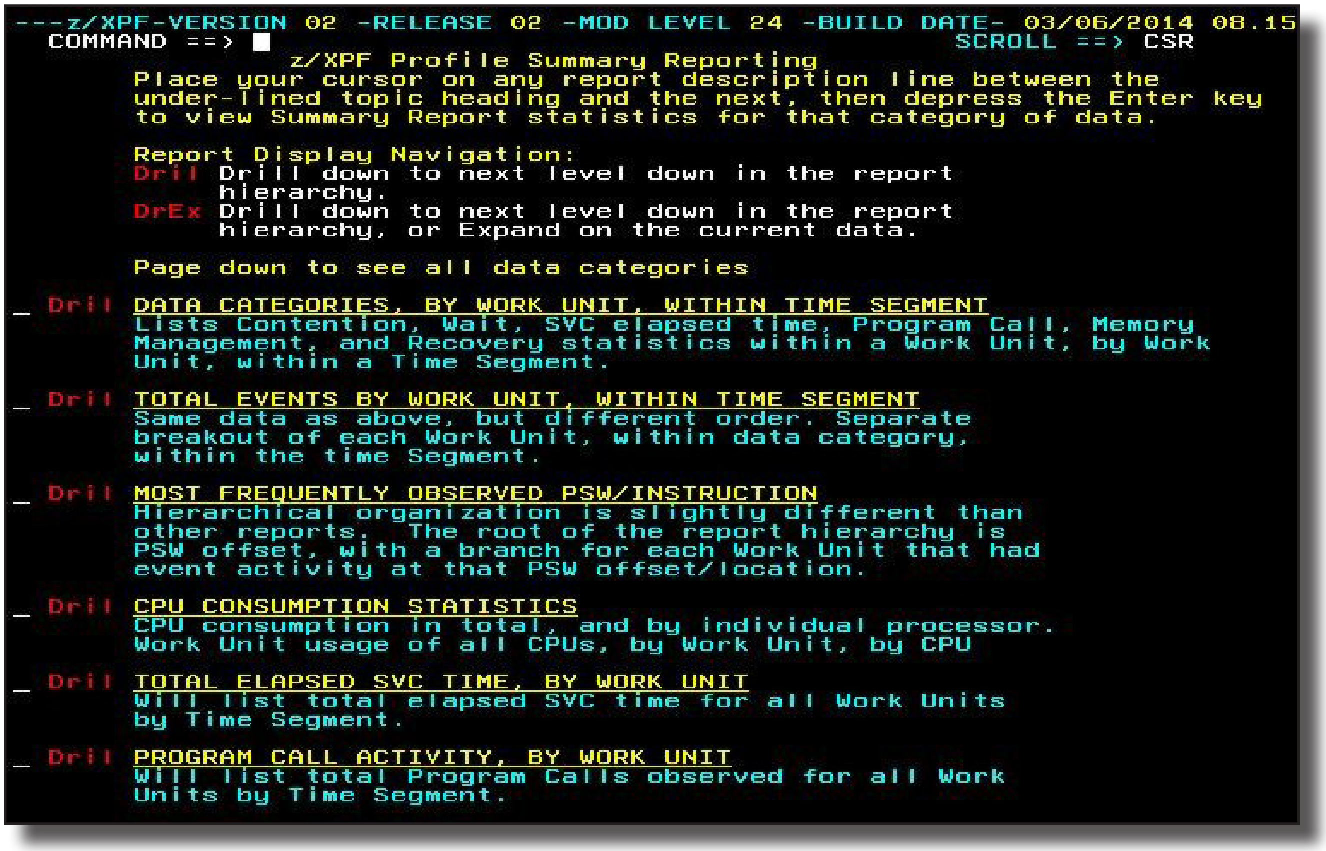
z/XPF goes to work parsing the report. The line of yellow asterisks cruises across the page to completion. If your report is a large one (one million records or more), you may as well find something else to do. See Figure 5-20-8 below:



```
---z/XPF--------------------PROFILE CREATE PROCESSING STATUS----------------
     TOTAL NUMBER OF RECORDS TO PROCESS          320,157

     TOTAL PROCESSED                              12,804

     Percent complete                              03.99

        0                                                     100
         **

                                             HH.MM.SS
     Process start time                      10.20.06

     Current time                            10.20.09

     Elapsed since process start             00.00.02
```

Figure 5-20-8.

Now,we arrive at the top panel in our Summary Report. See Figure 5-20-9 below:

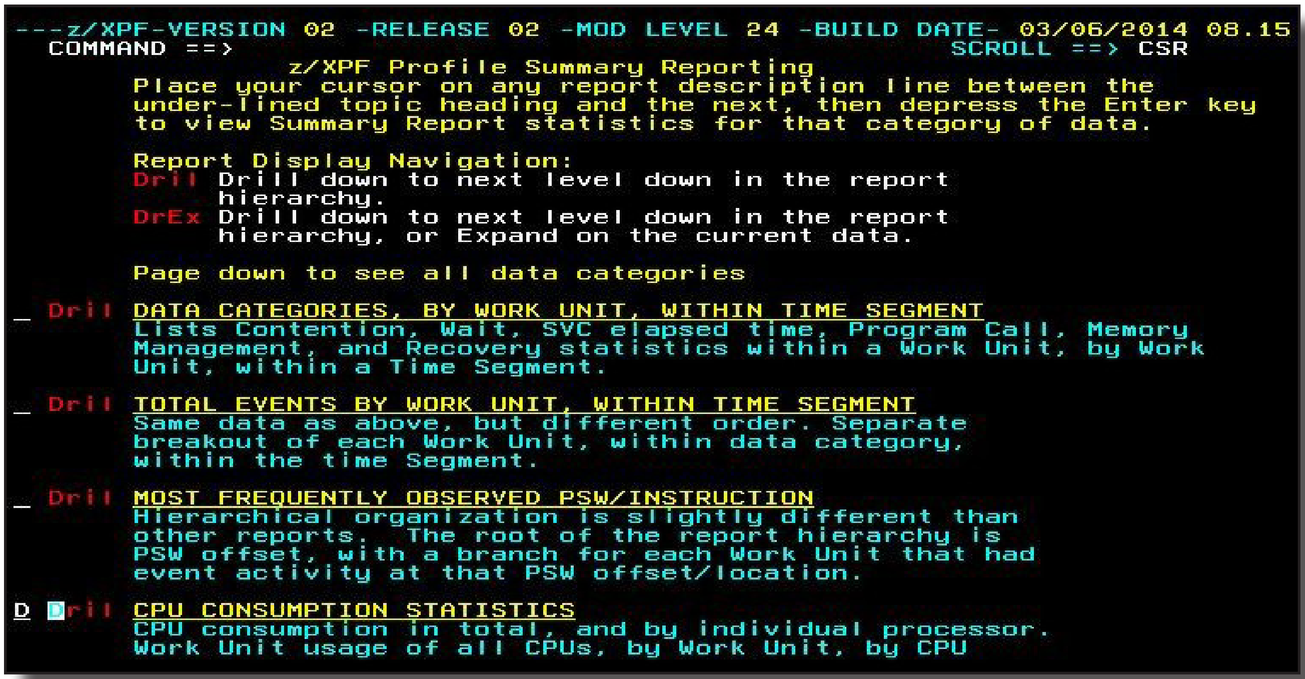


```
---z/XPF-VERSION 02 -RELEASE 02 -MOD LEVEL 24 -BUILD DATE- 03/06/2014 08.15
  COMMAND ==> █                                             SCROLL ==> CSR
                      z/XPF Profile Summary Reporting
       Place your cursor on any report description line between the
       under-lined topic 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 CPU CONSUMPTION STATISTICS
       CPU consumption in total, and by individual processor.
       Work Unit usage of all CPUs, by Work Unit, by CPU

_ 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.
```

Figure 5-20-9.

Now for the fun part!

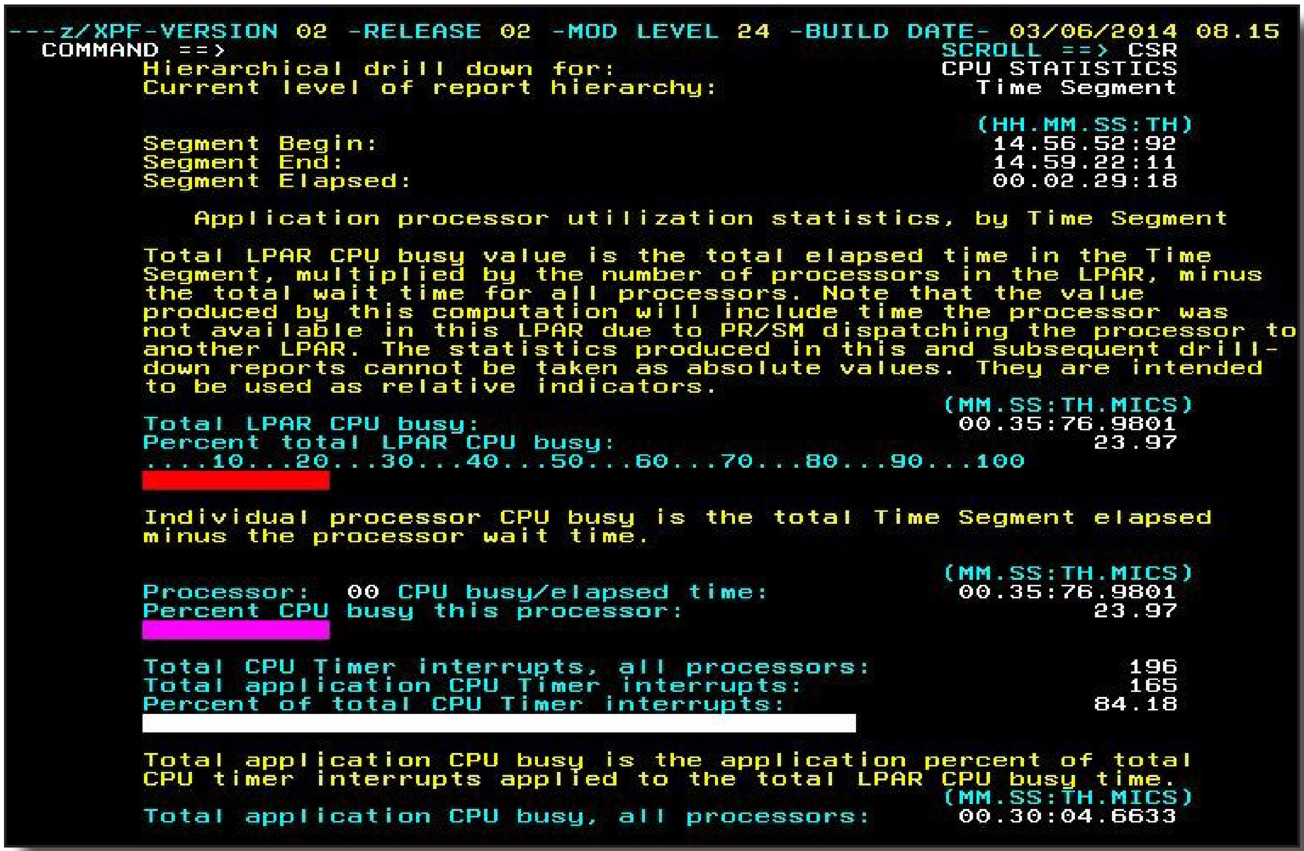
In this example, the program FHCTEST really doesn't "do" much. What I've done here is run z/XDC against FHCTEST, which means that z/XDC actually did almost all the work and minimal time was spent executing FHCTEST. I'll drill into CPU Consumption Statistics. See Figure 5-20-10 below:



```
---z/XPF-VERSION 02 -RELEASE 02 -MOD LEVEL 24 -BUILD DATE- 03/06/2014 08.15
   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.

D  Dril CPU CONSUMPTION STATISTICS
        CPU consumption in total, and by individual processor.
        Work Unit usage of all CPUs, by Work Unit, by CPU
```

Figure 5-20-10.

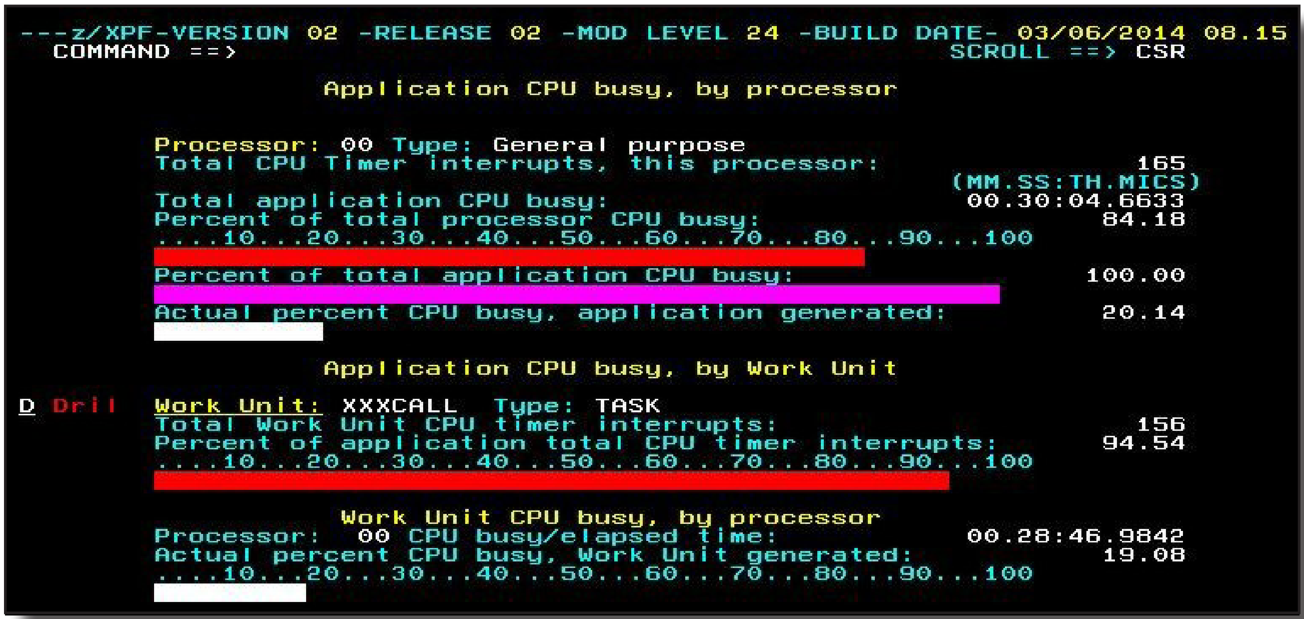
And we arrive here. z/XPF begins by telling me information on the LPAR and CPU busy statistics. See Figure 5-20-11 below:



```
---z/XPF-VERSION 02 -RELEASE 02 -MOD LEVEL 24 -BUILD DATE- 03/06/2014 08.15
 COMMAND ==>                                                  SCROLL ==> CSR
       Hierarchical drill down for:                           CPU STATISTICS
       Current level of report hierarchy:                       Time Segment

                                                              (HH.MM.SS:TH)
       Segment Begin:                                          14.56.52:92
       Segment End:                                            14.59.22:11
       Segment Elapsed:                                        00.02.29:18

          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. Note that the value
       produced by this computation will include time the processor was
       not available in this LPAR due to PR/SM dispatching the processor to
       another LPAR. The statistics produced in this and subsequent drill-
       down reports cannot be taken as absolute values. They are intended
       to be used as relative indicators.
                                                             (MM.SS:TH.MICS)
       Total LPAR CPU busy:                                   00.35:76.9801
       Percent total LPAR CPU busy:                                   23.97
       ....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:                  00.35:76.9801
       Percent CPU busy this processor:                               23.97


       Total CPU Timer interrupts, all processors:                      196
       Total application CPU Timer interrupts:                          165
       Percent of total CPU Timer interrupts:                         84.18


       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:            00.30:04.6633
```

Figure 5-20-11.

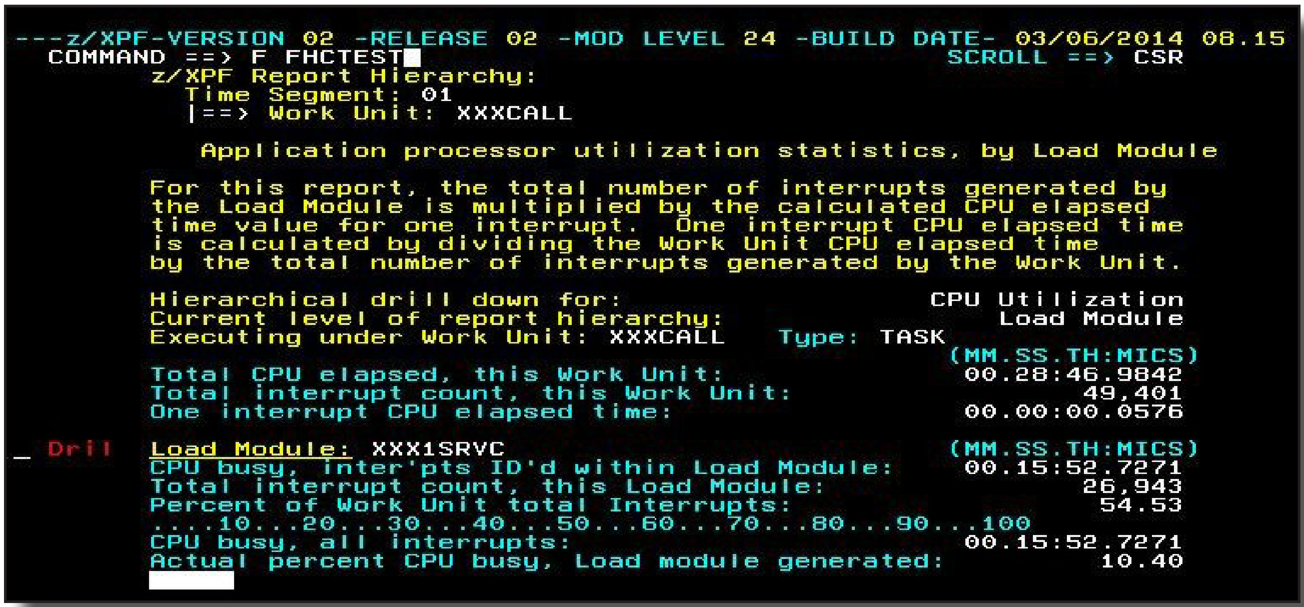
However, the panel above isn't what I'm interested in now. So, I page down to the Work Unit "XXXCALL". This is where I'll eventually find FHCTEST. See Figure 5-20-12 below. I drill here...



```
---z/XPF-VERSION 02 -RELEASE 02 -MOD LEVEL 24 -BUILD DATE- 03/06/2014 08.15
   COMMAND ==>                                                SCROLL ==> CSR

                      Application CPU busy, by processor

           Processor: 00 Type: General purpose
           Total CPU Timer interrupts, this processor:                  165
                                                               (MM.SS:TH.MICS)
           Total application CPU busy:                          00.30:04.6633
           Percent of total processor CPU busy:                         84.18
           ....10...20...30...40...50...60...70...80...90...100

           Percent of total application CPU busy:                      100.00

           Actual percent CPU busy, application generated:              20.14


                      Application CPU busy, by Work Unit

D Dril     Work Unit: XXXCALL    Type: TASK
           Total Work Unit CPU timer interrupts:                        156
           Percent of application total CPU timer interrupts:           94.54
           ....10...20...30...40...50...60...70...80...90...100


                        Work Unit CPU busy, by processor
           Processor:  00 CPU busy/elapsed time:                00.28:46.9842
           Actual percent CPU busy, Work Unit generated:                19.08
           ....10...20...30...40...50...60...70...80...90...100
```

Figure 5-20-12.

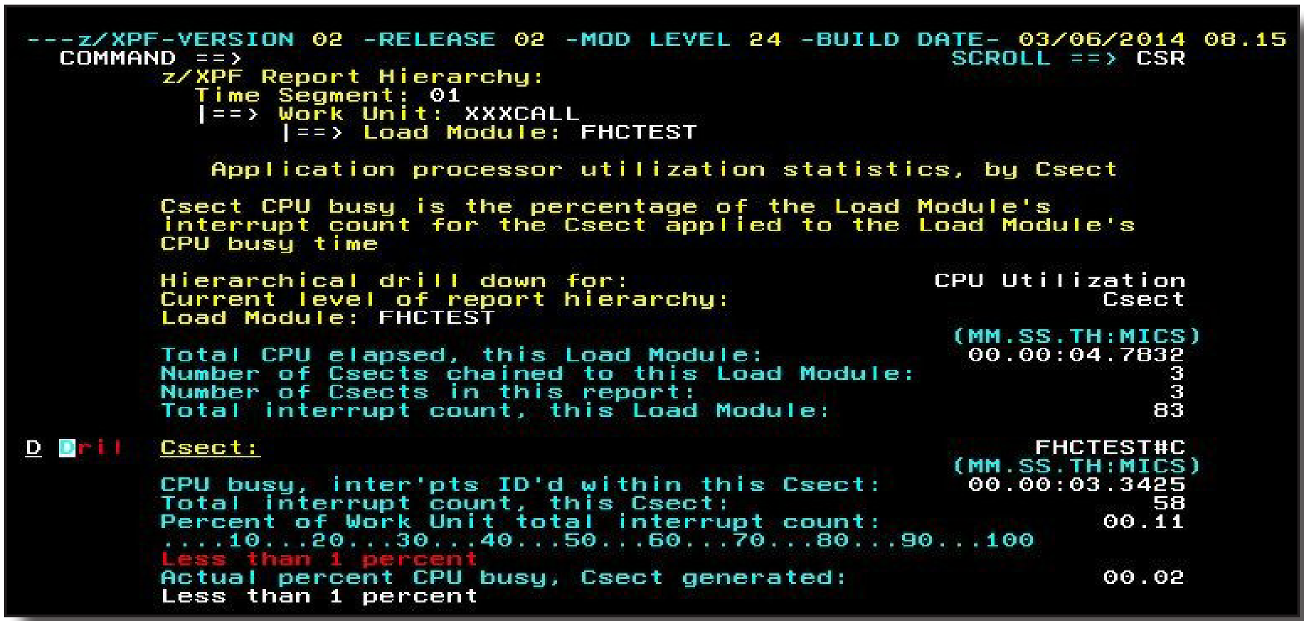
And I arrive at the display of the load modules that were part of the Work Unit. I issue a "FIND" for FHCTEST. See Figure 5-20-13:



```
---z/XPF-VERSION 02 -RELEASE 02 -MOD LEVEL 24 -BUILD DATE- 03/06/2014 08.15
   COMMAND ==> F FHCTEST                                      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:                   00.28:46.9842
           Total interrupt count, this Work Unit:                      49,401
           One interrupt CPU elapsed time:                      00.00:00.0576

_ Dril     Load Module: XXX1SRVC                               (MM.SS.TH:MICS)
           CPU busy, inter'pts ID'd within Load Module:         00.15:52.7271
           Total interrupt count, this Load Module:                    26,943
           Percent of Work Unit total Interrupts:                       54.53
           ....10...20...30...40...50...60...70...80...90...100
           CPU busy, all interrupts:                            00.15:52.7271
           Actual percent CPU busy, Load module generated:              10.40
```

Figure 5-20-13.

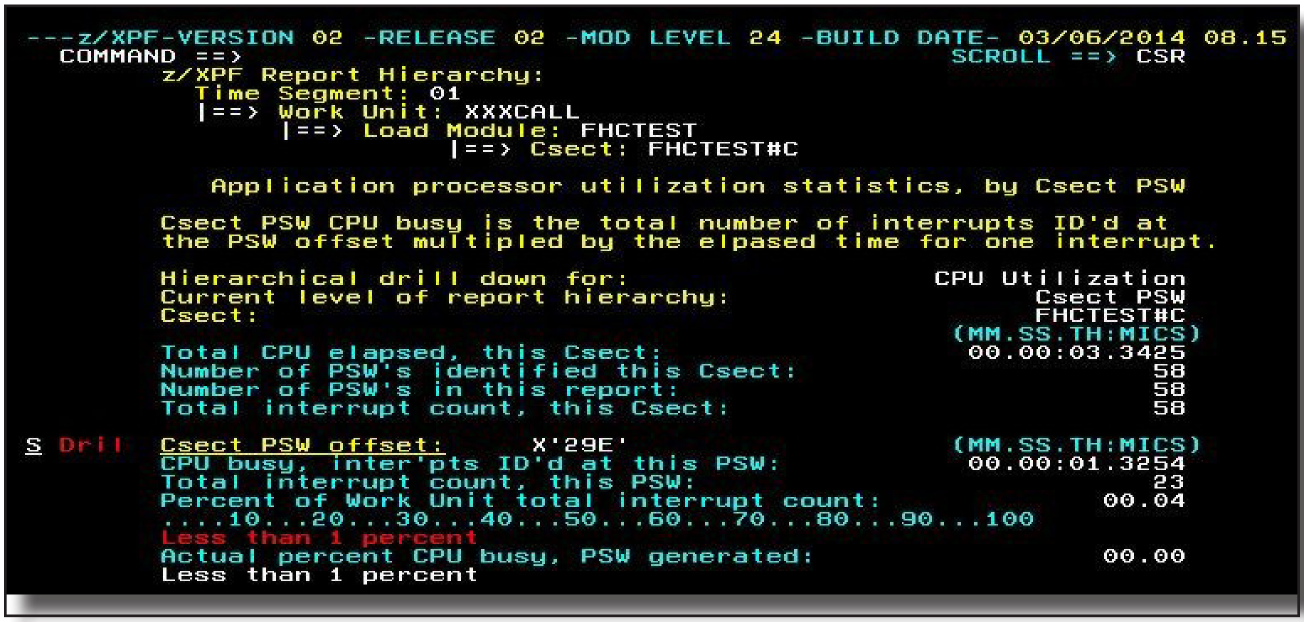
And there it is. You can tell that FHCTEST wasn't doing very much, can't you? See that I am "drilling" into the load module FHCTEST, in Figure 5-20-14 below:



```
---z/XPF-VERSION 02 -RELEASE 02 -MOD LEVEL 24 -BUILD DATE- 03/06/2014 08.15
   COMMAND ==>                                                 SCROLL ==> CSR
         z/XPF Report Hierarchy:
            Time Segment: 01
            |==> Work Unit: XXXCALL
                   |==> Load Module: FHCTEST

              Application processor utilization statistics, by Csect

         Csect CPU busy is the percentage of the Load Module's
         interrupt count for the Csect applied to the Load Module's
         CPU busy time

         Hierarchical drill down for:                         CPU Utilization
         Current level of report hierarchy:                             Csect
         Load Module: FHCTEST
                                                             (MM.SS.TH:MICS)
         Total CPU elapsed, this Load Module:                  00.00:04.7832
         Number of Csects chained to this Load Module:                     3
         Number of Csects in this report:                                  3
         Total interrupt count, this Load Module:                         83

 D Dril  Csect:                                                    FHCTEST#C
                                                             (MM.SS.TH:MICS)
         CPU busy, inter'pts ID'd within this Csect:           00.00:03.3425
         Total interrupt count, this Csect:                               58
         Percent of Work Unit total interrupt count:                   00.11
         ....10...20...30...40...50...60...70...80...90...100
         Less than 1 percent
         Actual percent CPU busy, Csect generated:                     00.02
         Less than 1 percent
```

Figure 5-20-14.

I arrive at the display of all csects within the FHCTEST module. I pick one at random, and drill into it: See Figure 5-20-15 below:

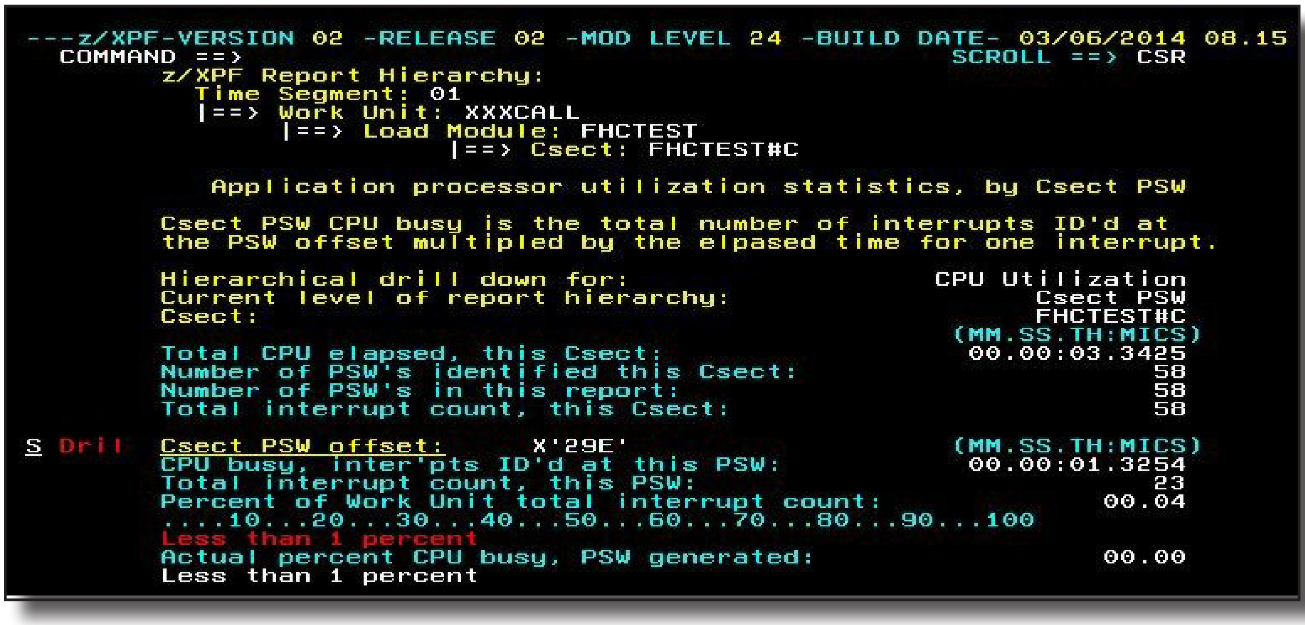


```
---z/XPF-VERSION 02 -RELEASE 02 -MOD LEVEL 24 -BUILD DATE- 03/06/2014 08.15
   COMMAND ==>                                                 SCROLL ==> CSR
         z/XPF Report Hierarchy:
            Time Segment: 01
            |==> Work Unit: XXXCALL
                   |==> Load Module: FHCTEST
                             |==> Csect: FHCTEST#C

              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:                         Csect PSW
         Csect:                                                    FHCTEST#C
                                                             (MM.SS.TH:MICS)
         Total CPU elapsed, this Csect:                        00.00:03.3425
         Number of PSW's identified this Csect:                           58
         Number of PSW's in this report:                                  58
         Total interrupt count, this Csect:                               58

 S Dril  Csect PSW offset:     X'29E'                        (MM.SS.TH:MICS)
         CPU busy, inter'pts ID'd at this PSW:                 00.00:01.3254
         Total interrupt count, this PSW:                                 23
         Percent of Work Unit total interrupt count:                   00.04
         ....10...20...30...40...50...60...70...80...90...100
         Less than 1 percent
         Actual percent CPU busy, PSW generated:                       00.00
         Less than 1 percent
```

Figure 5-20-15.

Here is the display of PSWs within the FHCTEST#C Csect. It's nice that z/XPF "tracks" the depth of your inquiry at the top of the panel, isn't it?

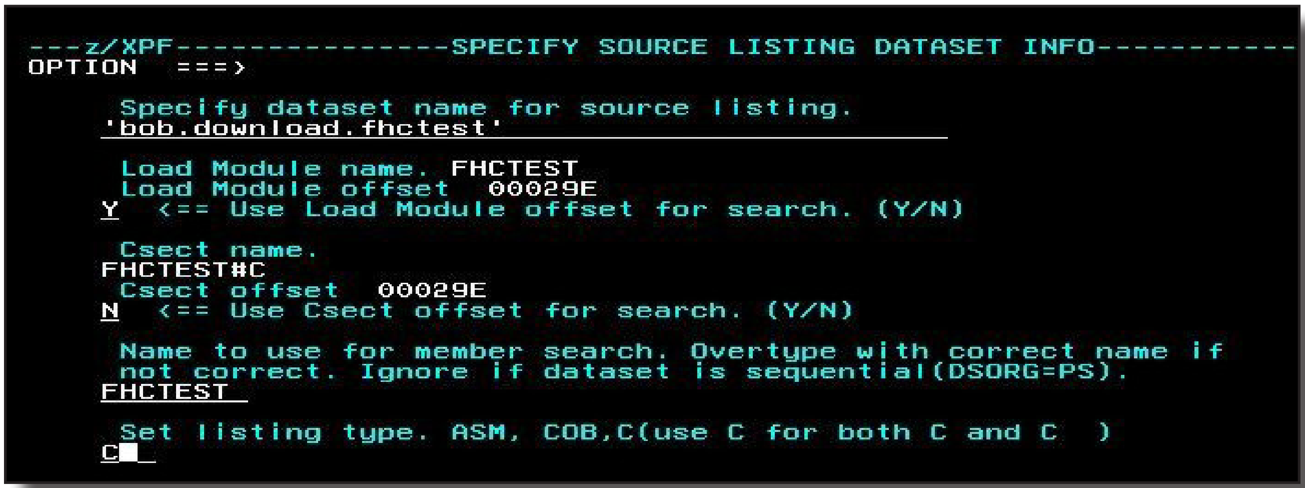
At this point I can invoke the SOURCE command (which will have the effect of showing you the PSW that appears first in the display). I can issue the SOURCE command or I can put an "S" next to any Csect PSW offset. See Figure 5-20-16 below:



```
---z/XPF-VERSION 02 -RELEASE 02 -MOD LEVEL 24 -BUILD DATE- 03/06/2014 08.15
   COMMAND ==>                                                  SCROLL ==> CSR
           z/XPF Report Hierarchy:
              Time Segment: 01
              |==> Work Unit: XXXCALL
                     |==> Load Module: FHCTEST
                                 |==> Csect: FHCTEST#C

                 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:                      Csect PSW
           Csect:                                                  FHCTEST#C
                                                            (MM.SS.TH:MICS)
           Total CPU elapsed, this Csect:                    00.00:03.3425
           Number of PSW's identified this Csect:                       58
           Number of PSW's in this report:                              58
           Total interrupt count, this Csect:                           58

 S Drill   Csect PSW offset:       X'29E'                   (MM.SS.TH:MICS)
           CPU busy, inter'pts ID'd at this PSW:             00.00:01.3254
           Total interrupt count, this PSW:                             23
           Percent of Work Unit total interrupt count:               00.04
           ....10...20...30...40...50...60...70...80...90...100
           Less than 1 percent
           Actual percent CPU busy, PSW generated:                   00.00
           Less than 1 percent
```

Figure 5-20-16.

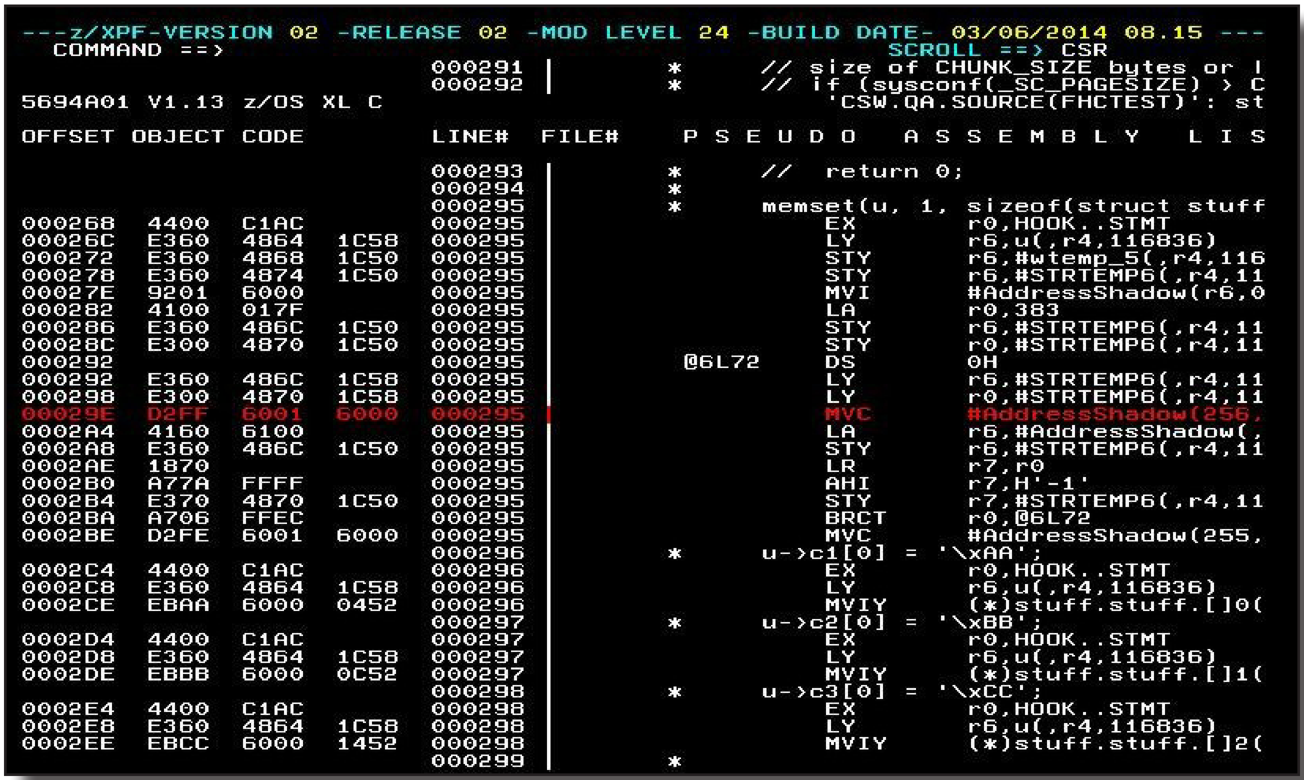
Now I have to tell z/XPF where the listing can be found, and what language z/XPF should expect to find. You'll only have to do this one time for each csect within a load module map, because once you associate a listing with your csect, z/XPF "remembers" that association and won't prompt you for this information the next time you access the report. On the other hand, if the same csect is linked into other modules, then z/XPF WILL prompt you the next time you try to see the source code. Sere Figure 5-20-17 below:



```
---z/XPF---------------SPECIFY SOURCE LISTING DATASET INFO-----------
OPTION   ===>

      Specify dataset name for source listing.
     'bob.download.fhctest'

      Load Module name. FHCTEST
      Load Module offset   00029E
     Y  <== Use Load Module offset for search. (Y/N)

      Csect name.
     FHCTEST#C
      Csect offset   00029E
     N  <== Use Csect offset for search. (Y/N)

      Name to use for member search. Overtype with correct name if
      not correct. Ignore if dataset is sequential(DSORG=PS).
     FHCTEST

      Set listing type. ASM, COB,C(use C for both C and C  )
     C
```

Figure 5-20-17.

Voila! You see the source code! The PSW that I put the "S" next to in the Summary Report panel is displayed in red, in Figure 5-20-18 below:



```
---z/XPF-VERSION 02 -RELEASE 02 -MOD LEVEL 24 -BUILD DATE- 03/06/2014 08.15 ---
  COMMAND ==>                                           SCROLL ==> CSR
                          000291 |       *    // size of CHUNK_SIZE bytes or |
                          000292 |       *    // if (sysconf(_SC_PAGESIZE) > C
5694A01 V1.13 z/OS XL C                                  'CSW.QA.SOURCE(FHCTEST)': st

OFFSET OBJECT CODE        LINE#  FILE#    P S E U D O   A S S E M B L Y   L I S

                          000293 |       *    //  return 0;
                          000294 |       *
                          000295 |       *    memset(u, 1, sizeof(struct stuff
000268  4400  C1AC        000295 |                EX       r0,HOOK..STMT
00026C  E360  4864  1C58  000295 |                LY       r6,u(,r4,116836)
000272  E360  4868  1C50  000295 |                STY      r6,#wtemp_5(,r4,116
000278  E360  4874  1C50  000295 |                STY      r6,#STRTEMP6(,r4,11
00027E  9201  6000        000295 |                MVI      #AddressShadow(r6,0
000282  4100  017F        000295 |                LA       r0,383
000286  E360  486C  1C50  000295 |                STY      r6,#STRTEMP6(,r4,11
00028C  E300  4870  1C50  000295 |                STY      r0,#STRTEMP6(,r4,11
000292                    000295 |        @6L72   DS       0H
000292  E360  486C  1C58  000295 |                LY       r6,#STRTEMP6(,r4,11
000298  E300  4870  1C58  000295 |                LY       r0,#STRTEMP6(,r4,11
00029E  D2FF  6001  6000  000295 |                MVC      #AddressShadow(256,
0002A4  4160  6100        000295 |                LA       r6,#AddressShadow(,
0002A8  E360  486C  1C50  000295 |                STY      r6,#STRTEMP6(,r4,11
0002AE  1870              000295 |                LR       r7,r0
0002B0  A77A  FFFF        000295 |                AHI      r7,H'-1'
0002B4  E370  4870  1C50  000295 |                STY      r7,#STRTEMP6(,r4,11
0002BA  A706  FFEC        000295 |                BRCT     r0,@6L72
0002BE  D2FE  6001  6000  000295 |                MVC      #AddressShadow(255,
                          000296 |       *    u->c1[0] = '\xAA';
0002C4  4400  C1AC        000296 |                EX       r0,HOOK..STMT
0002C8  E360  4864  1C58  000296 |                LY       r6,u(,r4,116836)
0002CE  EBAA  6000  0452  000296 |                MVIY     (*)stuff.stuff.[]0(
                          000297 |       *    u->c2[0] = '\xBB';
0002D4  4400  C1AC        000297 |                EX       r0,HOOK..STMT
0002D8  E360  4864  1C58  000297 |                LY       r6,u(,r4,116836)
0002DE  EBBB  6000  0C52  000297 |                MVIY     (*)stuff.stuff.[]1(
                          000298 |       *    u->c3[0] = '\xCC';
0002E4  4400  C1AC        000298 |                EX       r0,HOOK..STMT
0002E8  E360  4864  1C58  000298 |                LY       r6,u(,r4,116836)
0002EE  EBCC  6000  1452  000298 |                MVIY     (*)stuff.stuff.[]2(
                          000299 |       *
```

Figure 5-20-18.

You are looking both at the source statements, and the object code that executes on the behalf of each statement. Pretty cool!

- You can page up or down in the display.
- If you wish to return to your PSW, you issue "PSW" on the command line, and z/XPF takes you back to that PSW.
- You can also issue PF3 to "back out", and explore another PSW if you like.

There you have it: z/XPF allows you to see your source code!

Chapter 6 - Creating DB2-Specific Summary Reports

z/XPF is able to profile DB2 address spaces. This sub-section of the book will discuss how to do that, and what you'll see in z/XPF's Summary Reports.

Here is some program logic information on DB2 and z/XPF:

During z/XPF server address space initialization, all defined DB2 systems on the z/OS image are identified. After identification of the DB2 sub-system name, a check is made to determine which, if any, are active. For those active, all of the address spaces associated with the sub-system are identified. z/XPF refers to these as "address spaces of interest". In addition, for each active DB2 system, a copy of the MEPL is made.

z/XPF's SMF exits for exit points IEFUSI and IEFACTRT (established during initialization) inform the z/XPF address space when DB2 address spaces initiate and terminate.

During data capture, z/XPF checks the event to determine if the event occurred while executing DB2 code. The PASN value for the event is checked against the list of known DB2 address space ids if the PASN is not equal to the HASN.

Part of the early processing logic for an SQL statement executes a space-switch Program Call that makes the PASN value equal to the ASID value of the DB2 DBM1 sub-system address space.

z/XPF marks the event as a DB2 event when either the PASN value for the event is equal to one of the DB2 sub-system address space ids OR when the load module is identified and the first three characters of the load module name begin with "DSN".

If either one of these conditions is met, a look inside the target profile address space is made to locate connection information for the Work Unit. If the Work Unit is connected to DB2, a further check is made to identify the DBRM and DBRM Section currently active on the connection.

DB2 activity is summarized within z/XPF first by DBRM and Section, and then by dynamic SQL text, if dynamic SQL was observed during data capture. A breakout by Work Unit within DBRM is given in the Segment Summary report. This view informs the user which DBRMs had the most activity across all Work Units, and which Work Units were active using the DBRM.

The same statistics are available for SQL text. SQL text Activity is reported by the text statement, and then by Work Unit within text statement.

6-1 How z/XPF Uses DB2 Catalog Information

During data capture, z/XPF identifies DBRMs that were used to access DB2 by the target application. At data capture termination, z/XPF does the following for each DBRM it identified:

Using the DBRM name, DB2 catalog table SYSIBM.SYSPACKAGE is queried to determine if the DBRM is bound as a DB2 package.

When the DBRM is found in SYSIBM.SYSPACKAGE, the following fields are pulled from the table:

- COLLID
- OWNER
- TIMESTAMP
- BINDTIME
- QUALIFIER
- ISOLATION
- EXPLAIN
- REOPTVAR
- TYPE
- PDSNAME
- PCTIMESTAMP

The SYSIBM.SYSPACKDEP table records the dependencies of packages on local tables, views, synonyms, table spaces, indexes, aliases, functions, and stored procedures.

Also, SYSIBM.SYSPACKDEP is queried for the following :

- BNAME
- BQUALIFIER
- BTYPE

When no entry is found in SYSIBM.SYSPACKAGE, SYSIBM.SYSPLAN is queried using the DBRM name.

The SYSIBM.SYSPLANDEP table records the dependencies of plans on tables, views, aliases, synonyms, table spaces, indexes, functions, and stored procedures.

SYSIBM.SYSPLAN is queried for the following:

- CREATOR
- ISOLATION
- EXPLAN
- QUALIFIER
- BOUNDTS
- REOPTVAR

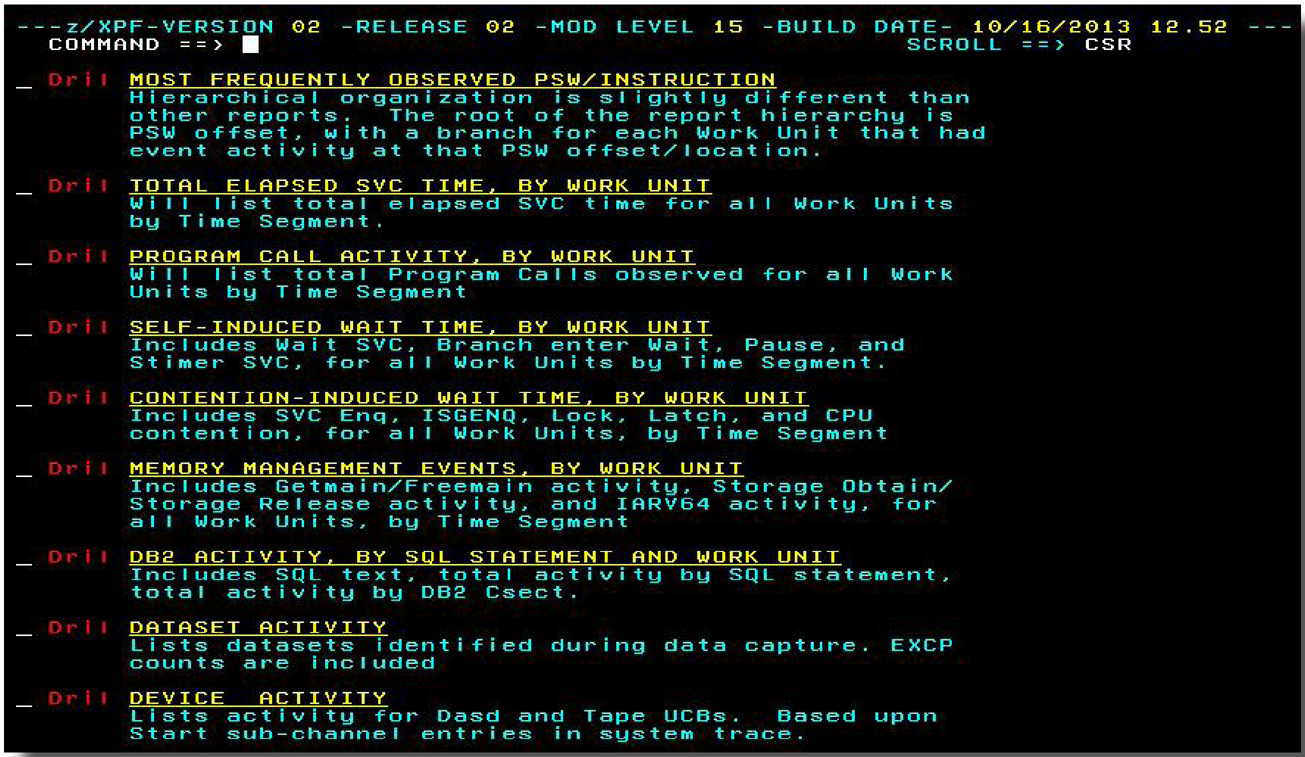
Also, SYSIBM.SYSPLANDEP is queried for:

- BNAME
- BCREATOR
- BTYPE

For more information on DB2, you can consult the IBM Manual, “The DB2 SQL Reference Manual (SC19-2983)”.

6-2 Accessing z/XPF’s DB2 Summary Reports.

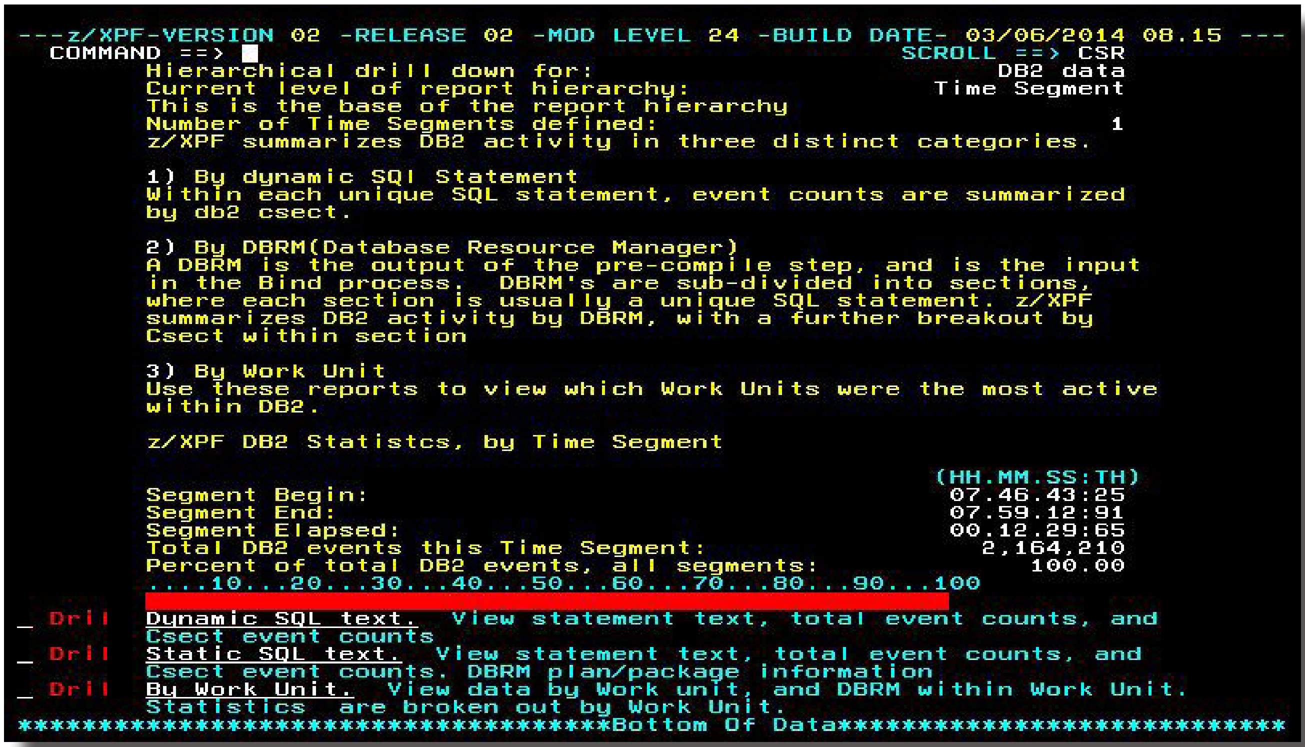
When you have created a z/XPF data capture dataset for Summary reporting, you can page down to the topic, “DB2 ACTIVITY, BY SQL STATEMENT AND WORK UNIT”. Here is a picture of that “tab” in the report. I’m ready to press Enter to look at the report. See Figure 6-1 below:



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

_  Drill  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.

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

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

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

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

_  Drill  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

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

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

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

Figure 6-2-1.

So, let’s drill into DB2 Activity. I press Enter and see Figure 6-2-2 below:

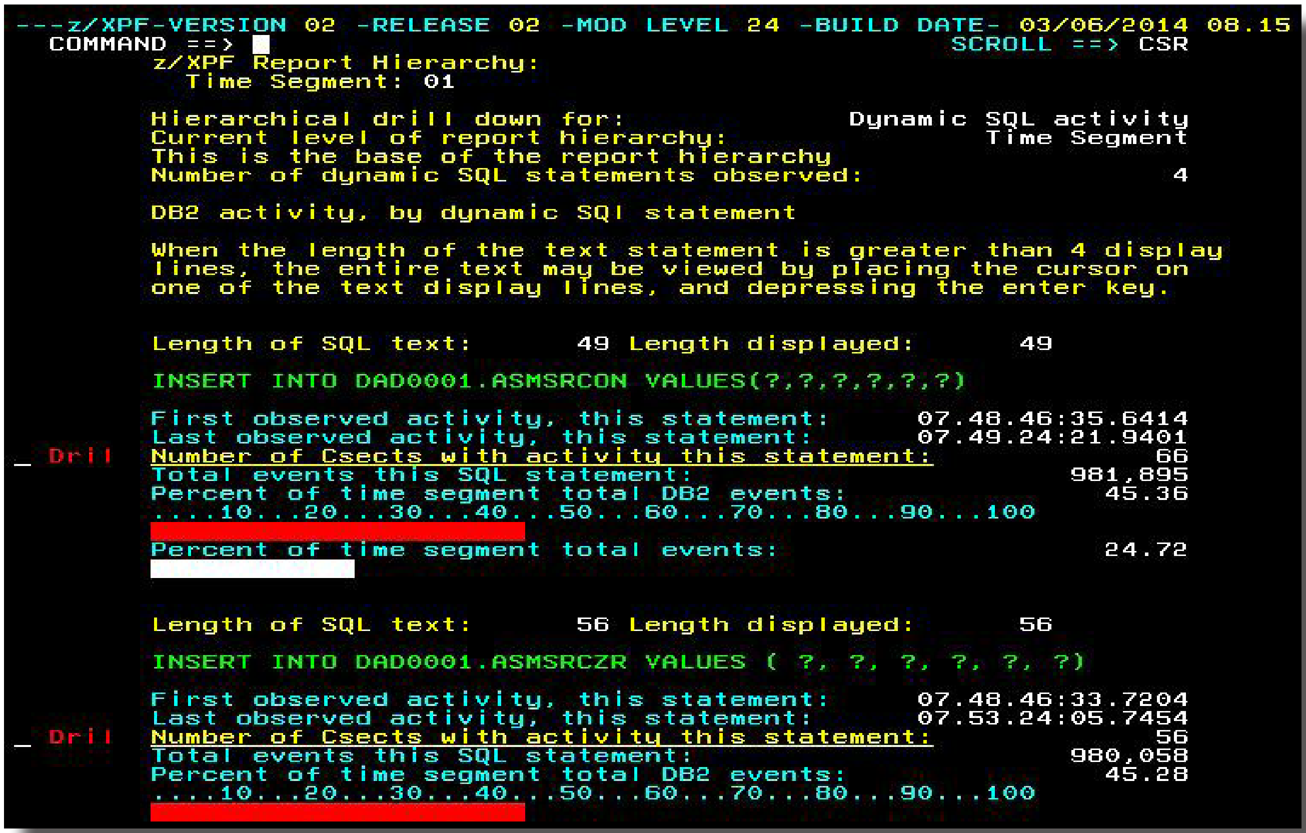


```
---z/XPF-VERSION 02 -RELEASE 02 -MOD LEVEL 24 -BUILD DATE- 03/06/2014 08.15 ---
  COMMAND ==>                                                 SCROLL ==> CSR
        Hierarchical drill down for:                                DB2 data
        Current level of report hierarchy:                      Time Segment
        This is the base of the report hierarchy
        Number of Time Segments defined:                                   1
        z/XPF summarizes DB2 activity in three distinct categories.

        1) By dynamic SQl Statement
        Within each unique SQL statement, event counts are summarized
        by db2 csect.

        2) By DBRM(Database Resource Manager)
        A DBRM is the output of the pre-compile step, and is the input
        in the Bind process.  DBRM's are sub-divided into sections,
        where each section is usually a unique SQL statement. z/XPF
        summarizes DB2 activity by DBRM, with a further breakout by
        Csect within section

        3) By Work Unit
        Use these reports to view which Work Units were the most active
        within DB2.

        z/XPF DB2 Statistcs, by Time Segment

                                                             (HH.MM.SS:TH)
        Segment Begin:                                        07.46.43:25
        Segment End:                                          07.59.12:91
        Segment Elapsed:                                      00.12.29:65
        Total DB2 events this Time Segment:                     2,164,210
        Percent of total DB2 events, all segments:                 100.00
        ....10...20...30...40...50...60...70...80...90...100

_ Drill Dynamic SQL text.  View statement text, total event counts, and
        Csect event counts
_ Drill Static SQL text.  View statement text, total event counts, and
        Csect event counts. DBRM plan/package information
_ Drill By Work Unit.  View data by Work unit, and DBRM within Work Unit.
        Statistics  are broken out by Work Unit.
*************************************Bottom Of Data*****************************
```

Figure 6-2-2.

There are three actions I can perform here. I can see dynamic SQL statements, static SQL statements and look at DB2's activity by the Work Unit.

First things first, I suppose. Let's pick Dynamic SQL. I'll tab to the first "DrEx" area on the screen and press Enter (for a drill operation). Now I see Figure 6-2-3, below:

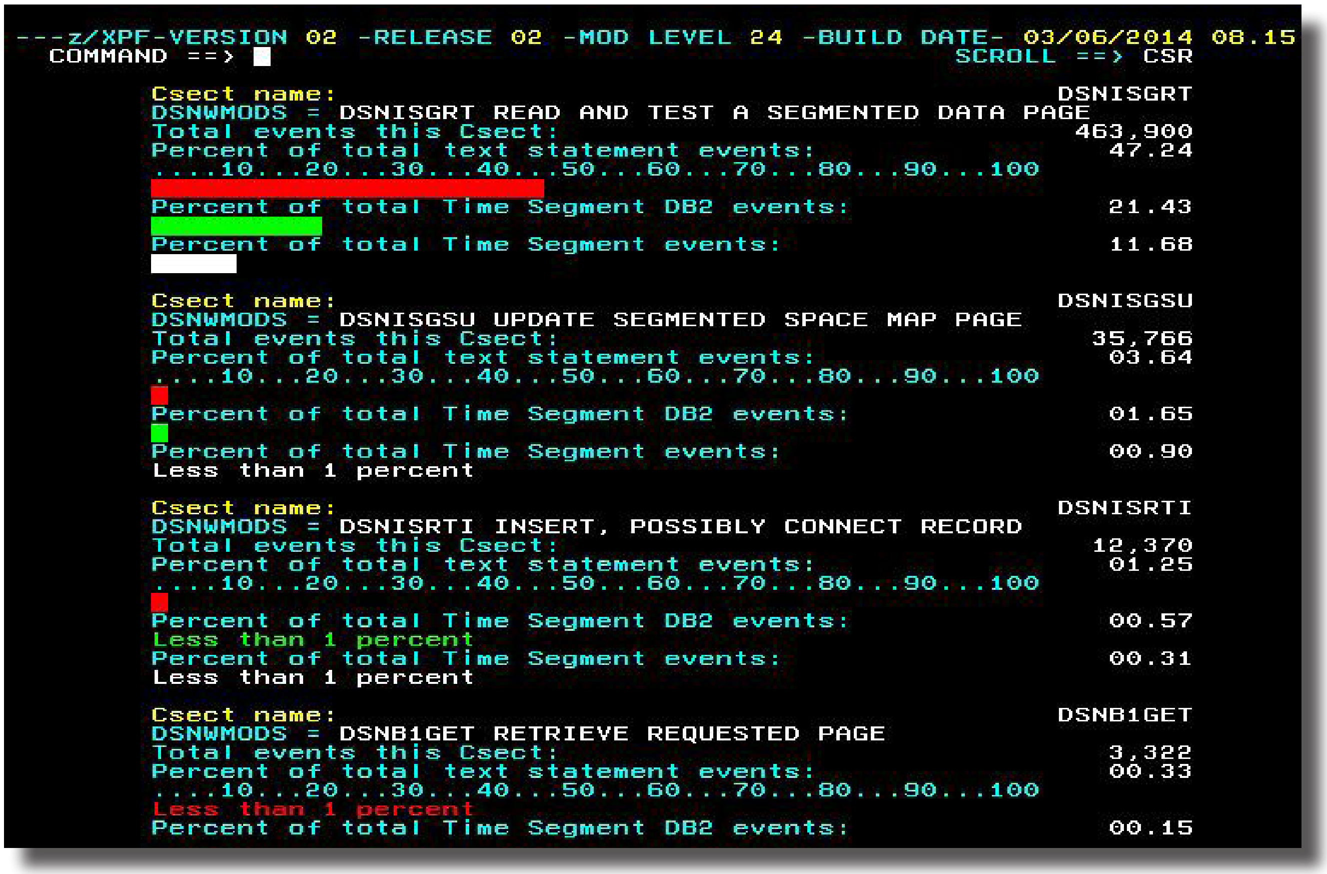


```
---z/XPF-VERSION 02 -RELEASE 02 -MOD LEVEL 24 -BUILD DATE- 03/06/2014 08.15
   COMMAND ==>                                              SCROLL ==> CSR
            z/XPF Report Hierarchy:
               Time Segment: 01

            Hierarchical drill down for:          Dynamic SQL activity
            Current level of report hierarchy:            Time Segment
            This is the base of the report hierarchy
            Number of dynamic SQL statements observed:                4

            DB2 activity, by dynamic SQl statement

            When the length of the text statement is greater than 4 display
            lines, the entire text may be viewed by placing the cursor on
            one of the text display lines, and depressing the enter key.


            Length of SQL text:       49 Length displayed:       49

            INSERT INTO DAD0001.ASMSRCON VALUES(?,?,?,?,?,?)

            First observed activity, this statement:     07.48.46:35.6414
            Last observed activity, this statement:      07.49.24:21.9401
_  Dril     Number of Csects with activity this statement:           66
            Total events this SQL statement:                    981,895
            Percent of time segment total DB2 events:             45.36
            ....10...20...30...40...50...60...70...80...90...100

            Percent of time segment total events:                 24.72



            Length of SQL text:       56 Length displayed:       56

            INSERT INTO DAD0001.ASMSRCZR VALUES ( ?, ?, ?, ?, ?, ?)

            First observed activity, this statement:     07.48.46:33.7204
            Last observed activity, this statement:      07.53.24:05.7454
_  Dril     Number of Csects with activity this statement:           56
            Total events this SQL statement:                    980,058
            Percent of time segment total DB2 events:             45.28
            ....10...20...30...40...50...60...70...80...90...100
```

Figure 6-2-3.

In Figure 6-3 you can see “orientation” information in yellow, then the first of five SQL statements, along with statistics on each one. Just for the heck of it, let’s drill again into the first SQL statement and see what we get.

Now I have the panel in Figure 6-2-4 below:



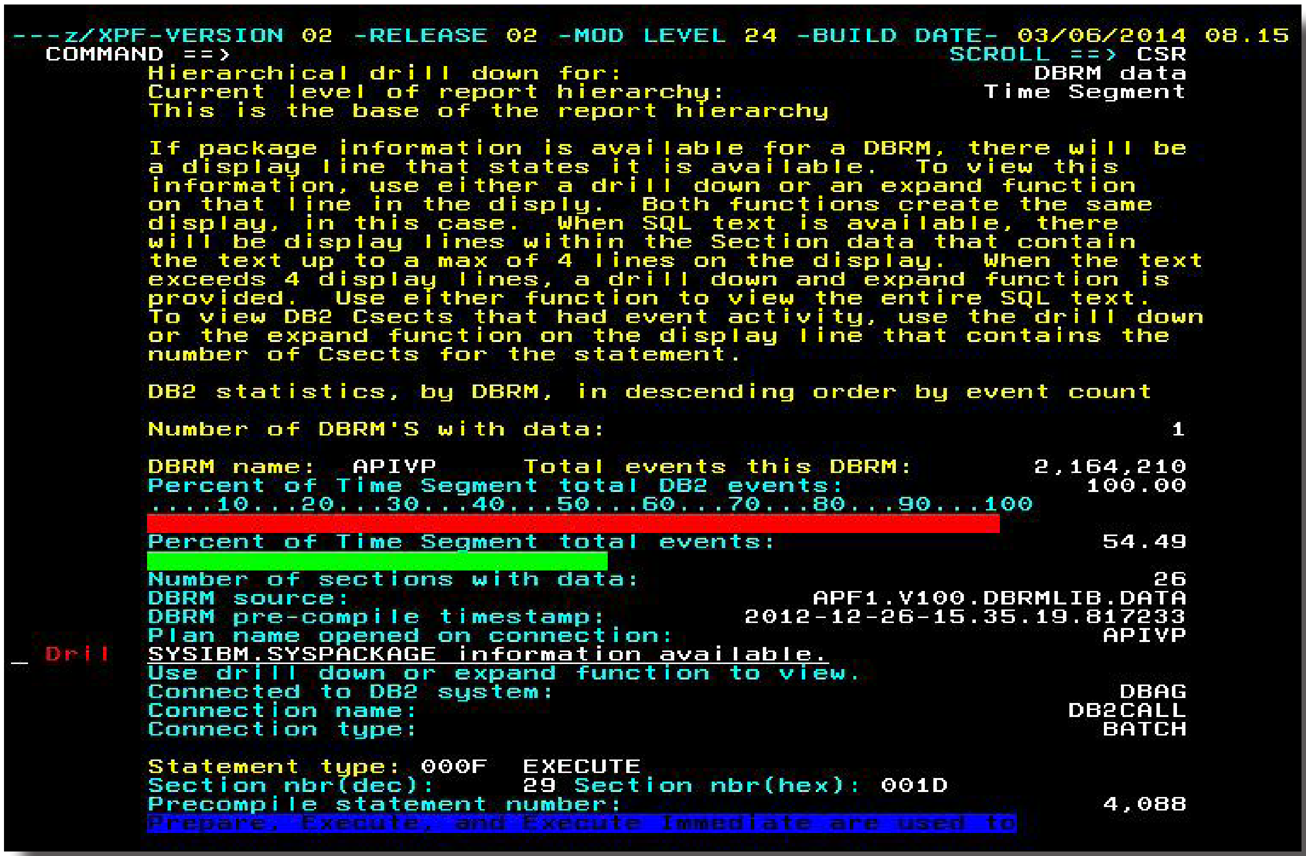
```
---z/XPF-VERSION 02 -RELEASE 02 -MOD LEVEL 24 -BUILD DATE- 03/06/2014 08.15
    COMMAND ==>                                                SCROLL ==> CSR

                 Csect name:                                         DSNISGRT
                 DSNWMODS = DSNISGRT READ AND TEST A SEGMENTED DATA PAGE
                 Total events this Csect:                             463,900
                 Percent of total text statement events:                47.24
                 ....10...20...30...40...50...60...70...80...90...100

                 Percent of total Time Segment DB2 events:              21.43

                 Percent of total Time Segment events:                  11.68


                 Csect name:                                         DSNISGSU
                 DSNWMODS = DSNISGSU UPDATE SEGMENTED SPACE MAP PAGE
                 Total events this Csect:                              35,766
                 Percent of total text statement events:                03.64
                 ....10...20...30...40...50...60...70...80...90...100

                 Percent of total Time Segment DB2 events:              01.65

                 Percent of total Time Segment events:                  00.90
                 Less than 1 percent

                 Csect name:                                         DSNISRTI
                 DSNWMODS = DSNISRTI INSERT, POSSIBLY CONNECT RECORD
                 Total events this Csect:                              12,370
                 Percent of total text statement events:                01.25
                 ....10...20...30...40...50...60...70...80...90...100

                 Percent of total Time Segment DB2 events:              00.57
                 Less than 1 percent
                 Percent of total Time Segment events:                  00.31
                 Less than 1 percent

                 Csect name:                                         DSNB1GET
                 DSNWMODS = DSNB1GET RETRIEVE REQUESTED PAGE
                 Total events this Csect:                               3,322
                 Percent of total text statement events:                00.33
                 ....10...20...30...40...50...60...70...80...90...100
                 Less than 1 percent
                 Percent of total Time Segment DB2 events:              00.15
```

Figure 6-2-4.

In Figure 6-4 above, z/XPF has sorted the actions within each DB2 csect from most-active to least active and tells us what each DB2 csect's function is. Data capture picked up 463,900 events with instruction addresses that mapped to that DB2 SQL statement.

Let's pick "Door #2", "Static SQL text". I'll back out from here using PF3 and will arrive back at the panel in Figure 6-2. I'll select "Static SQL Text" and will drill down.

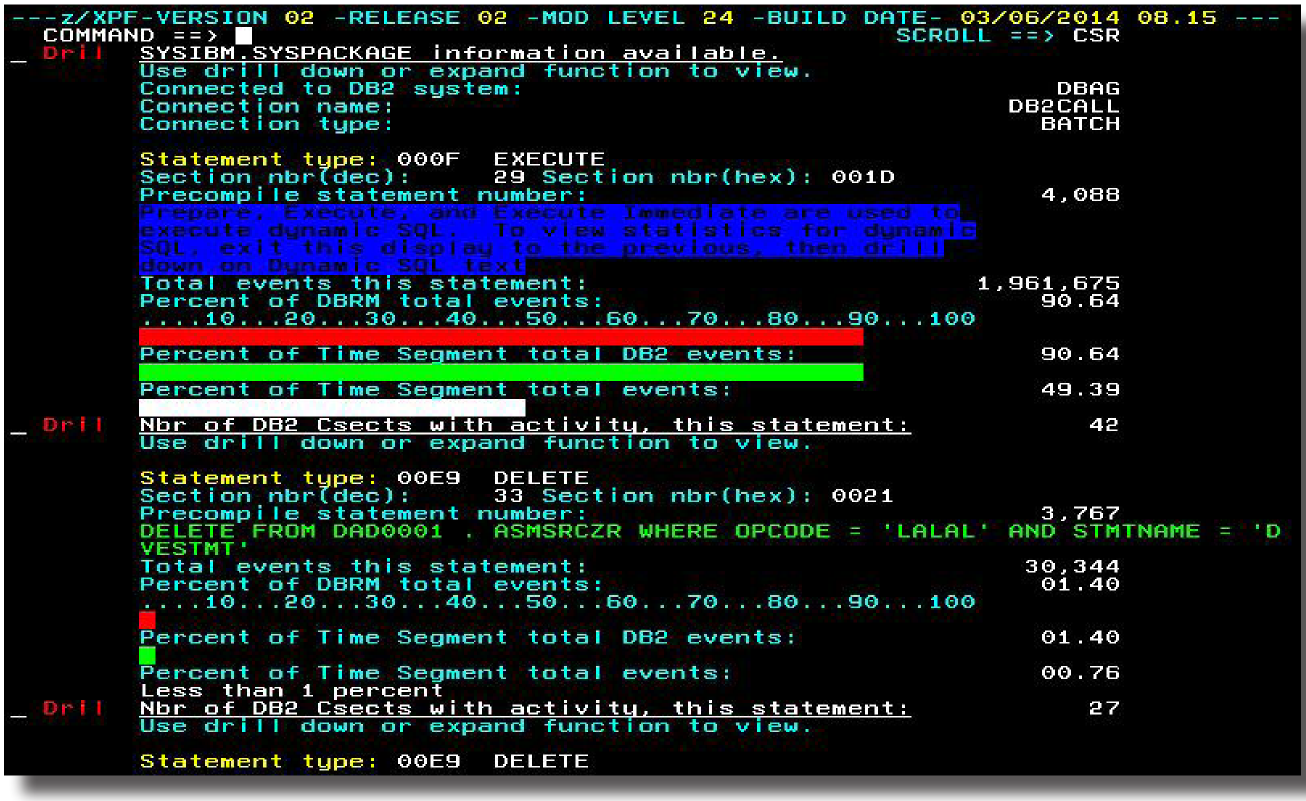
I arrive at the panel below, in Figure 6-2-5:



```
---z/XPF-VERSION 02 -RELEASE 02 -MOD LEVEL 24 -BUILD DATE- 03/06/2014 08.15
 COMMAND ==>                                                   SCROLL ==> CSR
       Hierarchical drill down for:                                DBRM data
       Current level of report hierarchy:                       Time Segment
       This is the base of the report hierarchy

       If package information is available for a DBRM, there will be
       a display line that states it is available.  To view this
       information, use either a drill down or an expand function
       on that line in the disply.  Both functions create the same
       display, in this case.  When SQL text is available, there
       will be display lines within the Section data that contain
       the text up to a max of 4 lines on the display.  When the text
       exceeds 4 display lines, a drill down and expand function is
       provided.  Use either function to view the entire SQL text.
       To view DB2 Csects that had event activity, use the drill down
       or the expand function on the display line that contains the
       number of Csects for the statement.

       DB2 statistics, by DBRM, in descending order by event count

       Number of DBRM'S with data:                                     1

       DBRM name:  APIVP     Total events this DBRM:           2,164,210
       Percent of Time Segment total DB2 events:                  100.00
       ....10...20...30...40...50...60...70...80...90...100

       Percent of Time Segment total events:                        54.49

       Number of sections with data:                                  26
       DBRM source:                                 APF1.V100.DBRMLIB.DATA
       DBRM pre-compile timestamp:              2012-12-26-15.35.19.817233
       Plan name opened on connection:                              APIVP
_ Dril SYSIBM.SYSPACKAGE information available.
       Use drill down or expand function to view.
       Connected to DB2 system:                                      DBAG
       Connection name:                                          DB2CALL
       Connection type:                                            BATCH

       Statement type: 000F  EXECUTE
       Section nbr(dec):     29 Section nbr(hex): 001D
       Precompile statement number:                                 4,088
       Prepare, Execute, and Execute Immediate are used to
```

Figure 6-2-5.

Most of the panels I’ve shown so far have much more going on than a single page of display. The panel in Figure 6-5 is just the start of what you may display. Let’s locate the cursor on “SYSIBM.SYSPACKAGE”, page down and re-orient the panel, in Figure 6-2-6:



```
---z/XPF-VERSION 02 -RELEASE 02 -MOD LEVEL 24 -BUILD DATE- 03/06/2014 08.15 ---
 COMMAND ==>                                                   SCROLL ==> CSR
_ Dril    SYSIBM.SYSPACKAGE information available.
          Use drill down or expand function to view.
          Connected to DB2 system:                                        DBAG
          Connection name:                                            DB2CALL
          Connection type:                                              BATCH

          Statement type: 000F   EXECUTE
          Section nbr(dec):      29 Section nbr(hex): 001D
          Precompile statement number:                                   4,088
          Prepare, Execute, and Execute Immediate are used to
          execute dynamic SQL.  To view statistics for dynamic
          SQL, exit this display to the previous, then drill
          down on Dynamic SQL text
          Total events this statement:                               1,961,675
          Percent of DBRM total events:                                  90.64
          ....10...20...30...40...50...60...70...80...90...100

          Percent of Time Segment total DB2 events:                      90.64

          Percent of Time Segment total events:                          49.39

_ Dril    Nbr of DB2 Csects with activity, this statement:                  42
          Use drill down or expand function to view.

          Statement type: 00E9   DELETE
          Section nbr(dec):      33 Section nbr(hex): 0021
          Precompile statement number:                                   3,767
          DELETE FROM DAD0001 . ASMSRCZR WHERE OPCODE = 'LALAL' AND STMTNAME = 'D
          VESTMT'
          Total events this statement:                                  30,344
          Percent of DBRM total events:                                  01.40
          ....10...20...30...40...50...60...70...80...90...100

          Percent of Time Segment total DB2 events:                      01.40

          Percent of Time Segment total events:                          00.76
          Less than 1 percent
_ Dril    Nbr of DB2 Csects with activity, this statement:                  27
          Use drill down or expand function to view.

          Statement type: 00E9   DELETE
```

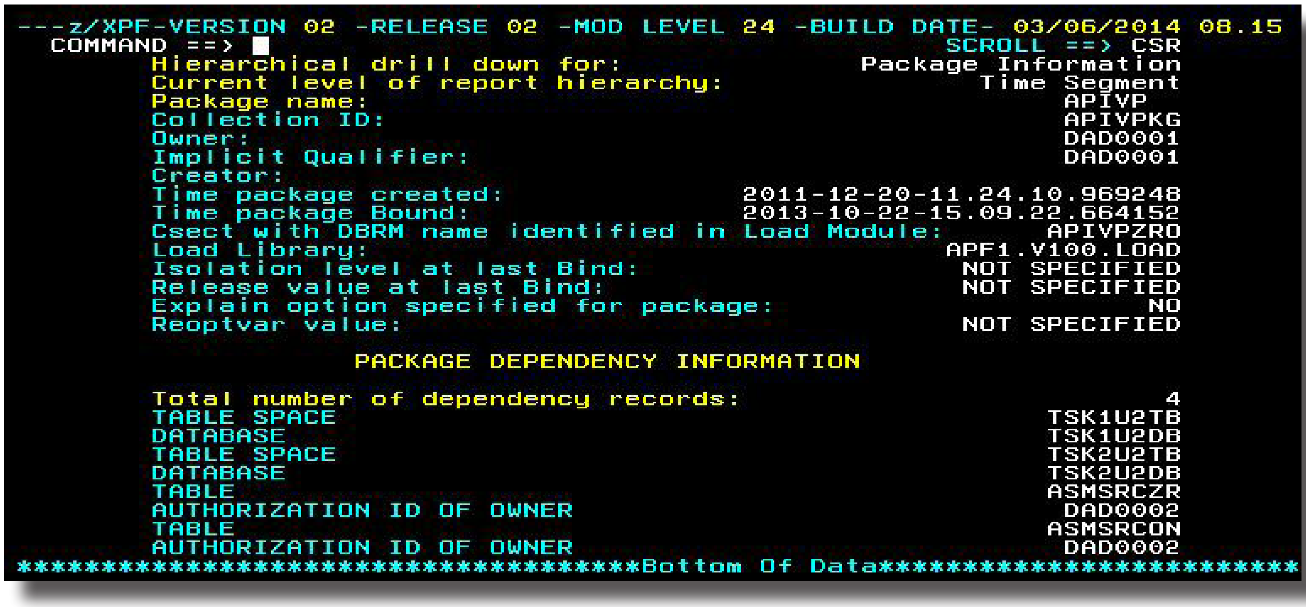
Figure 6-2-6.

In the Figure above, I can see that SYSIBM.SYSPACKAGE had 1,961,675 events which took up 90.64% of the total DB2 events, and 49.39% of the total events for the Time Segment of this report (there's only one Time Segment defined).

Let's drill into SYSIBM.SYSPACKAGE, and see what we get. Here's the result, in Figure 6-6 below:

The life of a technical writer is a perplexing one. One attempts to find the right balance of helpfulness, clarity and completeness without revealing one's near-total ignorance of the subject matter. Dave Day swears that the panel above in Figure 5-44 will be instantly useful to any DB2-oriented person. May it be so.

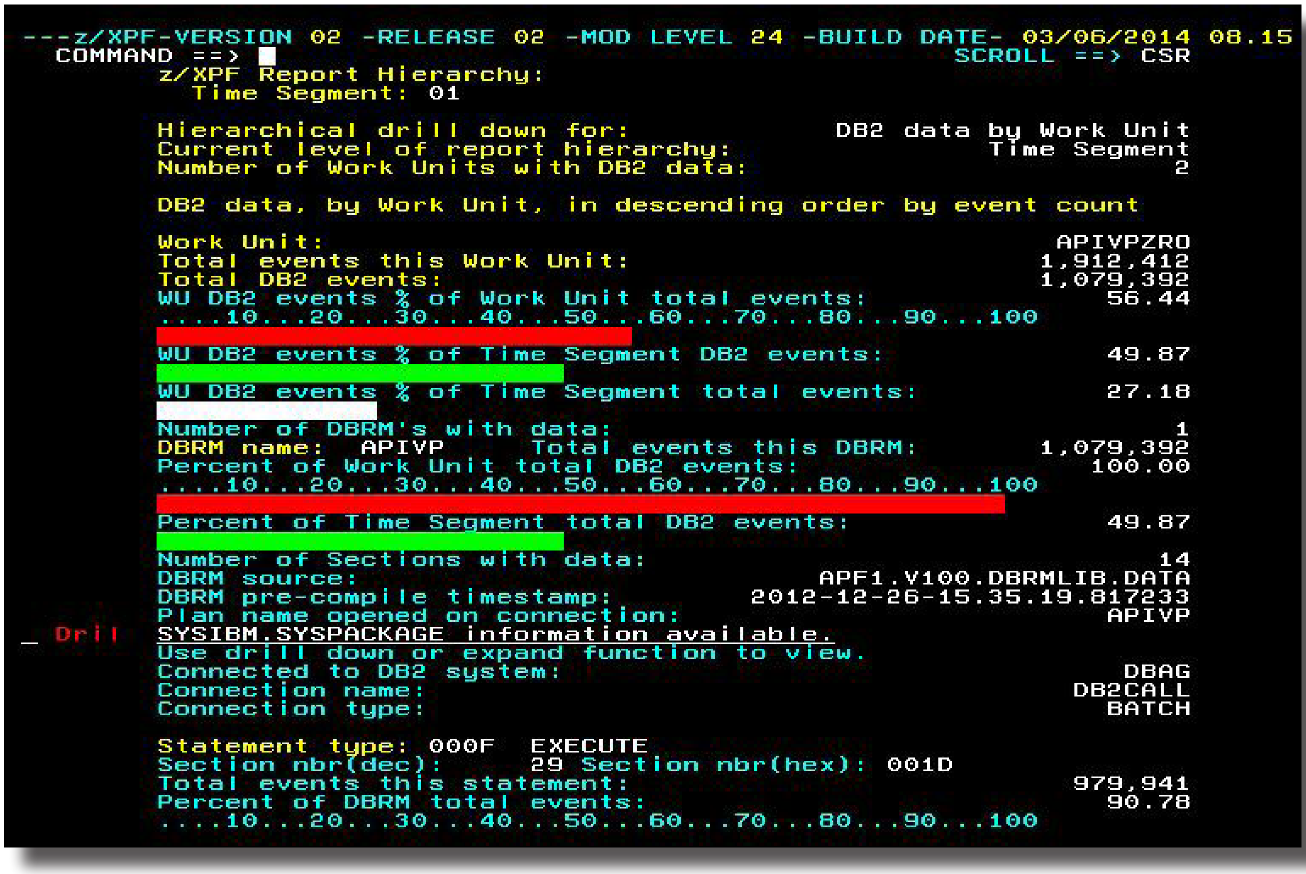
Let's try "Door #3" from Figure 6-2. This time we'll drill down into "By Work Unit". I tab down to that item and drill down to arrive at the panel below, in Figure 6-2-7:



```
---z/XPF-VERSION 02 -RELEASE 02 -MOD LEVEL 24 -BUILD DATE- 03/06/2014 08.15
   COMMAND ==>                                                 SCROLL ==> CSR
         Hierarchical drill down for:                    Package Information
         Current level of report hierarchy:                     Time Segment
         Package name:                                                 APIVP
         Collection ID:                                             APIVPKG
         Owner:                                                     DAD0001
         Implicit Qualifier:                                        DAD0001
         Creator:
         Time package created:                   2011-12-20-11.24.10.969248
         Time package Bound:                     2013-10-22-15.09.22.664152
         Csect with DBRM name identified in Load Module:            APIVPZRO
         Load Library:                                         APF1.V100.LOAD
         Isolation level at last Bind:                          NOT SPECIFIED
         Release value at last Bind:                            NOT SPECIFIED
         Explain option specified for package:                             NO
         Reoptvar value:                                        NOT SPECIFIED

                     PACKAGE DEPENDENCY INFORMATION

         Total number of dependency records:                                4
         TABLE SPACE                                                 TSK1U2TB
         DATABASE                                                    TSK1U2DB
         TABLE SPACE                                                 TSK2U2TB
         DATABASE                                                    TSK2U2DB
         TABLE                                                       ASMSRCZR
         AUTHORIZATION ID OF OWNER                                    DAD0002
         TABLE                                                       ASMSRCON
         AUTHORIZATION ID OF OWNER                                    DAD0002
*************************************Bottom Of Data**************************
```

Figure 6-2-7.

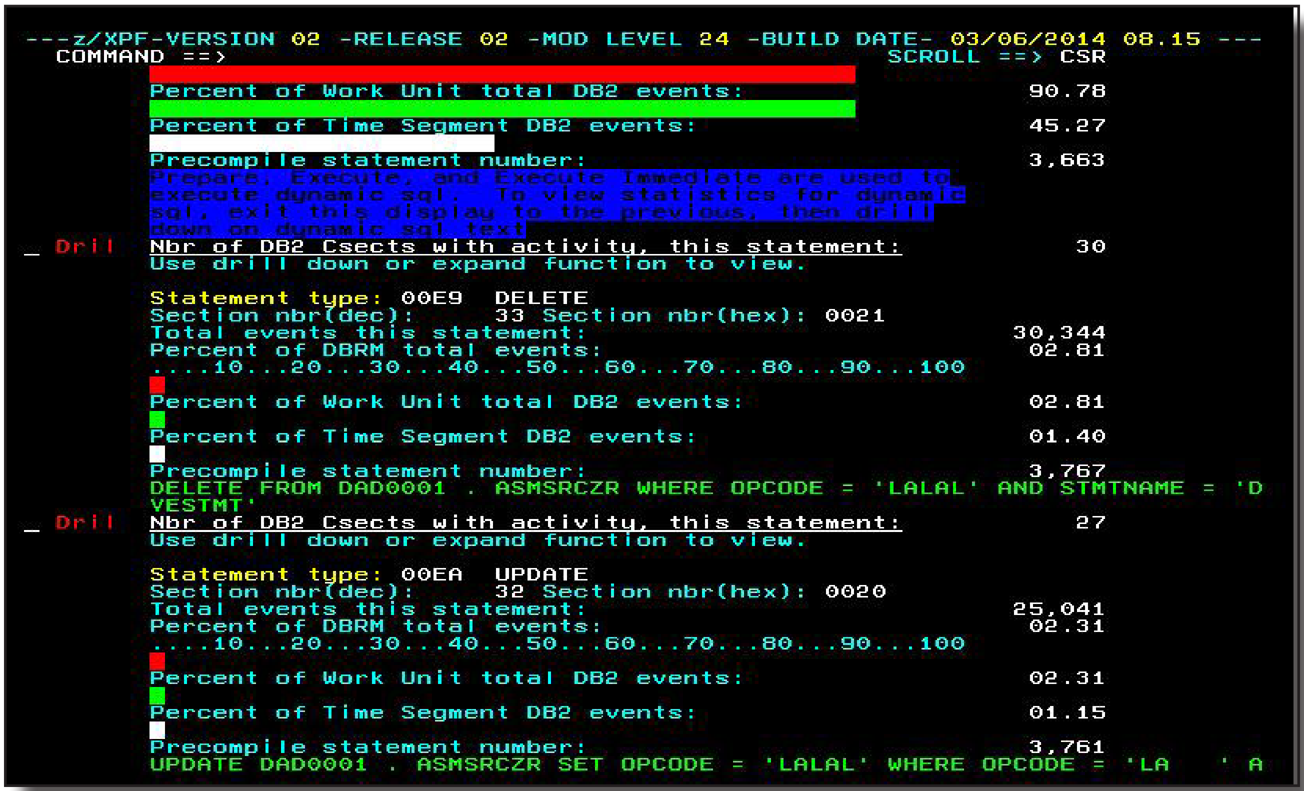
Let’s try Door #3 now. I’ll use PF3 to back out to the panel in Figure 6-2, and I’ll drill into “By Work Unit”. I arrive at the panel in Figure 6-2-8:



```
---z/XPF-VERSION 02 -RELEASE 02 -MOD LEVEL 24 -BUILD DATE- 03/06/2014 08.15
   COMMAND ==>                                                 SCROLL ==> CSR
         z/XPF Report Hierarchy:
           Time Segment: 01

         Hierarchical drill down for:              DB2 data by Work Unit
         Current level of report hierarchy:                     Time Segment
         Number of Work Units with DB2 data:                               2

         DB2 data, by Work Unit, in descending order by event count

         Work Unit:                                                 APIVPZRO
         Total events this Work Unit:                              1,912,412
         Total DB2 events:                                         1,079,392
         WU DB2 events % of Work Unit total events:                    56.44
         ....10...20...30...40...50...60...70...80...90...100

         WU DB2 events % of Time Segment DB2 events:                   49.87

         WU DB2 events % of Time Segment total events:                 27.18

         Number of DBRM's with data:                                       1
         DBRM name:  APIVP     Total events this DBRM:             1,079,392
         Percent of Work Unit total DB2 events:                       100.00
         ....10...20...30...40...50...60...70...80...90...100

         Percent of Time Segment total DB2 events:                     49.87

         Number of Sections with data:                                    14
         DBRM source:                                    APF1.V100.DBRMLIB.DATA
         DBRM pre-compile timestamp:              2012-12-26-15.35.19.817233
         Plan name opened on connection:                               APIVP
 _ Dril  SYSIBM.SYSPACKAGE information available.
         Use drill down or expand function to view.
         Connected to DB2 system:                                       DBAG
         Connection name:                                          DB2CALL
         Connection type:                                            BATCH

         Statement type: 000F   EXECUTE
         Section nbr(dec):       29 Section nbr(hex): 001D
         Total events this statement:                                979,941
         Percent of DBRM total events:                                 90.78
         ....10...20...30...40...50...60...70...80...90...100
```

Figure 6-2-8.

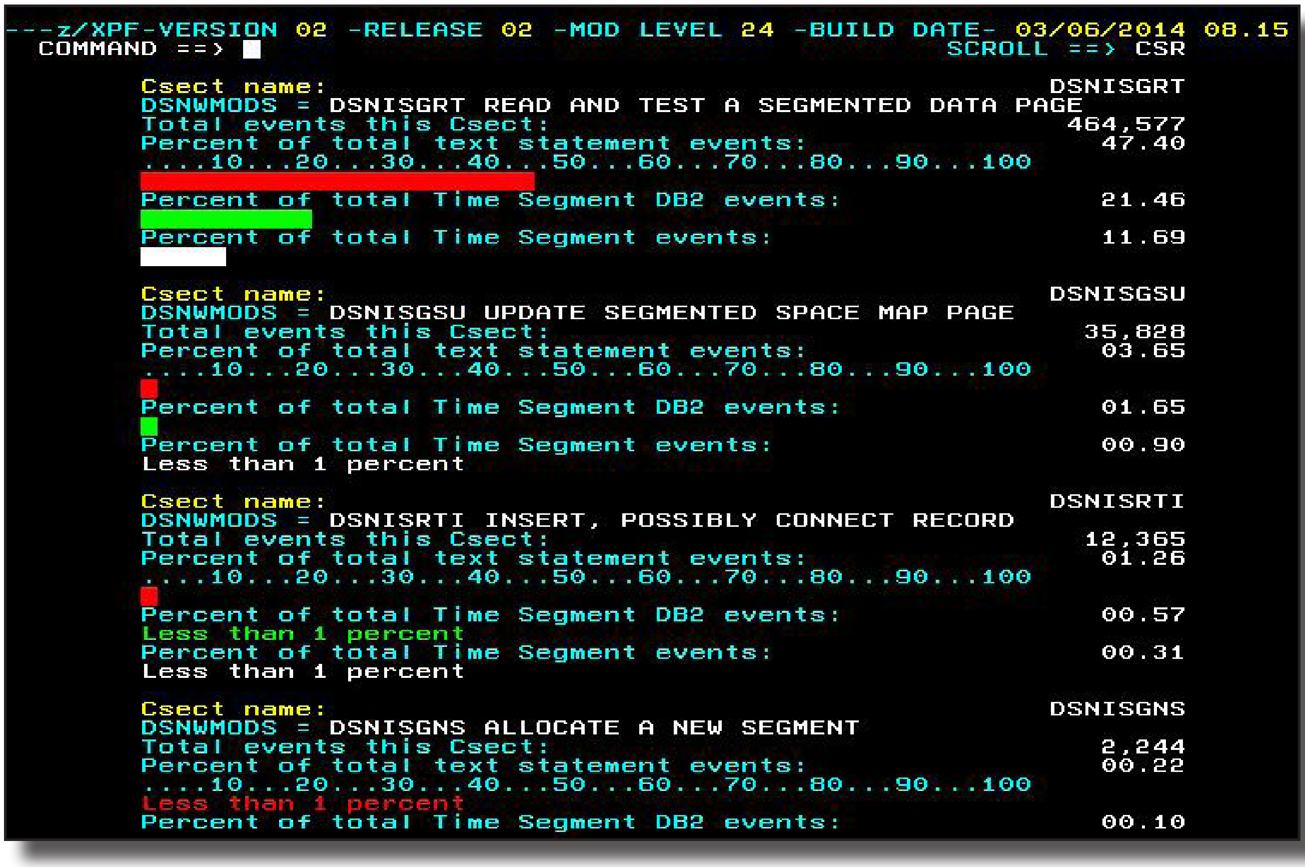
Again, SYSIBM.SYSPACKAGE is the "big dog" with 979,941 events. Let's page down and look at the next biggest Work Unit. I page down and see Figure 6-2-9:



```
---z/XPF-VERSION 02 -RELEASE 02 -MOD LEVEL 24 -BUILD DATE- 03/06/2014 08.15 ---
  COMMAND ==>                                                   SCROLL ==> CSR

            Percent of Work Unit total DB2 events:                        90.78

            Percent of Time Segment DB2 events:                           45.27

            Precompile statement number:                                  3,663
            Prepare, Execute, and Execute Immediate are used to
            execute dynamic sql.  To view statistics for dynamic
            sql, exit this display to the previous, then drill
            down on dynamic sql text
_ Drill     Nbr of DB2 Csects with activity, this statement:                 30
            Use drill down or expand function to view.

            Statement type: 00E9  DELETE
            Section nbr(dec):     33 Section nbr(hex): 0021
            Total events this statement:                                 30,344
            Percent of DBRM total events:                                 02.81
            ....10...20...30...40...50...60...70...80...90...100

            Percent of Work Unit total DB2 events:                        02.81

            Percent of Time Segment DB2 events:                           01.40

            Precompile statement number:                                  3,767
            DELETE FROM DAD0001 . ASMSRCZR WHERE OPCODE = 'LALAL' AND STMTNAME = 'D
            VESTMT'
_ Drill     Nbr of DB2 Csects with activity, this statement:                 27
            Use drill down or expand function to view.

            Statement type: 00EA  UPDATE
            Section nbr(dec):     32 Section nbr(hex): 0020
            Total events this statement:                                 25,041
            Percent of DBRM total events:                                 02.31
            ....10...20...30...40...50...60...70...80...90...100

            Percent of Work Unit total DB2 events:                        02.31

            Percent of Time Segment DB2 events:                           01.15

            Precompile statement number:                                  3,761
            UPDATE DAD0001 . ASMSRCZR SET OPCODE = 'LALAL' WHERE OPCODE = 'LA    ' A
```

Figure 6-2-9.

Well, the next largest Work Unit is a DELETE, with 36,344 events. Let's drill down, and we'll arrive at Figure 6-2-10:



```
---z/XPF-VERSION 02 -RELEASE 02 -MOD LEVEL 24 -BUILD DATE- 03/06/2014 08.15
  COMMAND ==>                                                  SCROLL ==> CSR

         Csect name:                                                 DSNISGRT
         DSNWMODS = DSNISGRT READ AND TEST A SEGMENTED DATA PAGE
         Total events this Csect:                                     464,577
         Percent of total text statement events:                        47.40
         ....10...20...30...40...50...60...70...80...90...100

         Percent of total Time Segment DB2 events:                      21.46

         Percent of total Time Segment events:                          11.69


         Csect name:                                                 DSNISGSU
         DSNWMODS = DSNISGSU UPDATE SEGMENTED SPACE MAP PAGE
         Total events this Csect:                                      35,828
         Percent of total text statement events:                        03.65
         ....10...20...30...40...50...60...70...80...90...100

         Percent of total Time Segment DB2 events:                      01.65

         Percent of total Time Segment events:                          00.90
         Less than 1 percent

         Csect name:                                                 DSNISRTI
         DSNWMODS = DSNISRTI INSERT, POSSIBLY CONNECT RECORD
         Total events this Csect:                                      12,365
         Percent of total text statement events:                        01.26
         ....10...20...30...40...50...60...70...80...90...100

         Percent of total Time Segment DB2 events:                      00.57
         Less than 1 percent
         Percent of total Time Segment events:                          00.31
         Less than 1 percent

         Csect name:                                                 DSNISGNS
         DSNWMODS = DSNISGNS ALLOCATE A NEW SEGMENT
         Total events this Csect:                                       2,244
         Percent of total text statement events:                        00.22
         ....10...20...30...40...50...60...70...80...90...100
         Less than 1 percent
         Percent of total Time Segment DB2 events:                      00.10
```

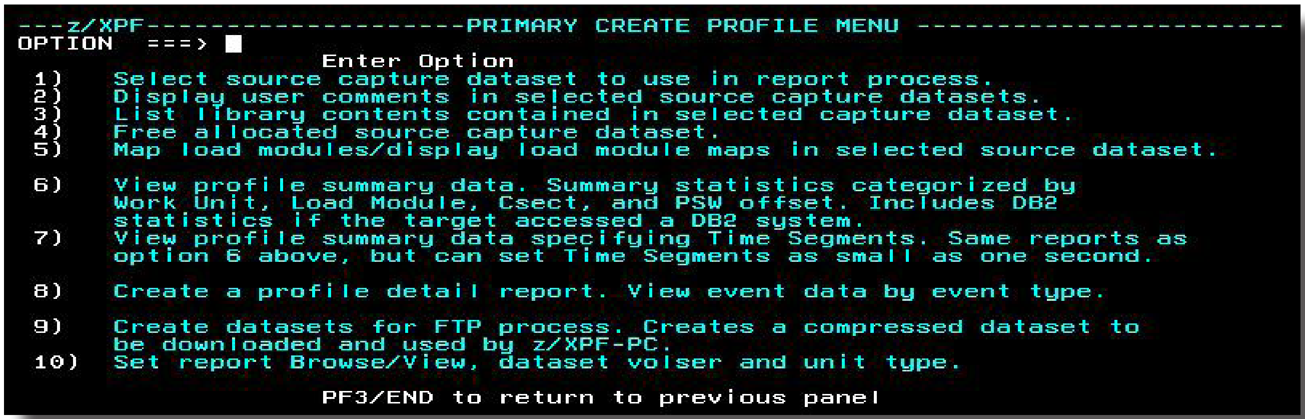
Figure 6-2-10.

We've reached the last level in this hiearchy. To be absolutely truthful, your author *knows nothing about DB2*, but is fairly certain this will be revelatory content to a real DB2 programmer!

Chapter 7 - Detail Reporting

7-1 Option 8: Create a profile detail report.

You can access z/XPF's Detail Reporting features by using "Option 8 Detail Reporting" from the Primary Create Profile Menu. Here is the panel again for easy reference:

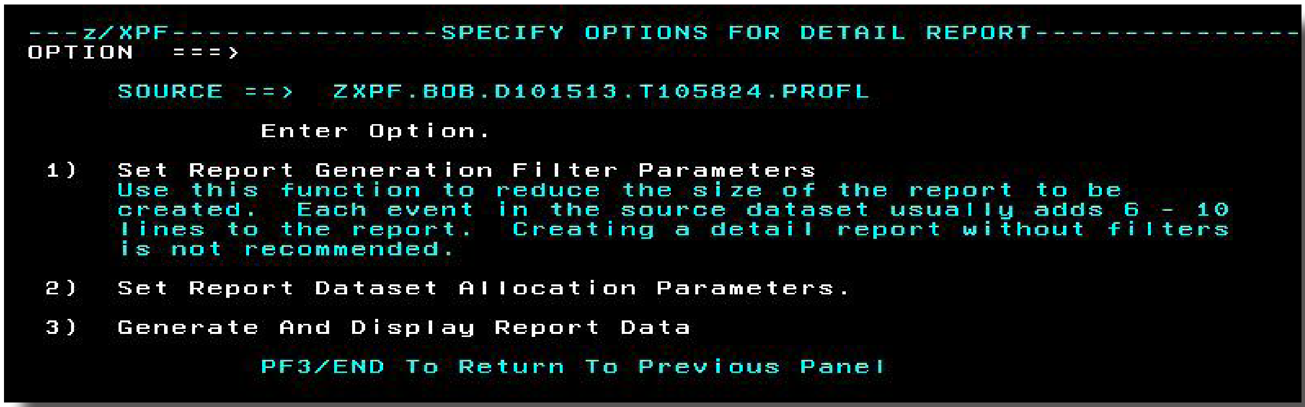


```
---z/XPF--------------------PRIMARY CREATE PROFILE MENU -----------------------
OPTION   ===> 
                   Enter Option
 1)    Select source capture dataset to use in report process.
 2)    Display user comments in selected source capture datasets.
 3)    List library contents contained in 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. Same reports as
       option 6 above, but can set Time Segments 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 7-1-1.

If you have identified something of interest in your review of Summary Reports, then you may use Detail Reporting to see the underlying event data. Detail Reports give you access to information at the very lowest levels. In effect you'll be looking at "dressed up" Trace Records formatted to deliver the information far more clearly than a raw Trace Record.

Option 8 from the PRIMARY CREATE PROFILE MENU brings you to this screen, Figure 7-1-2 below:



```
---z/XPF---------------SPECIFY OPTIONS FOR DETAIL REPORT---------------
OPTION   ===>

      SOURCE ==>  ZXPF.BOB.D101513.T105824.PROFL

              Enter Option.

 1)   Set Report Generation Filter Parameters
      Use this function to reduce the size of the report to be
      created.  Each event in the source dataset usually adds 6 - 10
      lines to the report.  Creating a detail report without filters
      is not recommended.

 2)   Set Report Dataset Allocation Parameters.

 3)   Generate And Display Report Data

              PF3/END To Return To Previous Panel
```

Figure 7-1-2.

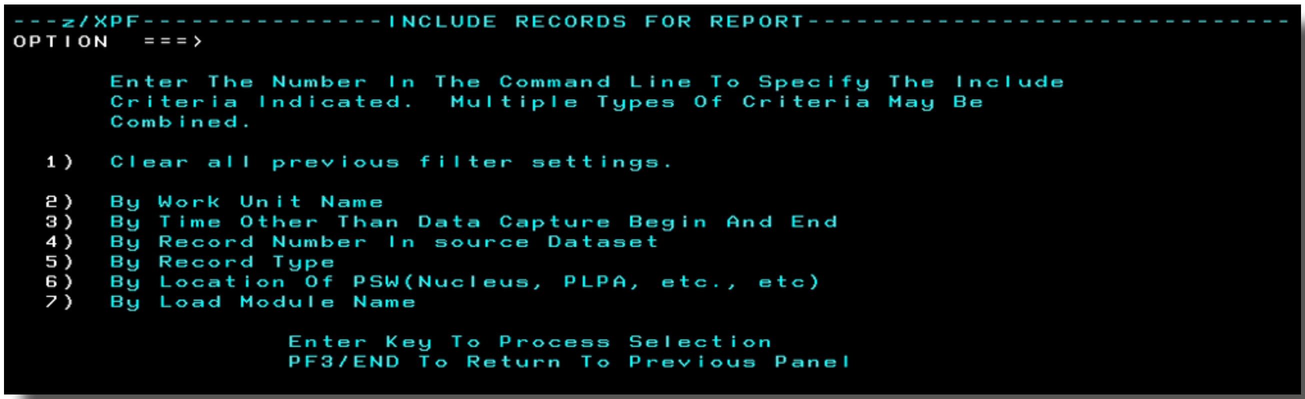
When z/XPF receives a raw Trace record, it edits the Trace record into a more intelligible format. Thereafter it's referred to as an "Event record". A Detail Report consists of all Event records in the source capture dataset that pass the filters you set. Each Event record generates from 8 to 10 lines of output in the report. You can view the unique events that contributed to the totals statistics for the various categories.

The amount of output in the report dataset is dependent upon the number of events in the source capture dataset, and the filters that you set prior to creating the report.

Please keep in mind that in Detail Reporting you will potentially be working with MILLIONS of data points. Therefore, if you don't employ z/XPF's powerful filtering technology you may well experience a SPACE ABEND because the resulting report will be so huge. You also may not be able to find the information you need amongs all the data.

As you use z/XPF's filters bear in mind that:

- The judicious use of filter settings yields a report that is easy to understand.
- Filter settings are collective in nature. They persist until and unless you clear them.
- Any time you include an item from a category (such as Work Units) all other items in that category are EXCLUDED.
- The in-judicious use of filter settings can cause the creation of a report containing hundreds of thousands of lines of output and possibly a space abend due to lack of DASD space.
- It is easy to *clear ALL filters at once* by choosing Option 1 from the "Include Records For Report panel, which is shown below in Figure 7-1-3:



```
---z/XPF---------------INCLUDE RECORDS FOR REPORT-----------------------------
OPTION  ===>

      Enter The Number In The Command Line To Specify The Include
      Criteria Indicated.  Multiple Types Of Criteria May Be
      Combined.

  1)  Clear all previous filter settings.

  2)  By Work Unit Name
  3)  By Time Other Than Data Capture Begin And End
  4)  By Record Number In source Dataset
  5)  By Record Type
  6)  By Location Of PSW(Nucleus, PLPA, etc., etc)
  7)  By Load Module Name

                 Enter Key To Process Selection
                 PF3/END To Return To Previous Panel
```

Figure 7-1-3.

In this book we'll go over how to use all of the menu choices for filtering and creating Detail reports. I'll list each of the options from Figure 7-2 in the following pages. However, there's a useful sub-section after this discussion that offers "Tips and Techniques for Creating Detail Reports." Let's get back to work on creating Detail reports.

On the panel above in Figure 7-1-2 you have three choices.

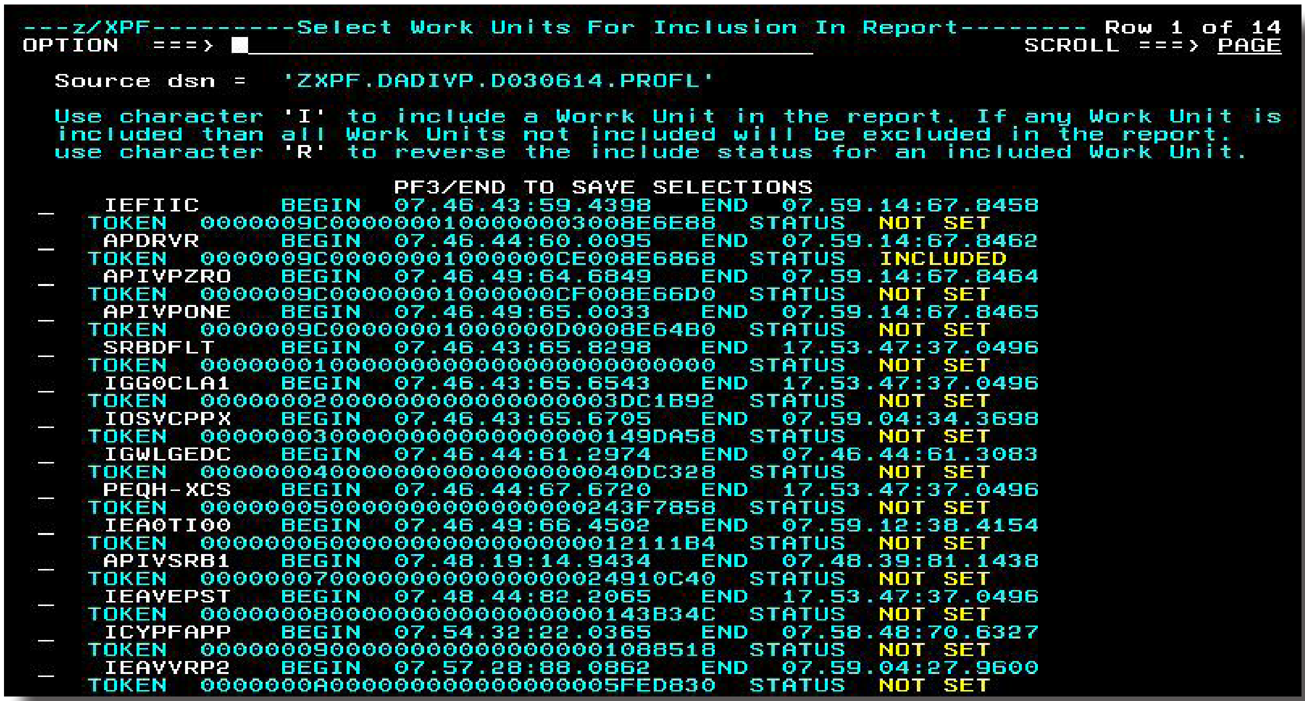
- Option 1: You can set your report's filters.
- Option 2: Control the allocation/volser/size of the dataset that will be populated by your report. This allows you to override system defaults as to DASD device or pool, change

VOLSERs and the primary/secondary allocations for your report.

- Option 3 Create your report.

7-2 Detail reporting by Work Unit Name

Let's set some filters and look at Work Unit statistics. From the panel in Figure 7-2, I'll pick Option 1, and then Option 2 from the panel in Figure 7-3 to select work units for my report. Please see Figure 7-2-1 below:

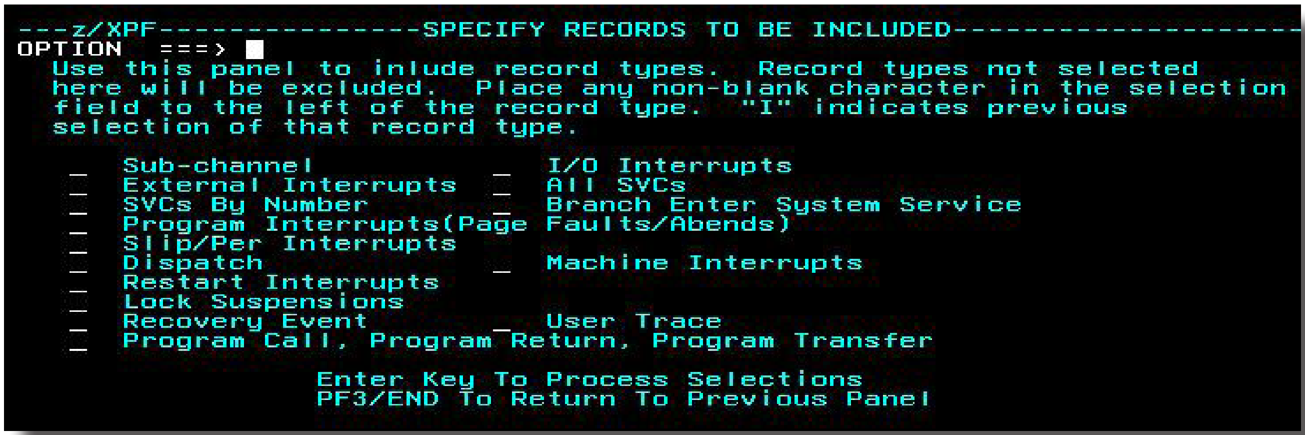


```
---z/XPF----------Select Work Units For Inclusion In Report--------- Row 1 of 14
OPTION   ===> _                                                SCROLL ===> PAGE

   Source dsn =   'ZXPF.DADIVP.D030614.PROFL'

   Use character 'I' to include a Worrk Unit in the report. If any Work Unit is
   included than all Work Units not included will be excluded in the report.
   use character 'R' to reverse the include status for an included Work Unit.

                       PF3/END TO SAVE SELECTIONS
 _     IEFIIC      BEGIN  07.46.43:59.4398   END  07.59.14:67.8458
     TOKEN  0000009C000000001000000003008E6E88  STATUS  NOT SET
 _     APDRVR      BEGIN  07.46.44:60.0095   END  07.59.14:67.8462
     TOKEN  0000009C0000000010000000CE008E6868  STATUS  INCLUDED
 _     APIVPZRO    BEGIN  07.46.49:64.6849   END  07.59.14:67.8464
     TOKEN  0000009C0000000010000000CF008E66D0  STATUS  NOT SET
 _     APIVPONE    BEGIN  07.46.49:65.0033   END  07.59.14:67.8465
     TOKEN  0000009C0000000010000000D0008E64B0  STATUS  NOT SET
 _     SRBDFLT     BEGIN  07.46.43:65.8298   END  17.53.47:37.0496
     TOKEN  0000000100000000000000000000000000  STATUS  NOT SET
 _     IGG0CLA1    BEGIN  07.46.43:65.6543   END  17.53.47:37.0496
     TOKEN  00000002000000000000000000003DC1B92  STATUS  NOT SET
 _     IOSVCPPX    BEGIN  07.46.43:65.6705   END  07.59.04:34.3698
     TOKEN  00000003000000000000000000149DA58  STATUS  NOT SET
 _     IGWLGEDC    BEGIN  07.46.44:61.2974   END  07.46.44:61.3083
     TOKEN  000000040000000000000000040DC328  STATUS  NOT SET
 _     PEQH-XCS    BEGIN  07.46.44:67.6720   END  17.53.47:37.0496
     TOKEN  0000000500000000000000000243F7858  STATUS  NOT SET
 _     IEAOTI00    BEGIN  07.46.49:66.4502   END  07.59.12:38.4154
     TOKEN  0000000600000000000000000012111B4  STATUS  NOT SET
 _     APIVSRB1    BEGIN  07.48.19:14.9434   END  07.48.39:81.1438
     TOKEN  00000007000000000000000024910C40  STATUS  NOT SET
 _     IEAVEPST    BEGIN  07.48.44:82.2065   END  17.53.47:37.0496
     TOKEN  0000000800000000000000000143B34C  STATUS  NOT SET
 _     ICYPFAPP    BEGIN  07.54.32:22.0365   END  07.58.48:70.6327
     TOKEN  00000009000000000000000001088518  STATUS  NOT SET
 _     IEAVVRP2    BEGIN  07.57.28:88.0862   END  07.59.04:27.9600
     TOKEN  0000000A00000000000000000005FED830  STATUS  NOT SET
```

Figure 7-2-1.

I'll select APDRVR, z/XPF's little "test-drive" program. I put an "I" next to it so that it will be included. z/XPF replaces the phrase "NOT SET" with the word "INCLUDED". I press PF3 to back out of this panel, and I'm back to the previous panel (Figure 7-1-3).

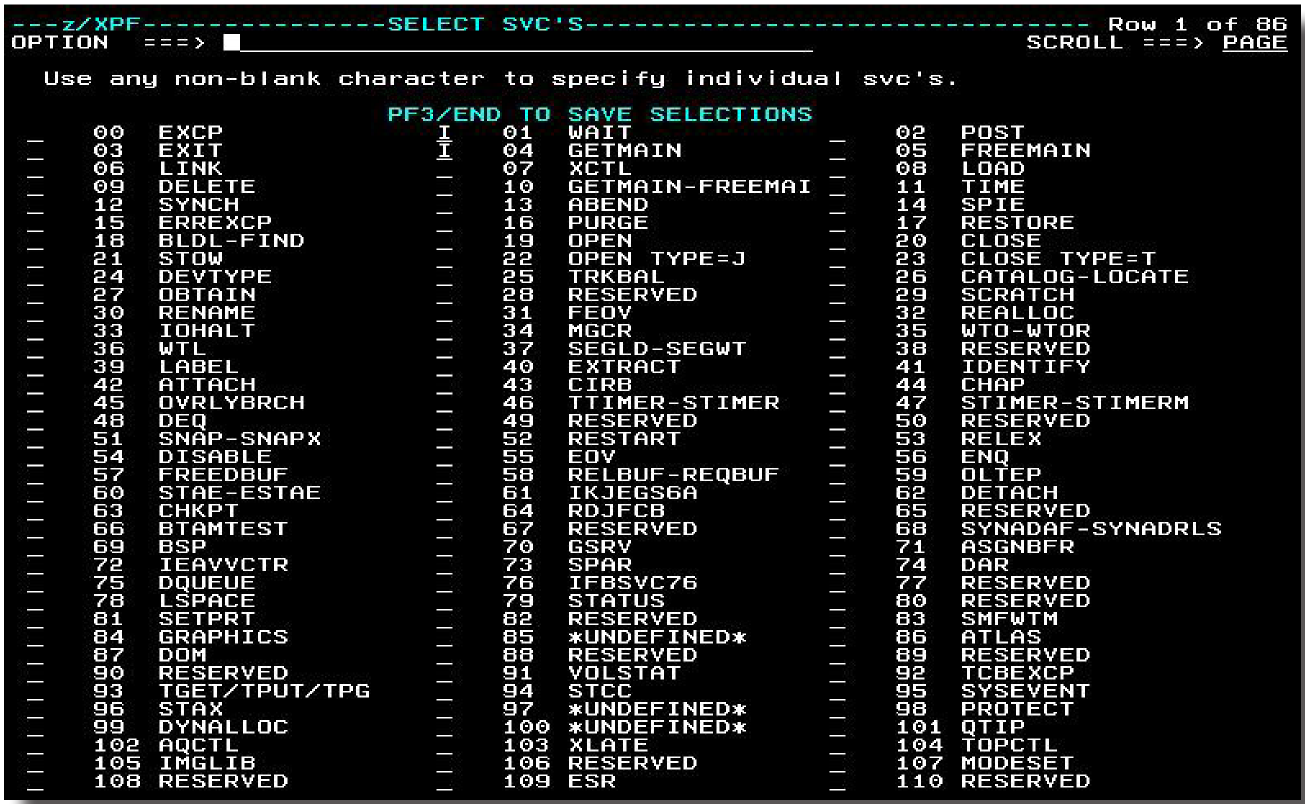
Let's set another filter for SVCs. This will have the effect of filtering SVCs ONLY within Work Unit APDRVR (remember, filter settings are cumulative). To do that, I'll pick Option 5 from Figure 7-1-3 (By Record type). Now I am presented with Figure 7-2-2 below:



```
---z/XPF---------------SPECIFY RECORDS TO BE INCLUDED-------------------
OPTION   ===> 
    Use this panel to inlude record types.  Record types not selected
    here will be excluded.  Place any non-blank character in the selection
    field to the left of the record type.  "I" indicates previous
    selection of that record type.

    _   Sub-channel                _   I/O Interrupts
    _   External Interrupts        _   All SVCs
    _   SVCs By Number             _   Branch Enter System Service
    _   Program Interrupts(Page Faults/Abends)
    _   Slip/Per Interrupts
    _   Dispatch                   _   Machine Interrupts
    _   Restart Interrupts
    _   Lock Suspensions
    _   Recovery Event             _   User Trace
    _   Program Call, Program Return, Program Transfer

                      Enter Key To Process Selections
                      PF3/END To Return To Previous Panel
```

Figure 7-2-2.

I tab down to "SVCs by Number and put an "I" next to it. Then, I am presented with a list of all the SVCs, as in Figure 7-2-3, below:



```
---z/XPF----------------SELECT SVC'S--------------------------------- Row 1 of 86
OPTION   ===> _________________________________________________ SCROLL ===> PAGE

   Use any non-blank character to specify individual svc's.

                           PF3/END TO SAVE SELECTIONS
 _    00  EXCP                 I  01  WAIT              _   02  POST
 _    03  EXIT                 I  04  GETMAIN           _   05  FREEMAIN
 _    06  LINK                 _  07  XCTL              _   08  LOAD
 _    09  DELETE               _  10  GETMAIN-FREEMAI   _   11  TIME
 _    12  SYNCH                _  13  ABEND             _   14  SPIE
 _    15  ERREXCP              _  16  PURGE             _   17  RESTORE
 _    18  BLDL-FIND            _  19  OPEN              _   20  CLOSE
 _    21  STOW                 _  22  OPEN TYPE=J       _   23  CLOSE TYPE=T
 _    24  DEVTYPE              _  25  TRKBAL            _   26  CATALOG-LOCATE
 _    27  OBTAIN               _  28  RESERVED          _   29  SCRATCH
 _    30  RENAME               _  31  FEOV              _   32  REALLOC
 _    33  IOHALT               _  34  MGCR              _   35  WTO-WTOR
 _    36  WTL                  _  37  SEGLD-SEGWT       _   38  RESERVED
 _    39  LABEL                _  40  EXTRACT           _   41  IDENTIFY
 _    42  ATTACH               _  43  CIRB              _   44  CHAP
 _    45  OVRLYBRCH            _  46  TTIMER-STIMER     _   47  STIMER-STIMERM
 _    48  DEQ                  _  49  RESERVED          _   50  RESERVED
 _    51  SNAP-SNAPX           _  52  RESTART           _   53  RELEX
 _    54  DISABLE              _  55  EOV               _   56  ENQ
 _    57  FREEDBUF             _  58  RELBUF-REQBUF     _   59  OLTEP
 _    60  STAE-ESTAE           _  61  IKJEGS6A          _   62  DETACH
 _    63  CHKPT                _  64  RDJFCB            _   65  RESERVED
 _    66  BTAMTEST             _  67  RESERVED          _   68  SYNADAF-SYNADRLS
 _    69  BSP                  _  70  GSRV              _   71  ASGNBFR
 _    72  IEAVVCTR             _  73  SPAR              _   74  DAR
 _    75  DQUEUE               _  76  IFBSVC76          _   77  RESERVED
 _    78  LSPACE               _  79  STATUS            _   80  RESERVED
 _    81  SETPRT               _  82  RESERVED          _   83  SMFWTM
 _    84  GRAPHICS             _  85  *UNDEFINED*       _   86  ATLAS
 _    87  DOM                  _  88  RESERVED          _   89  RESERVED
 _    90  RESERVED             _  91  VOLSTAT           _   92  TCBEXCP
 _    93  TGET/TPUT/TPG        _  94  STCC              _   95  SYSEVENT
 _    96  STAX                 _  97  *UNDEFINED*       _   98  PROTECT
 _    99  DYNALLOC             _  100 *UNDEFINED*       _   101 QTIP
 _    102 AQCTL                _  103 XLATE             _   104 TOPCTL
 _    105 IMGLIB               _  106 RESERVED          _   107 MODESET
 _    108 RESERVED             _  109 ESR               _   110 RESERVED
```

Figure 7-2-3.

If you want to see more of the available SVCs, you can page down to see them. In this example I have made a selection of the WAIT and GETMAIN SVCs.

That's enough filtering for an example. I'll use PF3 to back all the way out to the main Detail Report panel.

Now, to generate the report I'll enter Option 3 as in Figure 7-2-4 below:

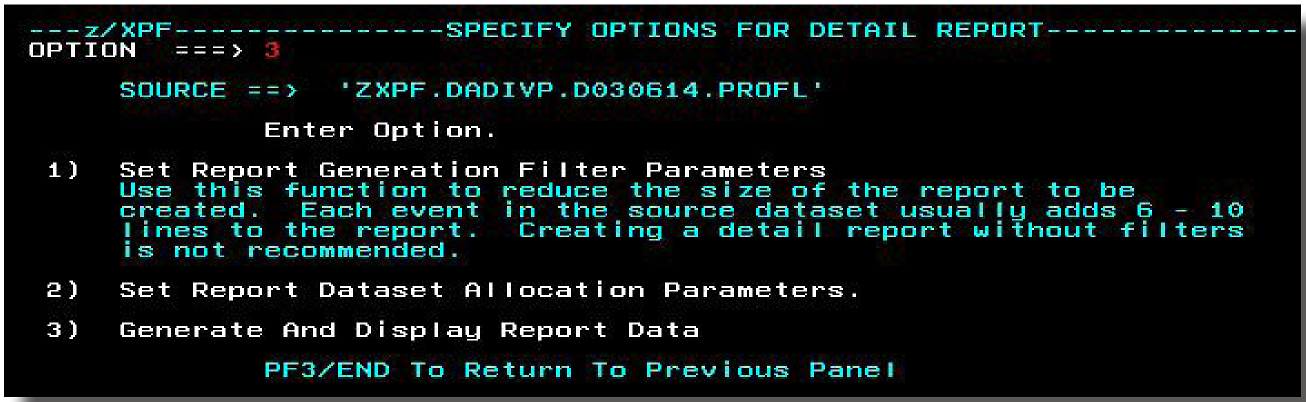

```
---z/XPF----------------SPECIFY OPTIONS FOR DETAIL REPORT--------------
OPTION   ===> 3

     SOURCE ==>   'ZXPF.DADIVP.D030614.PROFL'

             Enter Option.

 1)   Set Report Generation Filter Parameters
      Use this function to reduce the size of the report to be
      created.  Each event in the source dataset usually adds 6 - 10
      lines to the report.  Creating a detail report without filters
      is not recommended.

 2)   Set Report Dataset Allocation Parameters.

 3)   Generate And Display Report Data

             PF3/END To Return To Previous Panel
```

Figure 7-2-4.

After I press enter, z/XPF filters my report. It goes through the capture dataset and pulls out what I asked for. While it's processing, you'll see an intermediate screen like the one in Figure 7-2-5 below:

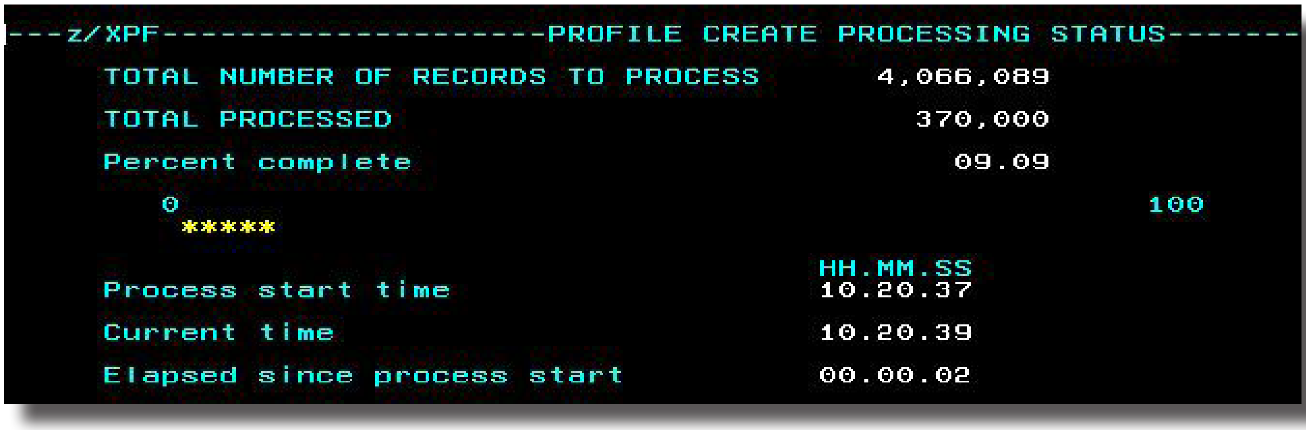
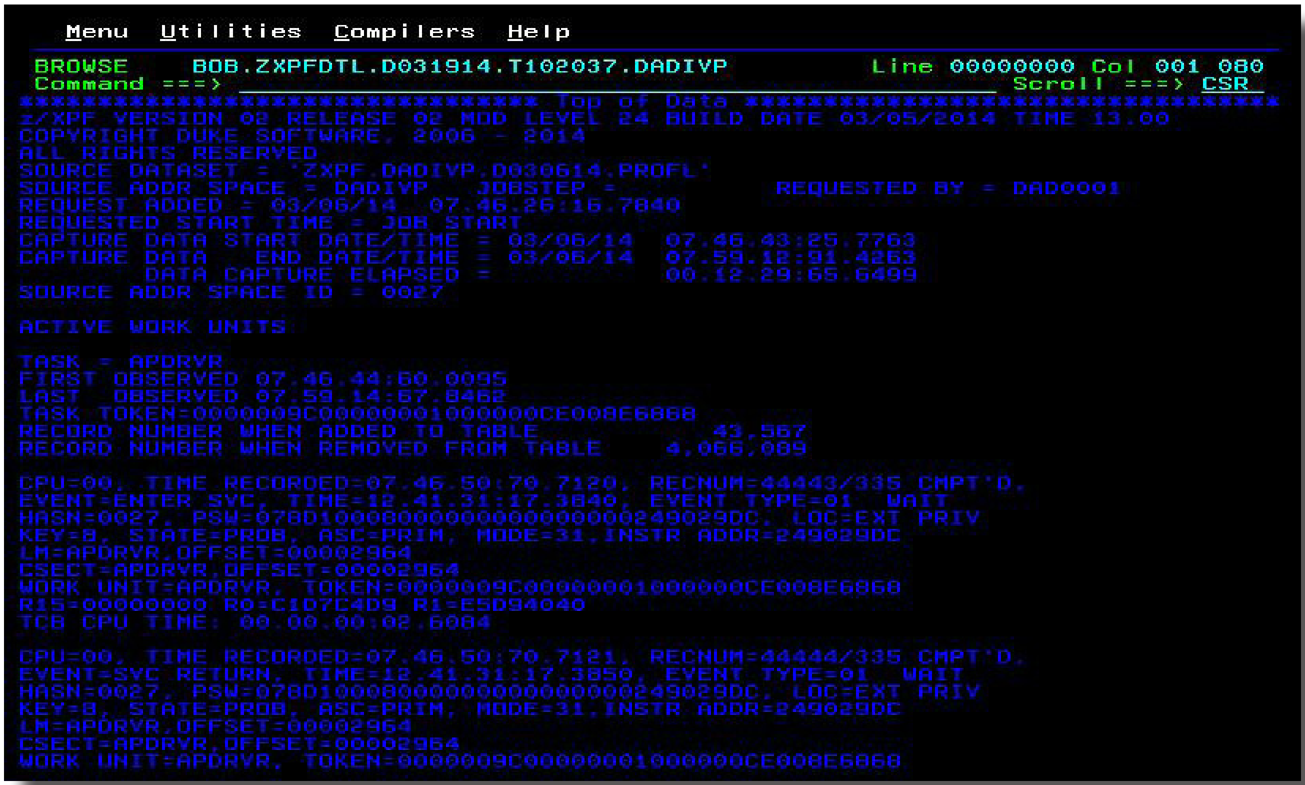

```
---z/XPF--------------------PROFILE CREATE PROCESSING STATUS-------
     TOTAL NUMBER OF RECORDS TO PROCESS        4,066,089

     TOTAL PROCESSED                             370,000

     Percent complete                              09.09

        0                                                      100
         *****

                                              HH.MM.SS
     Process start time                       10.20.37

     Current time                             10.20.39

     Elapsed since process start              00.00.02
```

Figure 7-2-5.

Staring at this screen is like watching the laundry go around in the dryer. If you have a large report for z/XPF to parse, find something else to do.

Once the sorting is finished, z/XPF presents the filtered report. See Figure 7-2-6 below:I



```
   Menu  Utilities  Compilers  Help
 BROWSE    BOB.ZXPFDTL.D031914.T102037.DADIVP           Line 00000000 Col 001 080
 Command ===> ________________________________________________ Scroll ===> CSR
********************************* Top of Data **********************************
z/XPF VERSION 02 RELEASE 02 MOD LEVEL 24 BUILD DATE 03/05/2014 TIME 13.00
COPYRIGHT DUKE SOFTWARE, 2006 - 2014
ALL RIGHTS RESERVED
SOURCE DATASET = 'ZXPF.DADIVP.D030614.PROFL'
SOURCE ADDR SPACE = DADIVP   JOBSTEP =           REQUESTED BY = DAD0001
REQUEST ADDED = 03/06/14  07.46.26:16.7840
REQUESTED START TIME = JOB START
CAPTURE DATA START DATE/TIME = 03/06/14  07.46.43:25.7763
CAPTURE DATA   END DATE/TIME = 03/06/14  07.59.12:91.4263
        DATA CAPTURE ELAPSED =            00.12.29:65.6499
SOURCE ADDR SPACE ID = 0027

ACTIVE WORK UNITS

TASK = APDRVR
FIRST OBSERVED 07.46.44:60.0095
LAST  OBSERVED 07.59.14:67.8462
TASK TOKEN=0000009C0000000001000000CE008E6868
RECORD NUMBER WHEN ADDED TO TABLE              43,567
RECORD NUMBER WHEN REMOVED FROM TABLE       4,066,089

CPU=00, TIME RECORDED=07.46.50:70.7120, RECNUM=44443/335 CMPT'D.
EVENT=ENTER SVC, TIME=12.41.31:17.3840, EVENT TYPE=01  WAIT
HASN=0027, PSW=078D1000800000000000000000249029DC, LOC=EXT PRIV
KEY=8, STATE=PROB, ASC=PRIM, MODE=31,INSTR ADDR=249029DC
LM=APDRVR,OFFSET=00002964
CSECT=APDRVR,OFFSET=00002964
WORK UNIT=APDRVR, TOKEN=0000009C0000000001000000CE008E6868
R15=00000000 R0=C1D7C4D9 R1=E5D94040
TCB CPU TIME: 00.00.00:02.6084

CPU=00, TIME RECORDED=07.46.50:70.7121, RECNUM=44444/335 CMPT'D.
EVENT=SVC RETURN, TIME=12.41.31:17.3850, EVENT TYPE=01  WAIT
HASN=0027, PSW=078D1000800000000000000000249029DC, LOC=EXT PRIV
KEY=8, STATE=PROB, ASC=PRIM, MODE=31,INSTR ADDR=249029DC
LM=APDRVR,OFFSET=00002964
CSECT=APDRVR,OFFSET=00002964
WORK UNIT=APDRVR, TOKEN=0000009C0000000001000000CE008E6868
```

Figure 7-2-6.

In the report we see some preamble text and then the first Event Record of the report. This report goes on for some pages, and I elected NOT to show you the whole report, for brevity's sake.

Let's repeat the steps I took in creating this detail report:

1. I selected a single task: APDRVR.
2. I selected two SVCs within that task.
3. I ran my report.

At this point, I could page up/down within the report or use the Find command to search for specific things. There's a LOT of information in each "formatted" Trace Record.

If you have decided to use z/XPF's ZXPFTRAC macro you could have signalled events in your code that normally don't create a Trace Record. This facility allows you to create your own "Trace Records", and these records appear ONLY in Detail Reporting.

Note: Remember that z/XPF's filters persist across Detail Reports in the same reporting session. So, if you've set filters for a previous report and you create another report then the previous filters remain active for the next report as well. This is by design, because as you drill down using filters, you may want to retain your existing filters, and refine your criteria,.

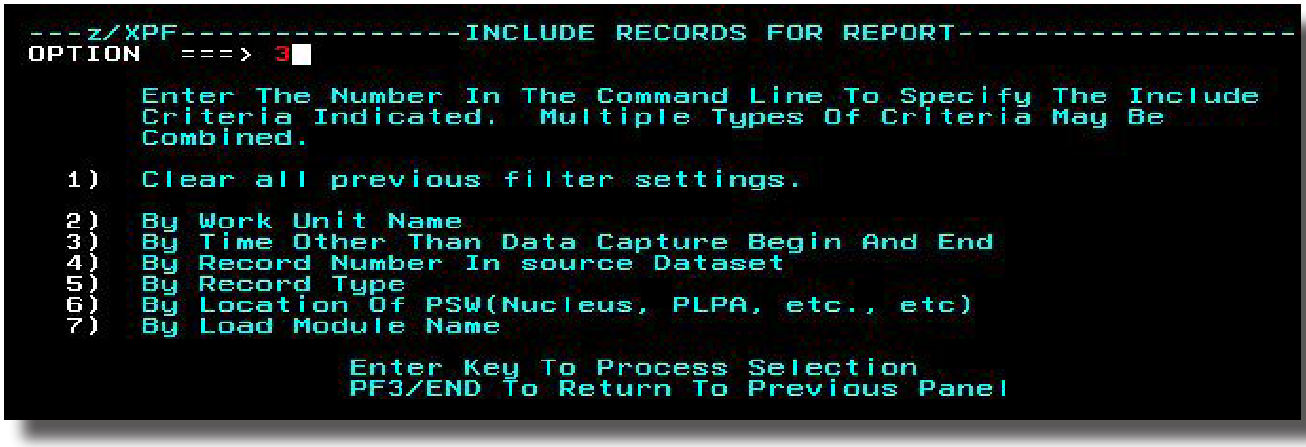
On the other hand, if you want to delete all filters you can press "PF3" to get back to the

SPECIFY OPTIONS FOR DETAIL REPORT (Figure 7-1), pick Option 1 to “Set Report Generations...” (Figure 7-3) and use Option 1 from that panel to clear your filters. Alternatively, you can free the allocated capture dataset, re-allocate it and go back into Detail Reporting. That’s the “brute force” method of clearing filters.

7-3 Detail reporting by Time other than Data Capture Begin and End

Each event that is contained in the source capture dataset has a timestamp set at the time the event was copied to a z/XPF-owned buffer. Detail reports normally include every record without regard to time-stamp, but you can influence that in the following panel when you select Option 3 in Figure 7-3 “INCLUDE RECORDS FOR REPORT”.

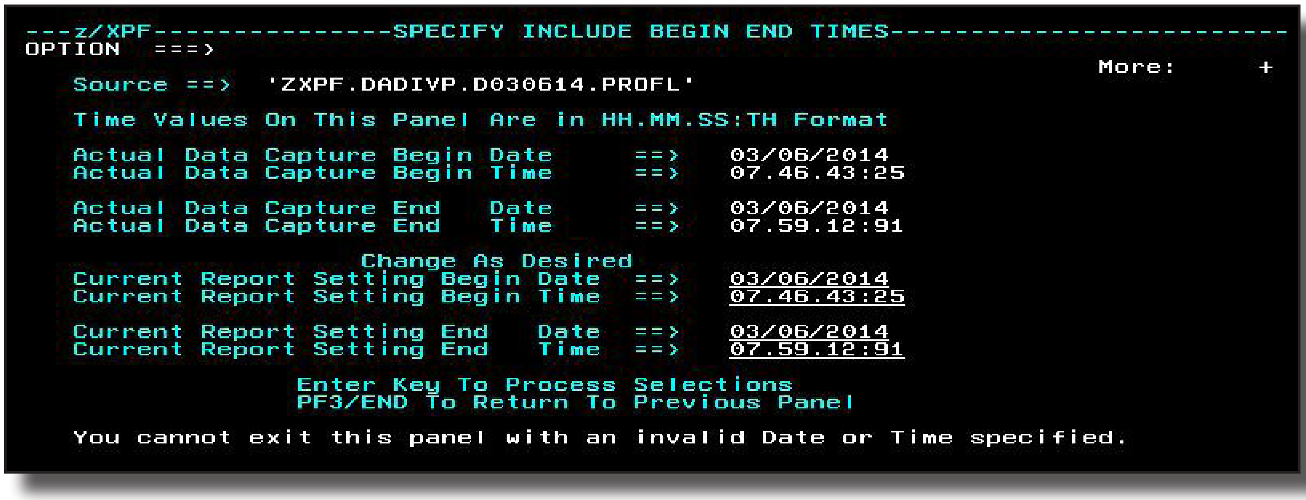
I’ll navigate back to this panel, and select Option 3. See Figure 7-3-1 below:



```
---z/XPF---------------INCLUDE RECORDS FOR REPORT------------------
OPTION  ===> 3

        Enter The Number In The Command Line To Specify The Include
        Criteria Indicated.  Multiple Types Of Criteria May Be
        Combined.

   1)   Clear all previous filter settings.

   2)   By Work Unit Name
   3)   By Time Other Than Data Capture Begin And End
   4)   By Record Number In source Dataset
   5)   By Record Type
   6)   By Location Of PSW(Nucleus, PLPA, etc., etc)
   7)   By Load Module Name

                     Enter Key To Process Selection
                     PF3/END To Return To Previous Panel
```

Figure 7-3-1.

Next, I press Enter, and see the panel in Figure 7-3-2 below:



```
---z/XPF---------------SPECIFY INCLUDE BEGIN END TIMES------------------------
OPTION  ===>
                                                               More:     +
    Source ==>  'ZXPF.DADIVP.D030614.PROFL'

    Time Values On This Panel Are in HH.MM.SS:TH Format

    Actual Data Capture Begin Date     ==>   03/06/2014
    Actual Data Capture Begin Time     ==>   07.46.43:25

    Actual Data Capture End   Date     ==>   03/06/2014
    Actual Data Capture End   Time     ==>   07.59.12:91

                 Change As Desired
    Current Report Setting Begin Date  ==>   03/06/2014
    Current Report Setting Begin Time  ==>   07.46.43:25

    Current Report Setting End   Date  ==>   03/06/2014
    Current Report Setting End   Time  ==>   07.59.12:91

                 Enter Key To Process Selections
                 PF3/END To Return To Previous Panel

    You cannot exit this panel with an invalid Date or Time specified.
```

Figure 7-3-2.

The top variables in this panel state the actual beginning and ending date and time for your data capture session.

The bottom variables in this panel can be changed from the original values in the top section of the panel. When you change these values, periods, forward slashes and colons must be in the correct positions.

You may not alter the time-stamp parameters to values EARLIER or LATER than the actual Data Capture beginning and ending time-stamps (Yep, I tried it). If you do this, z/XPF will return a "DATE/TIME INVALID" message.

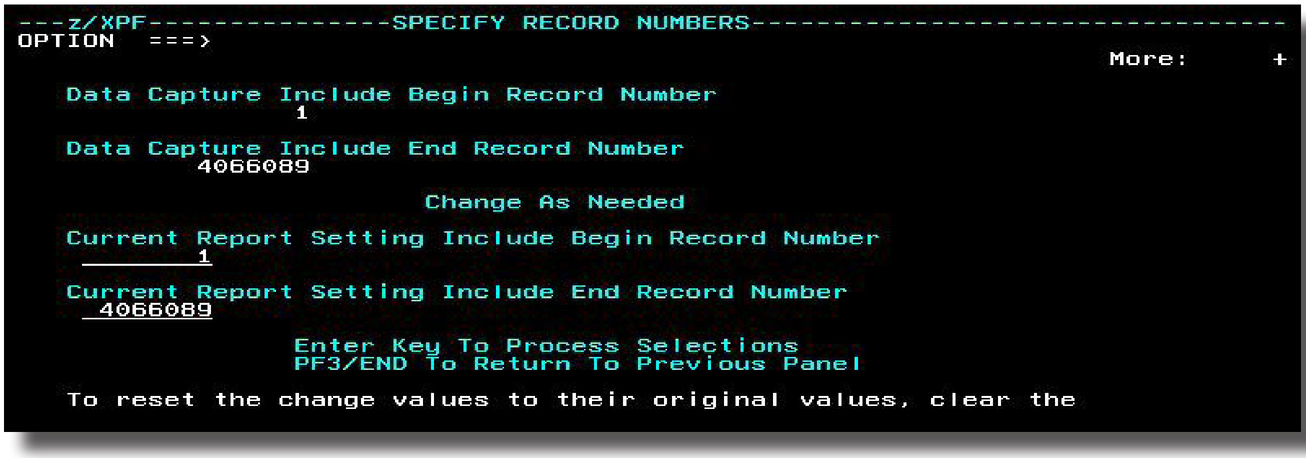
If you end up making a mistake on this panel, you can blank out the fields, press Enter, and z/XPF will return the original values to the panel.

There's a cryptic word saying "MORE +". If you want to see more of the explanatatory text at the bottom of this panel, press PageDown (typically PF8) to view it.

7-4 Detail reporting by Record Number In source Dataset

In addition to a timestamp, each record contains a unique, sequential record number. You may filter by record number in the report.

When you select Option 4 in the panel entitled, "INCLUDE RECORDS FOR REPORT" in Figure 7-10, then the panel below in Figure 7-4-1 will be displayed:



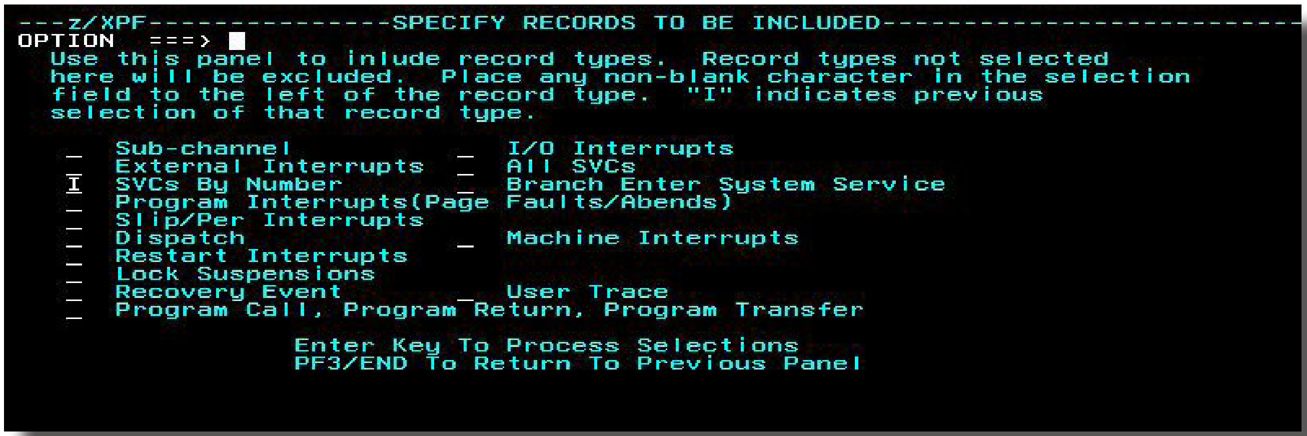
```
---z/XPF---------------SPECIFY RECORD NUMBERS----------------------------------
OPTION   ===>
                                                                    More:     +
   Data Capture Include Begin Record Number
                 1
   Data Capture Include End Record Number
           4066089
                              Change As Needed
   Current Report Setting Include Begin Record Number
    ________1
   Current Report Setting Include End Record Number
     4066089
                 Enter Key To Process Selections
                 PF3/END To Return To Previous Panel
   To reset the change values to their original values, clear the
```

Figure7-4-1.

Overtype either one of the bottom two variables on the panel to alter the beginning and end record numbers for the report.

7-5 Detail reporting by Record Type

Use Option 5 from Figure 7-10 to get “here”. Use this filter when only specific event types are to be examined. Multiple record types may be selected. See Figure 7-5-1 below:



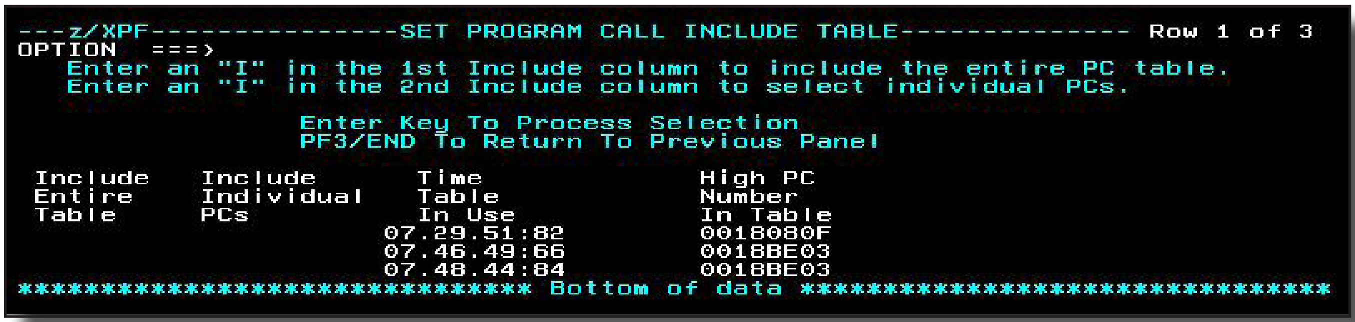
```
---z/XPF---------------SPECIFY RECORDS TO BE INCLUDED-------------------------
OPTION  ===> 
   Use this panel to inlude record types.  Record types not selected
   here will be excluded.  Place any non-blank character in the selection
   field to the left of the record type.  "I" indicates previous
   selection of that record type.

     _   Sub-channel              _  I/O Interrupts
     _   External Interrupts      _  All SVCs
     I   SVCs By Number           _  Branch Enter System Service
     _   Program Interrupts(Page Faults/Abends)
     _   Slip/Per Interrupts
     _   Dispatch                 _  Machine Interrupts
     _   Restart Interrupts
     _   Lock Suspensions
     _   Recovery Event           _  User Trace
     _   Program Call, Program Return, Program Transfer

                      Enter Key To Process Selections
                      PF3/END To Return To Previous Panel
```

Figure 7-5-1.

You can see that I still have an “INCLUDE” set for SVC by number on this panel. I’ll remove that filter by overtyping the “I” with a blank.

You may select as many or as few of the record types as desired. There are numerous other sub-menus that allow you to filter exactly the events you want to report on.

Setting an “INCLUDE” to most of these filters returns only the message, “INCLUDE PARAMETERS SET”, and normally that’s all you have to do. However, if you INCLUDE “Program Call, Program Return, Program Transfer”, z/XPF shows you a panel similar to the one below in Figure 7-5-2, below:

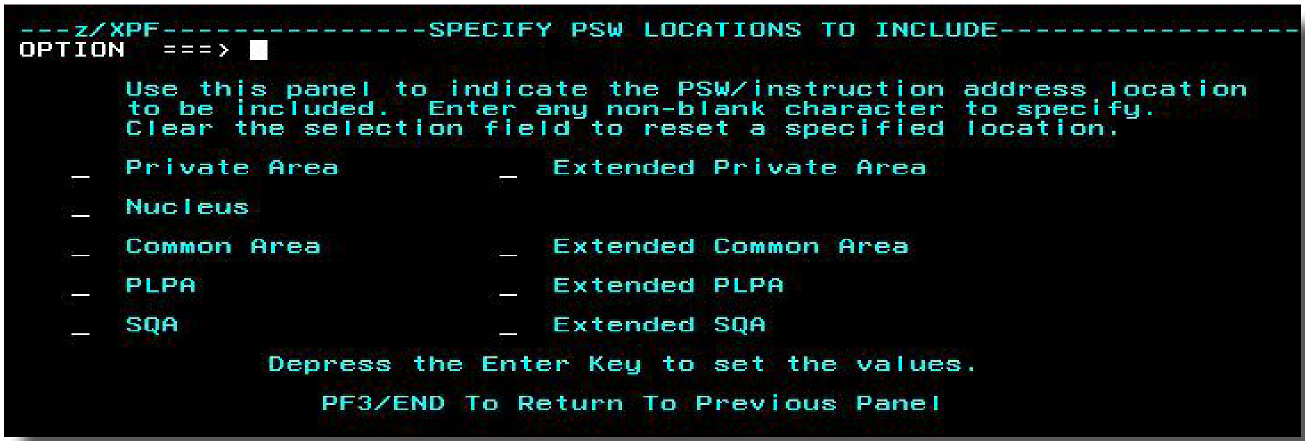


```
---z/XPF---------------SET PROGRAM CALL INCLUDE TABLE-------------- Row 1 of 3
OPTION  ===>
    Enter an "I" in the 1st Include column to include the entire PC table.
    Enter an "I" in the 2nd Include column to select individual PCs.

                      Enter Key To Process Selection
                      PF3/END To Return To Previous Panel

  Include    Include      Time              High PC
  Entire     Individual   Table             Number
  Table      PCs          In Use            In Table
                        07.29.51:82         0018080F
                        07.46.49:66         0018BE03
                        07.48.44:84         0018BE03
******************************* Bottom of data ********************************
```

Figure 7-5-2.

7-6 Detail reporting by Location Of PSW

z/XPF can identify the virtual storage location of an instruction address contained in an Event record. To do this, you select Option 6 in the Include Records For Report panel (see Figure 7-10).You may restrict events in the report to one or more virtual storage locations. The panel below in Figure 7-6-1 shows how to set location filters:

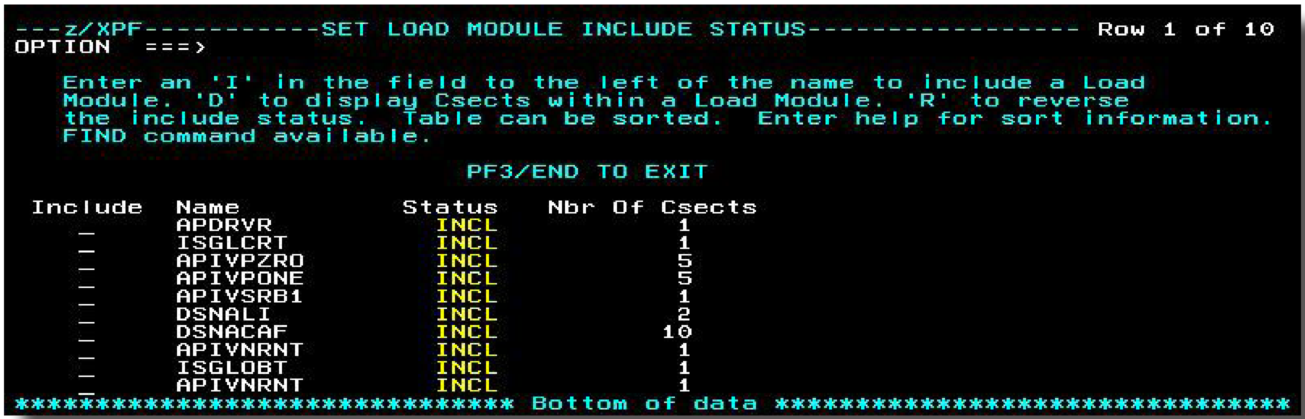


```
---z/XPF----------------SPECIFY PSW LOCATIONS TO INCLUDE-----------------
OPTION   ===>

      Use this panel to indicate the PSW/instruction address location
      to be included.  Enter any non-blank character to specify.
      Clear the selection field to reset a specified location.

   _  Private Area            _  Extended Private Area

   _  Nucleus

   _  Common Area             _  Extended Common Area

   _  PLPA                    _  Extended PLPA

   _  SQA                     _  Extended SQA

              Depress the Enter Key to set the values.

                 PF3/END To Return To Previous Panel
```

Figure 7-6-1.

7-7 Detail reporting by Load Module Name

z/XPF allows you to include or exclude load module names in a Detail Report. To do that select Option 7 in the Include Records For Report panel (see Figure 7-10). That panel is presented below in Figure 7-5-2:



```
---z/XPF-----------SET LOAD MODULE INCLUDE STATUS----------------- Row 1 of 10
OPTION   ===>
   Enter an 'I' in the field to the left of the name to include a Load
   Module. 'D' to display Csects within a Load Module. 'R' to reverse
   the include status.  Table can be sorted.  Enter help for sort information.
   FIND command available.

                            PF3/END TO EXIT

 Include    Name          Status    Nbr Of Csects
    _       APDRVR          INCL           1
    _       ISGLCRT         INCL           1
    _       APIVPZRO        INCL           5
    _       APIVPONE        INCL           5
    _       APIVSRB1        INCL           1
    _       DSNALI          INCL           2
    _       DSNACAF         INCL          10
    _       APIVNRNT        INCL           1
    _       ISGLOBT         INCL           1
    _       APIVNRNT        INCL           1
******************************* Bottom of data ********************************
```

Figure 7-7-1.

7-8 Tips And Techniques For Creating Detail Reports

Our first and best advice is to *use the filter setting logic to restrict the amount of output in the report.* If this isn't done, the report will contain so much data that it is almost impossible to find the events you are interested in. Here are some ideas to think about as you try to isolate events in your z/XPF reports. You can consider these "recipes" for various kinds of enquiries.

Remember, filter settings are persistent. If you want to change your approach you may have to back all the way out to the "INCLUDE RECORDS FOR REPORT" panel, and issue Option 1 to "Clear all previous filter settings".

Reporting on device I/O Activity:

- Go into the filter setting main panel and select Option 5, for record types.
- Select Sub-channel and I/O Interrupts.
- Press PF3/END back to the main filter setting panel.
- Depending upon the volume of I/O activity in the capture dataset, set another filter either by time or by record number.
- Create the report.

When you are browsing your report:

- Enter a FIND command for "SUBCH". This should position to the first START SUB-Channel event in the report.
- Note the device number, then page down in the report to the I/O Interrupt with a matching device number.
- Repeat this process as necessary.

To get a "feel" for how I/O operations affect a program:

- Go back into the filter setting logic and add include settings for SVCs EXCP, EXCPVR, and WAIT.
- On the record types main panel, select Dispatch.
- Then, request the report again.

The WAIT and the Dispatch events will add events of those types not related to the I/O activity.

When you are browsing your report:

- Search for "EXCP".
- The sequence of events should be the EXCP/EXCPVR, the WAIT, the Start Sub-channel, the I/O Interrupt, and finally the Dispatch at the PSW WAIT address.

Reporting on Page Faults and Program Interrupts:

- Under Option 5 In the filter setting logic, select "Program Interrupts" (Page Faults/Abends).
- Set the Work Unit INCLUDE flag as needed for the desired Work Unit.
- Set the load module name if known.
- Request the report.

When you are browsing your report:

- Locate page faults by searching for "PAGE TRANS" or "PAGE TRANS EXCEPTION".
- Each of these records identifies a program interrupt that occurred due to a page fault.

Reporting on Abends in a program:

If a Summary Report indicated Recovery events for a Work Unit, then:

- Set the INCLUDE flag for the Work Unit in the filter logic.
- Next, in the record types panel (Option 5), select Program Interrupts and Recovery events.
- Create a report.

When you are browsing your report:

- Search for the keyword "RECOVERY" to locate the first event. WORD 3 of the formatted Recovery event data will show the type of abend encountered.
- Search back up to the Program Interrupt event just preceding the Recovery event.
- Make sure the description of the abend matches the hex value in WORD 3 of the formatted Recovery event.
- Make note of the record number and the time of the Program Interrupt.

If the recovery routine in effect at the time of the abend has a retry address, and the load module where the retry from the abend will occur is known, then:

- Go into the filter logic and set the load module name for the Program Interrupt and the retry.
- Add Dispatch to the event types.
- Create a report.

When you are browsing your report:

- Go back into the report and locate the Program Interrupt again.
- Scroll forward/down to the Dispatch.
- You can now see both the event that marks the abend and the event that marks the retry after the abend.

If you would like to understand what events are generated as part of recovery termination manager and the application recovery routine, use either the timestamps from the abend and the retry event, or the record numbers.

- Go back into the filter setting logic and remove all the previous filters.
- Set either time or record number filtering to the values contained in the records. Create a report.

When you are browsing your report:

- Look at all of the events between the abend and the dispatch at the retry address.
- Some applications are coded to use an abend as a “signaling” mechanism to indicate that a problem exists, and action is required.
- If an application “rotates” through the abend/recovery logic a great deal, then there will be a higher price paid in CPU cycles and lost elapsed time.

To investigate lock suspension times:

- In the record types filter panel under Option 5, select Lock Suspensions.
- Create a report.

When you are browsing your report:

- Search the report for the load module name appearing in the event description for the lock suspend event.

If there are multiple events with different load module names, it may be easier to take them one at a time. So:

- In the records type panel, add Dispatch to the selected Lock Suspensions.
- In the load module filtering, add one or more of the names noted from the first report.
- Create a report.

When you are browsing your report:

- Find the first lock suspend event description.
- There may or may not be dispatch events in the report prior to the lock suspend. Immediately after the lock suspend event should be a Dispatch record.
- Make sure that the load module name and offset values match on both records.

Time Spent Waiting For A Latch:

As of the current release, z/XPF cannot identify the load module in the user's application that was suspended waiting for a Latch. However, it can identify the Program Call that transfers control from the ISGLOBTS Nucleus module to the GRS address space, and the Program Return to the same address in ISGLOBTS when the Latch is obtained.

From these two events, the user can get to within milliseconds of the actual elapsed time for the latch obtain request. The timestamp on the Program Call event is the time during the data capture interval when z/XPF found the event, and copied it to z/XPF's buffer. z/XPF's interval value (if the default is taken) is 20 milliseconds. Thus, z/XPF can get the time to within 20 milliseconds of the actual time of the latch obtain request. The same is true for the Program Return.

[As of April 2009 on a z/OS 1.10, the PC number used for the space-switch call to GRS is "0000010E". The "eye-catcher" from the target address in the Program Call indicates a module named ISGLRTR. It is doubtful if either the PC number, the calling module, or the target module will change, but if and when it does, this document will be updated with the new information.]

To view the events that were observed and used to compute Latch time in the Summary Reports:

- The report dataset will be easier to read if the INCLUDE flag is set for the Work Unit that experienced the latch delay time.
- Set ISGLOBTS in the load module name table.
- Create a report.

When you are browsing your report:

- Locate the Program Call and the Program Return.
- Note the time, or the record numbers.
- Go back into the filter logic and remove ISGLOBTS from the load module table.
- Set either the time or the record numbers for the Program Call and the Program Return.
- Create another report.

When you are browsing your report:

- Search for the Program Call you are interested in.
- Five or six events past the Program Call is a PC/BR EN SYS event record with a service id on the right of 011E IEAVPSE/IEAVXFR.
- The time on the event is the time the Work Unit entered Pause.
- The next record in the report should be a Dispatch record.
- The time on this record is the time the Work Unit obtained the Latch.

ENQ Contention using ISGENQ

[As of April 2009 on a z/OS 1.10 system, the PC number used for the space-switch call to GRS is “0000011A”. The module receiving control is ISGGRT. It is doubtful if either the PC number, or the target module will change, but if and when one or both do change, this document will be updated with the new information.]

When ISGENQ is used to obtain an ENQ, the Program Call to enter the function is executed directly from the program that is requesting the resource (There is no stub code that branches to a Nucleus module first, as in Latch Obtain).

Here is the sequence of events observed to obtain an ENQ on a resource when the resource is available is:

- A Program Call is executed using PC number 0000011A
- A PC/BR EN SYS event is executed where the service id for the event is 014C ISGENQ. The return address on the event is the same as the return address on the PC event.
- A Program Return is executed. The return address, load module, and csect is the same as the Program Call

The sequence of events observed to obtain an ENQ on a resource when the resource is not immediately available is this:

- A Program Call is executed using PC number 0000011A.
- A PC/BR EN SYS event is executed where the service ID is “0001 WAIT” (The return address on this event identifies load module ISGGHOM).
- A Dispatch event for load module ISGGHOM is executed at the same instruction address as the WAIT event above.
- PC/BR EN SYS event is executed where the service id for the event is 014C ISGENQ (The return address on the event is the same as the return address on the PC event).
- A Program Return is executed to the Program Call’s return address, csect and load module.

7-9 Understanding the Fields in Event Records

Here is an "index" into the meaning of each of the fields you'll find in z/XPF's Event Records. The terms are arranged in order of appearance, line by line.

1st line of each Event Record:

- **CPU=nn** CPU processor number where the event occurred.
- **TIME RECORDED=** The time that the record was moved from the processing buffer to z/XPF's buffer.
- **RECNUM=nnnn/nnnn** The record number assigned to this Event Record appears on the left side of the slash. The Interval number when the Event Record was copied from the z/OS buffer is on the right side of the slash.
- **CMPT'D** This indicates that z/XPF was able to record the event with no errors.
- **ABND'D** This indicates the z/XPF server logic encountered an abend during its interval processing while recording the event.

2nd line of each Event Record:

EVENT= The type of event is identified as one of the following:

- START SUBCH
- MODIFY SUBCH'
- HALT SUBCH'
- CLEAR SUBCH'
- RESUME SUBCH'
- SIGNAL ADPTR'
- CANCEL SUBCH'
- GENERAL EXTR'
- EMRGCY SIGNL'
- SRVICE SIGNL'
- CALL '
- CLOCK COMPRT'
- ENTER SVC '
- SVC RETURN '
- PC/BR EN SYS'
- TASK DISPTCH'
- SRB DISPATCH'
- SUSP SRB DSP'
- WAIT DISPTCH'
- PGM INTRRUPT'
- SLIP/PER EVT'
- I/O INTRRUPT'
- MCH INTRRUPT'

- RESTART INTR'
- ALT CPU RCVY'
- RECOVERY '
- LOCK SUSPEND'
- *ALTER TRACE '
- TIME '

TIME= The time value from the actual event.

EVENT TYPE= This is a further explanation of the EVENT= field, and could be considered a sub-type description. Not all events will have this field. And the sub-types are specific to events.

If an external interrupt, it might be one of these:

- INTERRUPT KEY '
- CLOCK COMPARATR'
- CPU TIMER '
- MALFNCTN ALERT'
- EMRGENCY SIGNAL'
- ETR'
- SERVICE SIGNAL '

If the Event Type is an SVC, it will be one of these (Yes, we're going to list the SVCs):

- 00 EXCP '
- 01 WAIT '
- 02 POST '
- 03 EXIT '
- 04 GETMAIN '
- 05 FREEMAIN '
- 06 LINK '
- 07 XCTL '
- 08 LOAD '
- 09 DELETE '
- 10 GETMAIN-FREEMAIN'
- 11 TIME '
- 12 SYNCH '
- 13 ABEND '
- 14 SPIE '
- 15 ERREXCP '
- 16 PURGE '
- 17 RESTORE '
- 18 BLDL-FIND '
- 19 OPEN '
- 20 CLOSE '

- 21 STOW ‘
- 22 OPEN TYPE=J ‘
- 23 CLOSE TYPE=T ‘
- 24 DEVTYPE ‘
- 25 TRKBAL ‘
- 26 CATALOG-LOCATE ‘
- 27 OBTAIN ‘
- 28 RESERVED ‘
- 29 SCRATCH
- 30 RENAME
- 31 FEOV
- 32 REALLOC
- 33 IOHALT
- 34 MGCR
- 35 WTO-WTOR
- 36 WTL
- 37 SEGLD-SEGWT
- 38 RESERVED
- 39 LABEL
- 40 EXTRACT
- 41 IDENTIFY
- 42 ATTACH
- 43 CIRB
- 44 CHAP
- 45 OVRLYBRCH
- 46 TTIMER-STIMER
- 47 STIMER-STIMERM
- 48 DEQ
- 49 RESERVED
- 50 RESERVED
- 51 SNAP-SNAPX
- 52 RESTART
- 53 RELEX
- 54 DISABLE
- 55 EOV
- 56 ENQ
- 57 FREEDBUF
- 58 RELBUF-REQBUF
- 59 OLTEP
- 60 STAE-ESTAE
- 61 IKJEGS6A
- 62 DETACH
- 63 CHKPT
- 64 RDJFCB
- 65 RESERVED
- 66 BTAMTEST

- 67 RESERVED
- 68 SYNADAF-SYNADRLS
- 69 BSP
- 70 GSRV
- 71 ASGNBFR
- 72 IEAVVCTR
- 73 SPAR
- 74 DAR
- 75 DQUEUE
- 76 IFBSVC76
- 77 RESERVED
- 78 LSPACE
- 79 STATUS
- 80 RESERVED
- 81 SETPRT
- 82 RESERVED
- 83 SMFWTM
- 84 GRAPHICS
- 86 ATLAS
- 87 DOM
- 88 RESERVED
- 89 RESERVED
- 90 RESERVED
- 91 VOLSTAT
- 92 TCBEXCP
- 93 TGET/TPUT/TPG
- 94 STCC
- 95 SYSEVENT
- 96 STAX
- 98 PROTECT
- 99 DYNALLOC
- 101 QTIP ‘
- 102 AQCTL ‘
- 103 XLATE ‘
- 104 TOPCTL ‘
- 105 IMGLIB ‘
- 106 RESERVED ‘
- 107 MODESET ‘
- 108 RESERVED ‘
- 109 ESR ‘
- 110 RESERVED ‘
- 112 PGRLSE ‘
- 113 PAGING SERVICES ‘
- 114 EXCPVR ‘
- 115 RESERVED ‘
- 116 ESR ‘

- 117 DEBCHECK ‘
- 118 RESRVED ‘
- 119 TESTAUTH ‘
- 120 GETMAIN-FREEMAIN’
- 121 VSAM ‘
- 122 ESR ‘
- 123 PURGEDQ ‘
- 124 TPIO ‘
- 125 EVENTS ‘
- 126 RESERVED ‘
- 127 RESERVED ‘
- 128 RESERVED ‘
- 129 RESERVED ‘
- 130 RACHECK ‘
- 131 RACINIT ‘
- 132 RACLIST ‘
- 133 RACDEF ‘
- 134 RESERVED ‘
- 135 RESERVED ‘
- 136 RESERVED ‘
- 137 ESR ‘
- 138 PGSER ‘
- 139 CVAF ‘
- 143 GENKEY ‘
- 146 BPSEVC ‘
- 148 and above, up to 255 are user defined SVC?s.

If the Event Record describes a Program Interrupt (“**EVENT=PGM INTRRUPT**”), then you will see one of the following

- OPERATION EXCEPTION’
- PRIV OPER EXCEPTION’
- EXECUTE EXCEPTION’
- PROTECTION EXCEPTION’
- ADDRESSING EXCEPTION’
- SPECIFICATION EXCPTN’
- DATA EXCEPTION ‘
- FXD POINT OVR EXCPTN’
- FXD POINT DIV EXCPTN’
- DEC OVFLOW EXCEPTION’
- DEC DIVIDE EXCEPTION’
- HFP EXP OVRFL EXCPTN’
- HFP EXP UDRFL EXCPTN’
- HFP SIGNFC EXCEPTION’
- HFP DIVIDE EXCEPTION’
- SGMT TRANS EXCEPTION’

- PAGE TRANS EXCEPTION’
- TRANS SPEC EXCEPTION’
- SPECIAL OP EXCEPTION’
- OPERAND EXCEPTION’
- TRACE TABLE EXCPTION’
- SPACE SWITCH EVENT ‘
- HFP SQ ROOT EXCPTION’
- PC TRANS SPEC EXCPTN’
- AFX TRNSL EXCEPTION’
- ASX TRNSL EXCEPTION’
- LX TRNSL EXCEPTION’
- EX-TRANSL EXCEPTION’
- PRIM AUTHORITY XCPTN’
- SCNDRY AUTH EXCPTION’
- ALET SPFC EXCEPTION’
- ALEN TRNSL EXCEPTION’
- ALE SEQNC EXCEPTION’
- ASTE VALIDITY EXCPTN’
- ASTE SEQNC EXCEPTION’
- EXTND AUTH EXCEPTION’
- STACK FULL EXCEPTION’
- STACK EMPTY EXCPTION’
- STACK SPEC EXCEPTION’
- STACK TYPE EXCEPTION’
- STACK OPER EXCEPTION’
- SPACE SWITCH EVENT ‘
- HFP SQ ROOT EXCPTION’
- PC TRANS SPEC EXCPTN’
- AFX TRNSL EXCEPTION’
- ASX TRNSL EXCEPTION’
- LX TRNSL EXCEPTION’
- EX-TRANSL EXCEPTION’
- PRIM AUTHORITY XCPTN’
- SCNDRY AUTH EXCPTION’
- ALET SPFC EXCEPTION’
- ALEN TRNSL EXCEPTION’
- ALE SEQNC EXCEPTION’
- ASTE VALIDITY EXCPTN’
- ASTE SEQNC EXCEPTION’
- EXTND AUTH EXCEPTION’
- STACK OPER EXCEPTION’
- ASCE TYPE EXCEPTION’
- RGN 1ST TRANS EXCPTN’
- RGN 2ND TRANS EXCPTN’
- RGN 3RD TRANS EXCPTN’
- MONITOR EVENT ‘

- CRYPTO OP EXCEPTION'

The following fields are on the next three lines

PASN= 4 hex characters identifying the Primary ASID value.
HASN= 4 hex characters identifying the Home ASID value.
PSW= The 16 byte content of the PSW recorded in the event.
LOC= The virtual storage location of the PSW. Will be one of the following:

- NUCLEUS
- PRIVATE
- EXT PRIV
- COMMON

KEY= The key of the PSW. The key the program was in when the event was recorded
STATE= Either PROB for problem state or SUP for supervisor state.
ASC= The Address Space Control mode from the PSW (PRIM, HOME, AR or SCND).
MODE= The amode bit from the PSW. 24-, 31- or 64-bit addressing.
INSTR ADDR= The instruction address from the PSW.
LM= The name of the Load Module that contains the PSW instruction address.
OFFSET= The offset into the Load Module load point for the instruction address.
WORK UNIT= The name of the Work Unit.
TOKEN= For tasks, the token value from the STOKEN field in the STCB control block. For SRBs, a unique number generated by z/XPF to identify the SRB

When the field reads "**EVENT=START SUBCH"**, You'll see these fields in the Event Record:

- **DEVICE=** A 4-character hex device address, from the start subchannel event record
- **BASE=** A 4-character hex device address. When PAV is active for the device, this will show the PAV base device address.
- **IOSASID=** The ASID value associated with the start sub-channel event. This will always match the target application home ASID (Address Space Identifier).
- **TCB=** The virtual storage address of the TCB associated with the event.

DRVR ID= This is the IO Supervisor code placed in the start sub-channel record. This will contain one of the following values:

- RESERVED FOR IOS
- MISC 24 BIT IOS
- EXCP PROCESSOR
- VSAM
- VTAM
- TCAM
- OLTEP
- PCI FETCH
- JES3

- MSC
- IECVIOPM PURGE
- VPSS
- CRYPTO ?????
- ASM
- MSG DSPL SERVICE
- ASGN/UNASGN SRVC
- DYNAMIC PATHING
- DAVV
- DEV CNTL SERVICE
- ASYC OPRTN MNGR
- DFSMS
- XCF CTC I/O DRVR
- IOS USE DRVR ID
- IOSVSLFD DRVR ID
- IOSVIOPA DRVR ID
- MISC 31 BIT IOS
- SVC33
- CLR DEVICE RCVRY
- SUBCHANNEL RCVRY
- SVC16 PURGE
- UNCONDTNL RESRVE
- MSNG INTRPT HNDL
- I/O PREV HANDLER
- RE-RESERVE SRVCE

When the event is an I/O interrupt, an additional line will be in the event that contains the channel and device status contained in the interrupt.

DEV STATUS = one or more of the following:

- UNIT EXCEPTN
- UNIT CHECK
- DEVICE END
- CHANNEL END
- BUSY
- CTL UNIT END
- STATUS MDFR
- ATTENTION

It is also possible to see a sub-channel status word contained in the I/O interrupt. In that case you will see “**SUBCH STATUS =**”. The value can be one of the following types:

- CHAINING CHECK
- NTRFC-CNTL CHCK
- CHNNL-CNTL CHCK

- CHNNL-DATA CHCK
- PROTCTN CHECK
- PROGRAM CHECK
- INCORRECT LNGTH
- PGM CTL INTRUPT

Chapter 8 - Using z/XPF-PC

z/XPF-PC extends z/XPF's reporting to the desktop environmnet. You can compress data capture reports and ship them down to the PC platform via FTP for processing on the decktop. z/XPF-PC allows you to use a GUI interface for summary reporting, with a number of advantages:

- The concept of "Time Segments" goes away. Each event is a discrete data point.
- You can see line graphs, bar graphs and pie charts for statistics in your reports
- You can easily drill down in the same way as you can with z/XPF's dynamic ISPF, but you get a good deal more granularity.

z/XPF-PC is tailor-made for people who look at many reports in a single day and need a way to quickly spot trends. z/XPF-PC is also good for people who are more comfortable with a graphical interface rather than tabular reports. It's fast and intuitive!

What are the downsides? Well, you need to compress your data capture reports on the mainframe side, and format them for FTP transmission to the desktop. Depending on the sized of your report, this can take significant time and resources to accomplish. Also, the desktop machine needs to have plenty of memory and a decent processor.

[Example: Our test code for this release contained about six million events. Running on a 3.06GHz processor with 8GB of RAM, the decompression took just over two minutes. The parsing operation took a LOT longer. This not unusual. It is the consequence of asking z/XPF-PC to process large amounts of data.]

8-1 System Requirements for z/XPF-PC

z/XPF-PC needs at least 2GB of RAM in order to process the large amounts of data that form its reports. More memory is desirable.

z/XPF-PC will run on Windows XP Service Pack 3 and above, with Microsoft .NET Framework 4.0 and any other components that .NET Framework requires.

8-2 Installing z/XPF-PC

z/XPF-PC is delivered with the rest of the z/XPF product in a file called "<zXPF-VnRnMnn>.zxpf". It has been compressed using WINZIP. In order to get past various e-mail filters, we alter the file-extension from ".zip" to ".zxpf". When the file is downloaded and decompressed you will be left with "zXPFPC.exe", which you merely execute in the desktop environment

When you execute zXPFPC.exe and agree to the terms of the EULA, z/XPF-PC will present a list of components to install. Make your life easy; accept the defaults!

Next, z/XPF-PC prompts you for a destination folder and a Start Menu folder. Accept the defaults or put z/XPF-PC wherever you wish. Now, click on the “install” button.

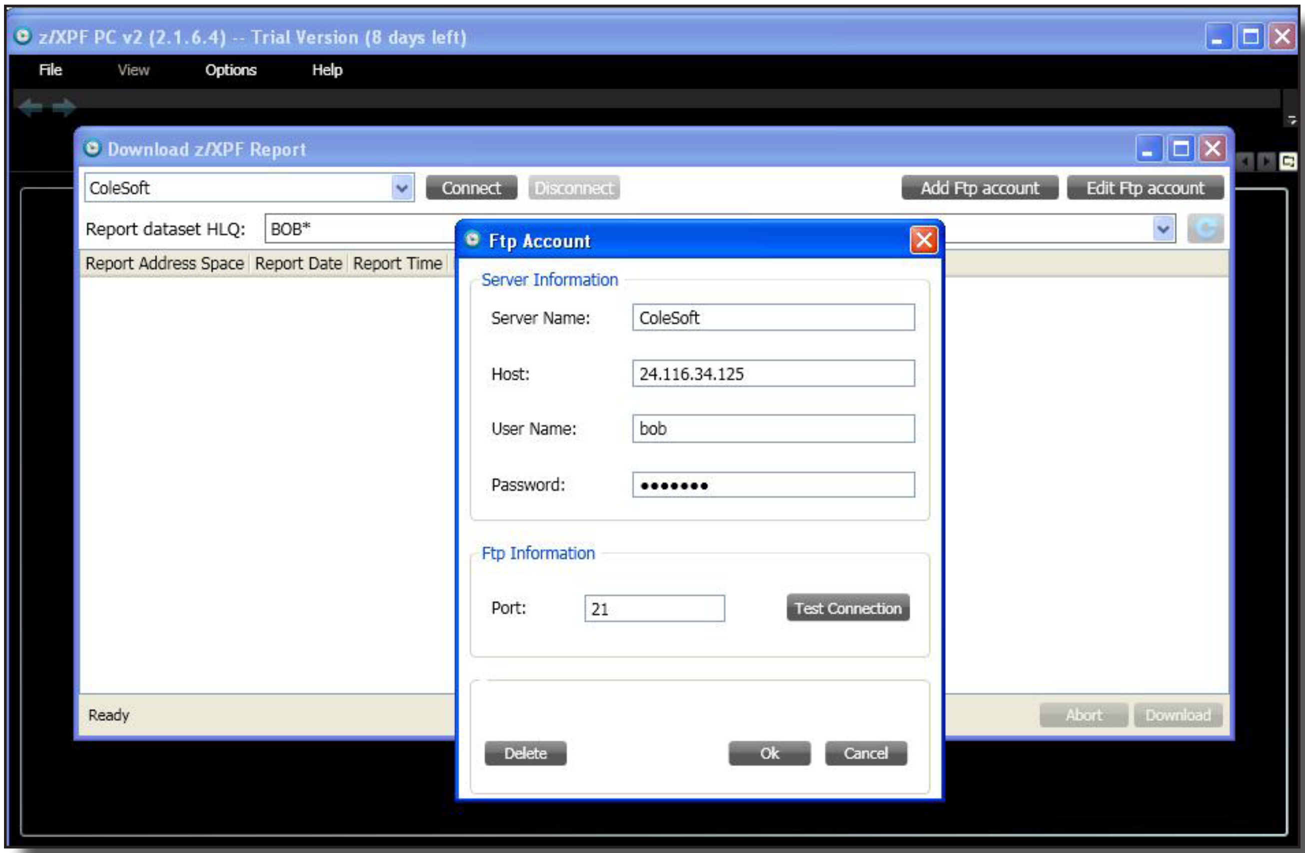
Upon installation, z/XPF-PC will check for MicroSoft .NET Framework release 4.0. You will see the executable file ”zXPF PC v2.exe” in the folder that you choose to install z/XPF-PC into. You may create a shortcut to this .exe file, and place the shortcut wherever you wish.

When you get the “Installation Complete” dialogue box, z/XPF is ready to use. Close this box and start up the product using the shortcut icon.

[The first time you create a new report with z/XPF-PC, you will be asked , “Please choose a default folder where z/XPF-PC will store report databases. Create a new folder, or select a folder on your hard drive.]

8-3 Creating an FTP Connection to Your Mainframe

After you install z/XPF-PC you will need to establish an FTP connection to your mainframe. Click File/New Report/From Mainframe, and z/XPF-PC will take you to the Download z/XPF Report file. Click on “Add FTP account” and you will see the panel below in Figure 8-3-1:



ColeSoft

Report dataset HLQ: BOB*

Server Name: ColeSoft

Host: 24.116.34.125

User Name: bob

Password: •••••••

Port: 21

Figure 8-3-1.

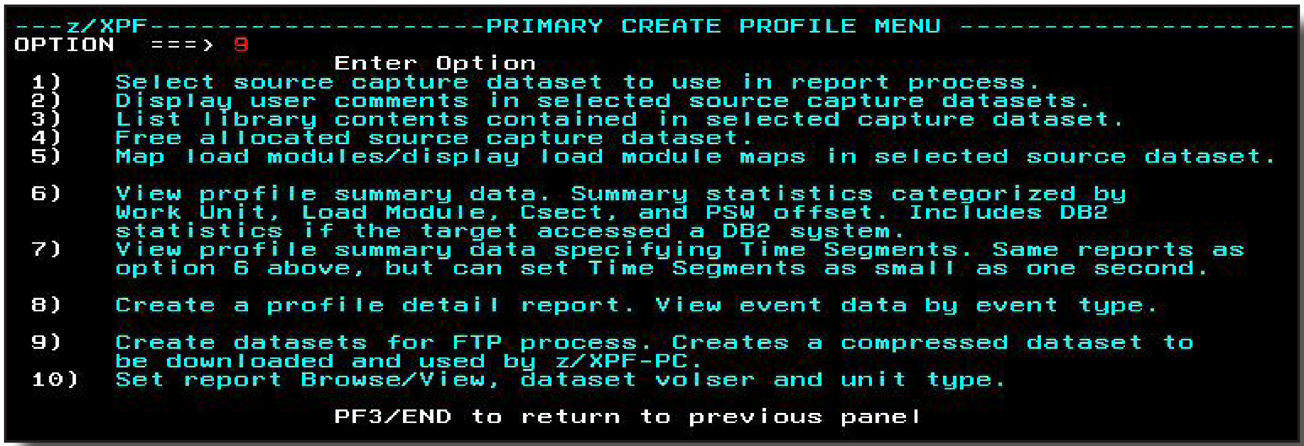
Fill in your own installation-specific values and test the connection. When the connection works, then you can move on to getting a report from the mainframe.

8-4 Prepare a z/XPF Data Set for Download to the Desktop

On the mainframe side within z/XPF, you need to allocate a report and compress it for FTP download to the desktop.

[Please note that in a future iteration of z/XPF-PC we'll remove the need to compress the dataset]

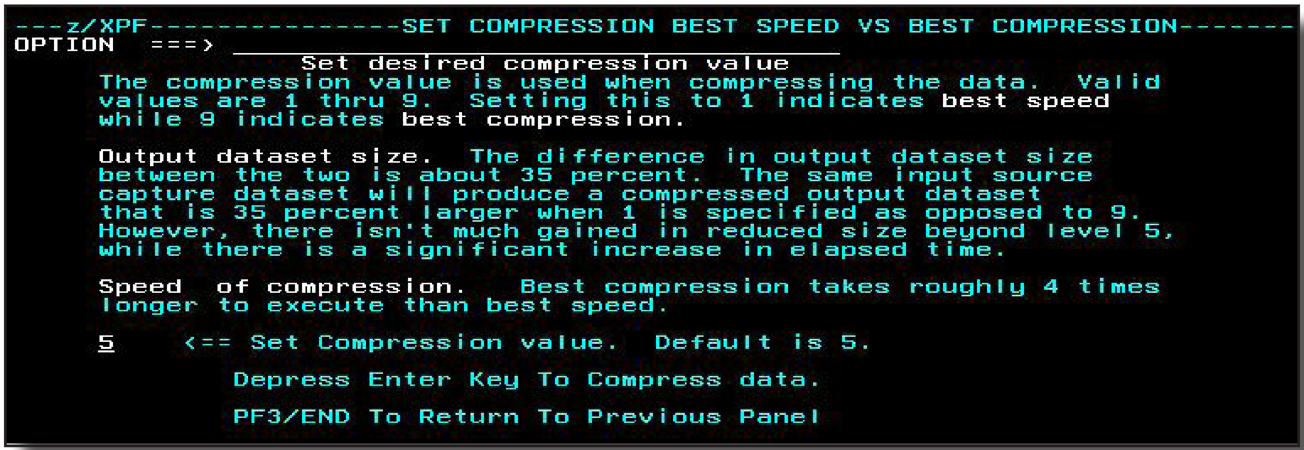
In the panel below, I'm getting ready to execute Option 9, to compress my already-allocated report: See Figure 8-4-1 below:



```
---z/XPF--------------------PRIMARY CREATE PROFILE MENU --------------------
OPTION   ===> 9
                    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. Same reports as
      option 6 above, but can set Time Segments 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 8-4-1.

Now I am given a chance to alter the compression value. I usually leave the compression value set to "5". See Figure 8-4-2 below:

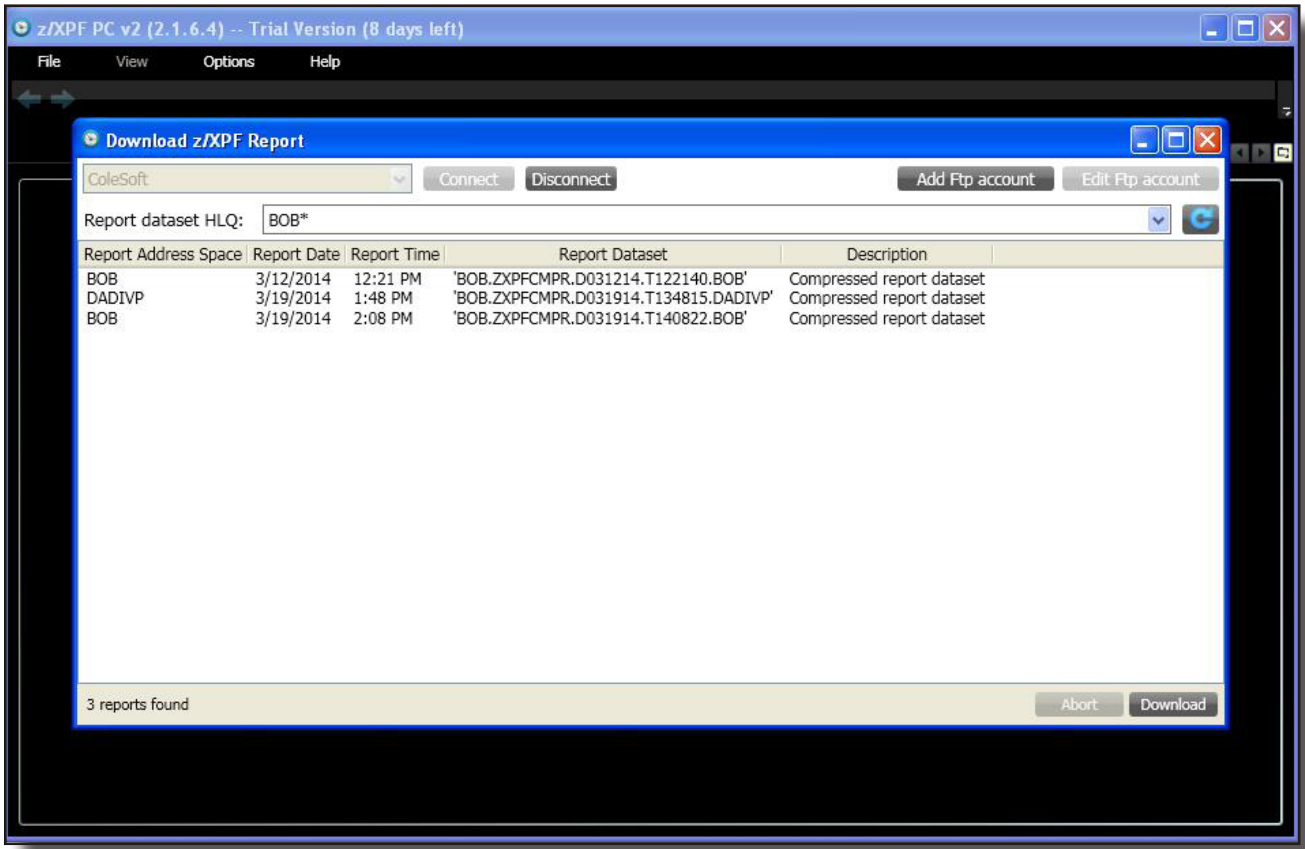


```
---z/XPF---------------SET COMPRESSION BEST SPEED VS BEST COMPRESSION-------
OPTION   ===> ____________________________________
                  Set desired compression value
     The compression value is used when compressing the data.  Valid
     values are 1 thru 9.  Setting this to 1 indicates best speed
     while 9 indicates best compression.

     Output dataset size.  The difference in output dataset size
     between the two is about 35 percent.  The same input source
     capture dataset will produce a compressed output dataset
     that is 35 percent larger when 1 is specified as opposed to 9.
     However, there isn't much gained in reduced size beyond level 5,
     while there is a significant increase in elapsed time.

     Speed  of compression.   Best compression takes roughly 4 times
     longer to execute than best speed.

     5    <== Set Compression value.  Default is 5.

             Depress Enter Key To Compress data.

             PF3/END To Return To Previous Panel
```

Figure 8-4-2.

If I press Enter on this panel, z/XPF performs the compression step. This grinds away, and you may as well find somethine else to do for a while.

Once this step is complete, I'll go to the desktop environment and download the file. To do that I go to z/XPF-PC and issue File/New Report/From Mainframe. Press the "Connect" button and z/XPF-PC presents the panel below, in Figure 8-4-3:



Report dataset HLQ: BOB*

Report Address Space	Report Date	Report Time	Report Dataset	Description
BOB	3/12/2014	12:21 PM	'BOB.ZXPFCMPR.D031214.T122140.BOB'	Compressed report dataset
DADIVP	3/19/2014	1:48 PM	'BOB.ZXPFCMPR.D031914.T134815.DADIVP'	Compressed report dataset
BOB	3/19/2014	2:08 PM	'BOB.ZXPFCMPR.D031914.T140822.BOB'	Compressed report dataset

3 reports found

Figure 8-4-3.

Select a file by high-lighting it and press the "Download" button. z/XPF-PC will prompt you to navigate to a folder that will store the report. Select that folder, the file will download and z/XPF-PC will parse the report. This can take a few minutes. You can watch the progress bar or find something else to do.

When the report has been prepared, you will see a panel that looks like this, in Figure 8-4-4 below:

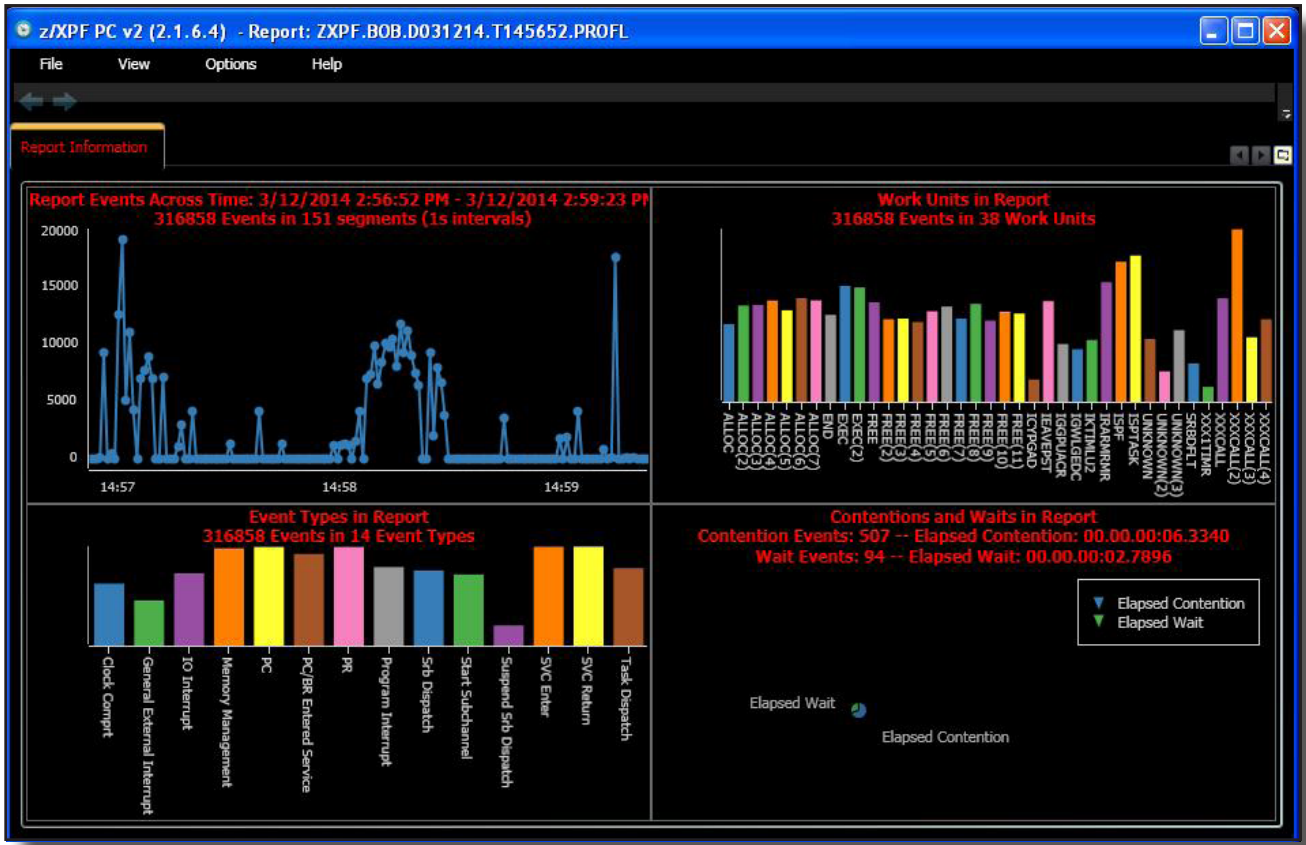
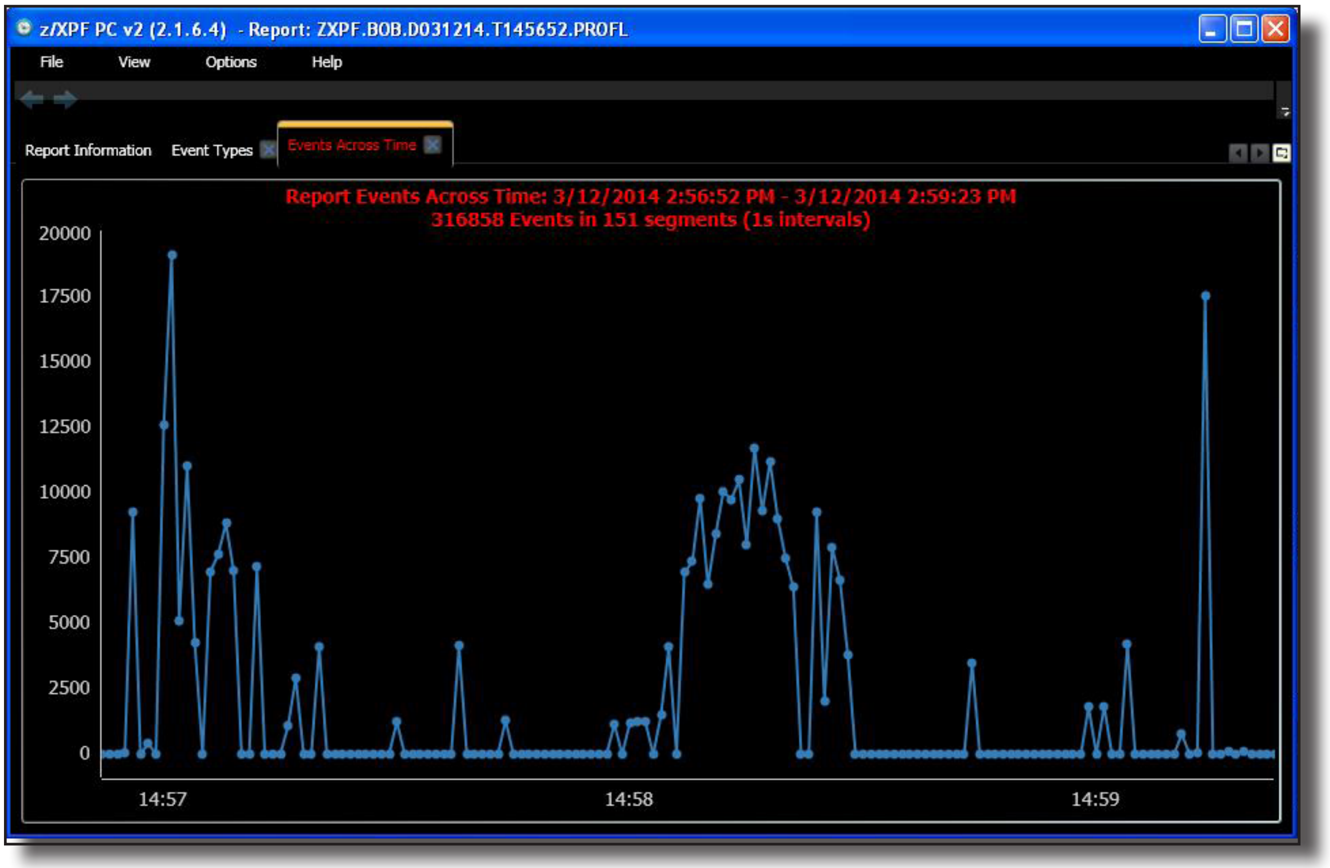


Figure 8-4-4.

You can select from one of the four major categories by clicking on it.

The first report that users tend to gravitate towards is Report Events Across Time. To zero in on that report, click View/Events Across Time. You'll isolate that report as below, in Figure 8-4-5:



Report Events Across Time: 3/12/2014 2:56:52 PM - 3/12/2014 2:59:23 PM
316858 Events in 151 segments (1s intervals)

Figure 8-4-5.

In any of z/XPF-PC's reports you can left-click and drag to zoom or pan. You can right click on any data point to reveal information on that instant in the report. It all works as you would expect a PC application to. I'm getting ahead of myself. Let's explore the interface.

8-5 Viewing Reports with the z/XPF-PC GUI interface

After parsing, or upon opening a report, z/XPF-PC opens its general interface. You will see this in the top of your panel, as in Figure 8-5-1 below:

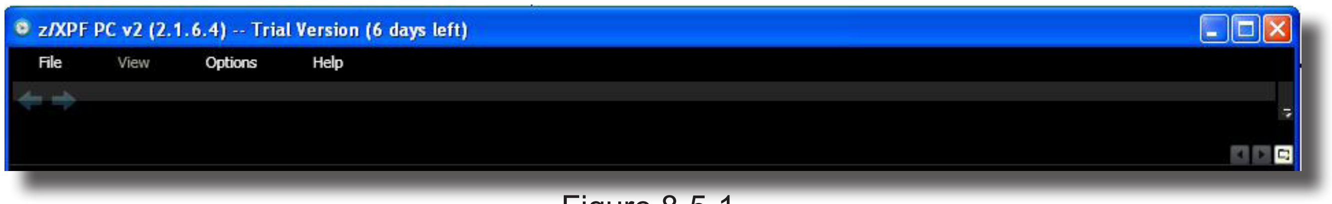


Figure 8-5-1.

Under the File menu, you can choose from New Report, Open Report, Close Report, Recent Reports and Exit.

Under the View Menu, you may use View to select from the folllowing reports:

- Events across time;
- Work Units;
- Event Breakdown;
- Contention/Wait Time;
- SVC Breakdown;
- PC Breakdown (Program Call);
- PR Breakdown (Program Return);
- PT Breakdown (Program Transfer).

Until you open a report, “View” is grayed out. Other choices also remain grayed out until they become relevant.

Under the Options menu, you can choosed to establish or change the default file directory, where
z/XPF-PC will store its report databases after parsing.

Under the Help menu, you can open the z/XPF-PC User Guide, view the z/XPF-PC Log file (useful for debugging), and register your copy of z/XPF-PC.

8-6 The z/XPF-PC display

The left hand side of the display shows the number of events on the vertical Y axis, and Data Points (or time) on the horizontal X access. While z/XPF on the mainframe offers you the ability to split your report in up to ten “Time Segments”, z/XPF-PC defaults to one second intervals OR a maximum of 600 “data points” along the horizontal X axis.

Please note that the meaning of the X and Y axes are fluid, and shift with the context of the report view. An example of a the Work Units report appears below, in Figure 8-5-2:

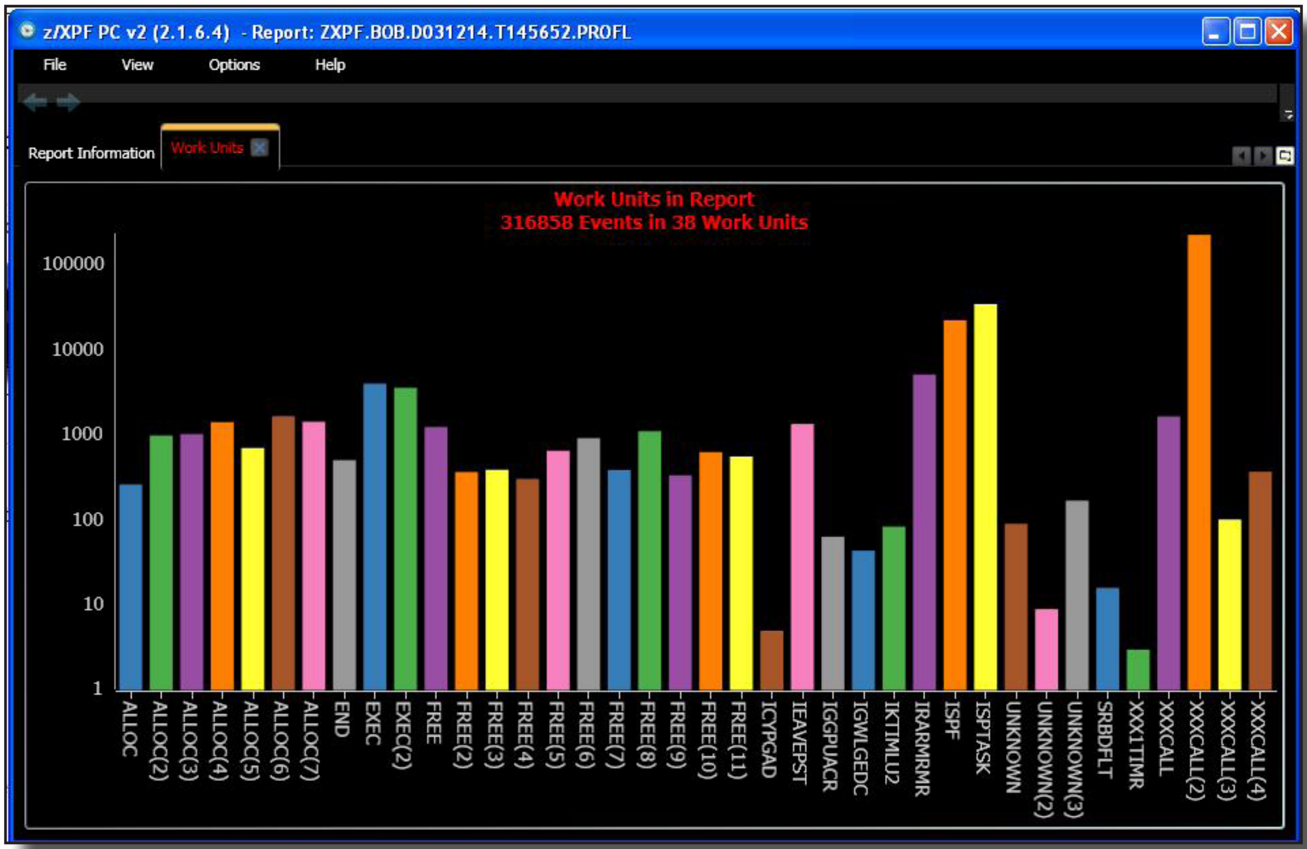


Figure 8-5-2.

8-6 A Short Review of z/XPF’s Reporting Structure

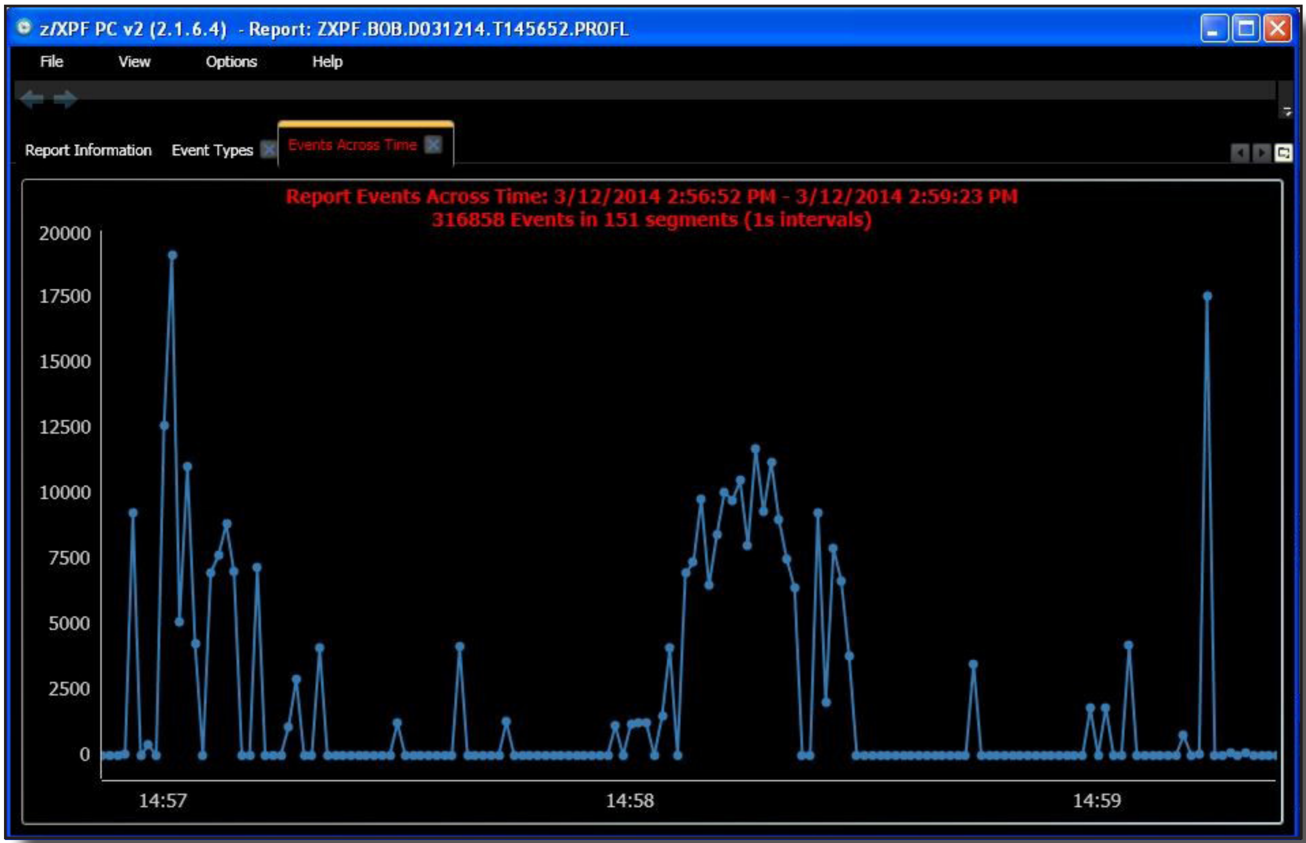
z/XPF’s reporting structure follows a distinct hierarchy:

- At the top level are time segments - the entire span of the job, divided evenly into blocks of time. On the mainframe, z/XPF allows as many as one-second time segments. z/XPF-PC, however, gives you much finer granularity.
- At the next level, z/XPF shows “Work Units”, which can either be individual tasks or individual SRBs.
- At the next level, z/XPF shows modules.
- Within modules are csects.
- Within csects, z/XPFdisplays individual PSW offsets.

8-7 Drilling down into "Events Across Time"

Looking at the Events Across Time display, one can quickly see where the most events occurred. In this section we'll drill down through the report until we can go no further, to find out WHY so many events occur at this time in our data capture.

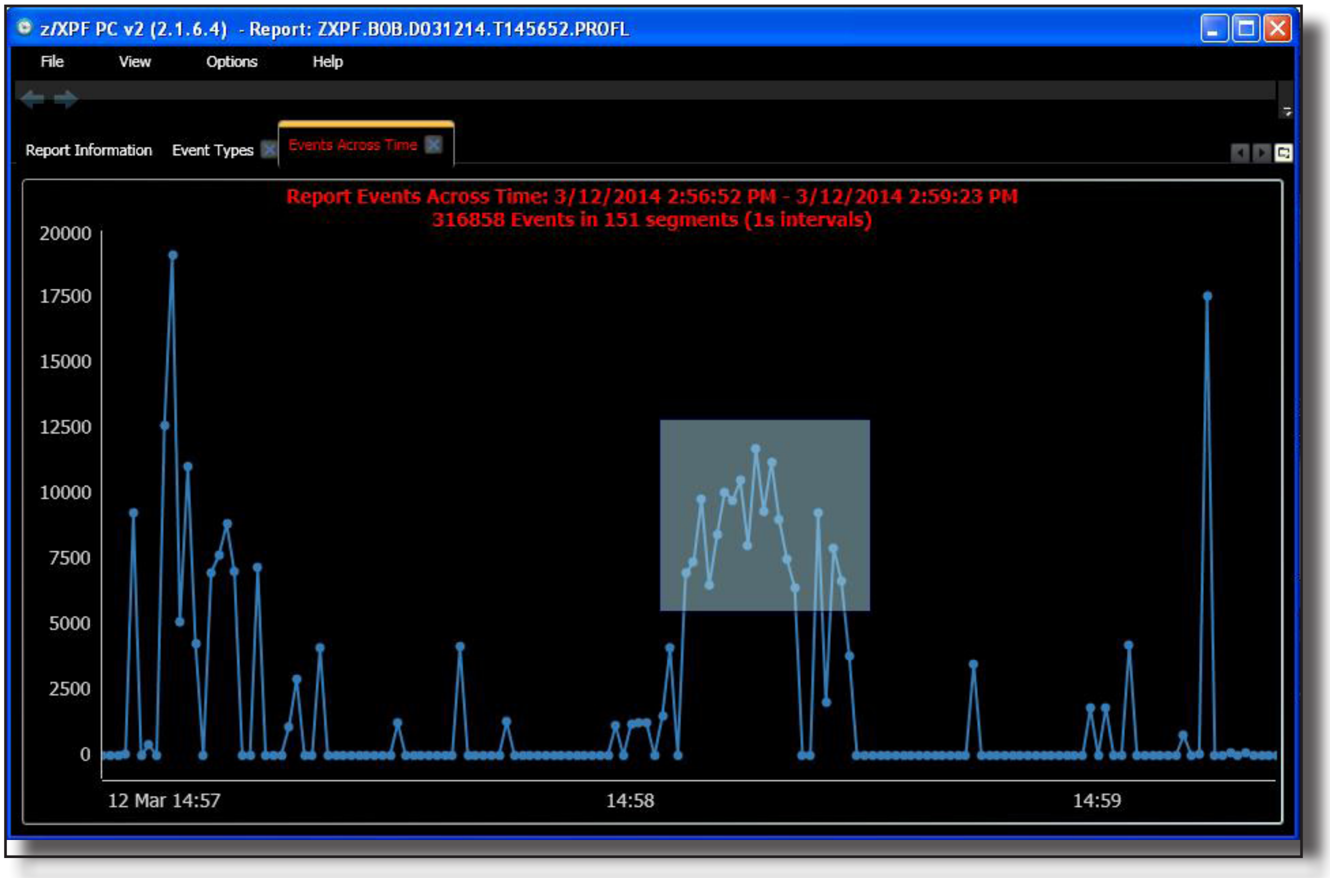
First, we select the "Events Across Time" tab in the report. See Figure 8-7-1 below: :



Report Events Across Time: 3/12/2014 2:56:52 PM - 3/12/2014 2:59:23 PM
316858 Events in 151 segments (1s intervals)

Figure 8-7-1.

In this graph, you would be most interested in the time segments with the greatest number of events. That's where most resources may be consumed. So, we left-click and drag to magnify the area as below, in Figure 8-7-2:



Report Events Across Time: 3/12/2014 2:56:52 PM - 3/12/2014 2:59:23 PM
316858 Events in 151 segments (1s intervals)

Figure 8-7-2.

After the left-click and drag operation, z/XPF-PC's display will zoom into the selected area. Now you can see each "time segment" in the graph more distinctly. Please see Figure 8-7-3, below:

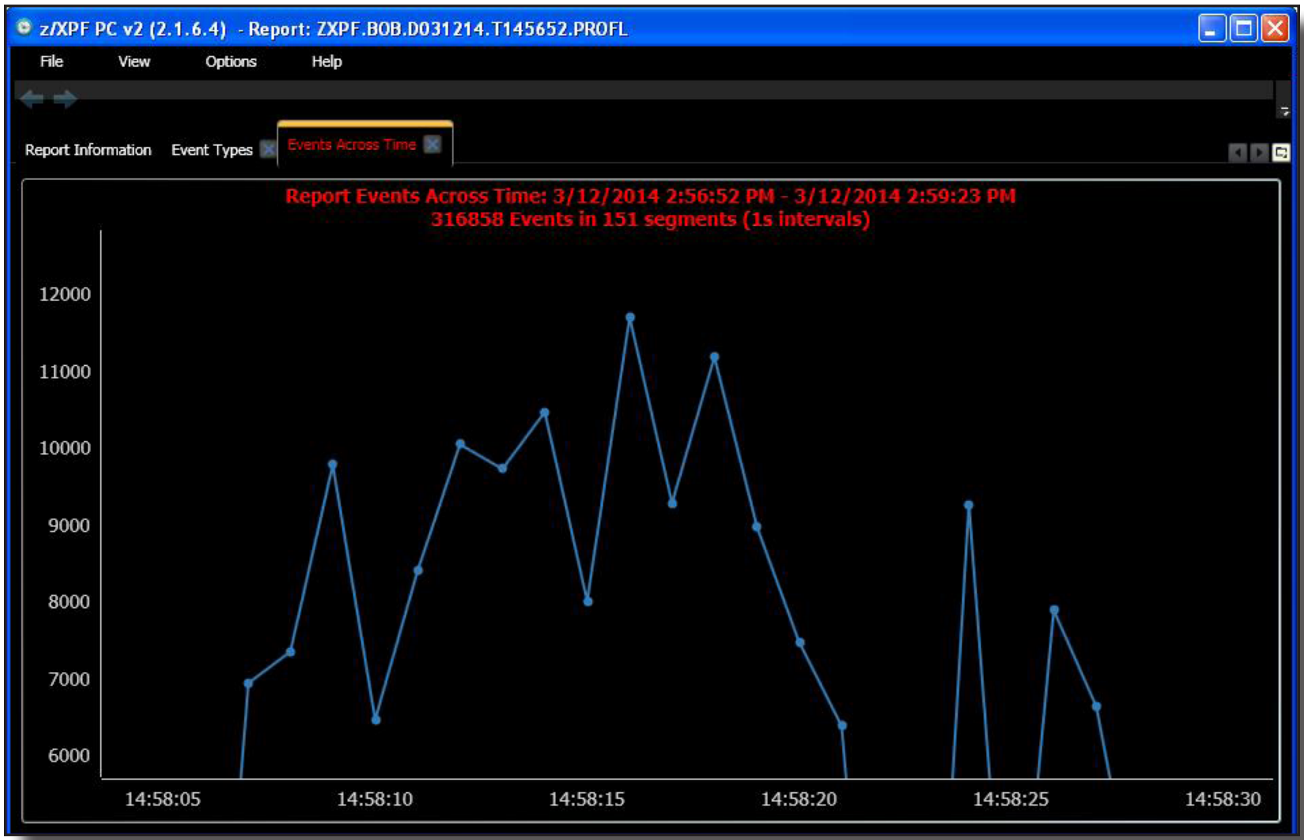

z/XPF PC v2 (2.1.6.4) - Report: ZXPF.BOB.D031214.T145652.PROFL
File
View
Options
Help
Report Information
Event Types
Events Across Time
Report Events Across Time: 3/12/2014 2:56:52 PM - 3/12/2014 2:59:23 PM
316858 Events in 151 segments (1s intervals)
12000
11000
10000
9000
8000
7000
6000
14:58:05
14:58:10
14:58:15
14:58:20
14:58:25
14:58:30

Figure 8-7-3.

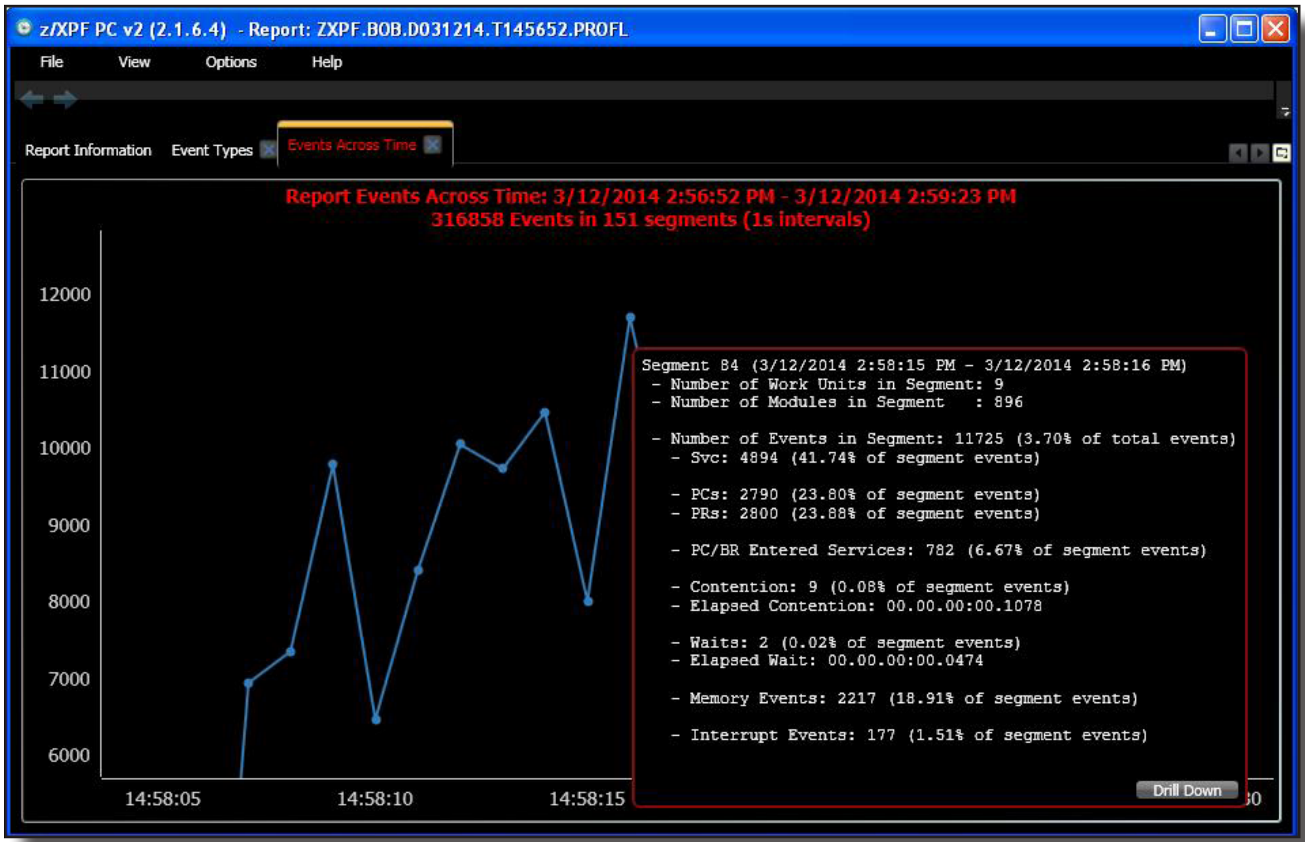
[To UN-ZOOM, please RIGHT-CLICK to reveal the functions available to you, which includes "Undo Zoom/Pan". Zooming can also be accomplished by using the mouse wheel-button to zoom in and out.]

You may also left-click, and drag to magnify your selection further.

8-8 Obtaining Detailed information, and drilling down

Whenever you wish, you may left-click on a point in the graph. You will receive a detailed report on that particular data point. You will also have access to the "Drill Down" function. This will take you to the next level down in the report. In the context of this display ("Events Across Time"), this will drill down from the Time Segment level to the Work Unit Level.

Obviously, one of the points in the graph shows a higher event count for that segment. We'll select that point on the graph below in Figure 8-8-1: :


z/XPF PC v2 (2.1.6.4) - Report: ZXPF.BOB.D031214.T145652.PROFL
File
View
Options
Help
Report Information
Event Types
Events Across Time
Report Events Across Time: 3/12/2014 2:56:52 PM - 3/12/2014 2:59:23 PM
316858 Events in 151 segments (1s intervals)
12000
11000
10000
9000
8000
7000
6000
14:58:05
14:58:10
14:58:15
Drill Down

```
Segment 84 (3/12/2014 2:58:15 PM - 3/12/2014 2:58:16 PM)
 - Number of Work Units in Segment: 9
 - Number of Modules in Segment   : 896

 - Number of Events in Segment: 11725 (3.70% of total events)
   - Svc: 4894 (41.74% of segment events)

   - PCs: 2790 (23.80% of segment events)
   - PRs: 2800 (23.88% of segment events)

   - PC/BR Entered Services: 782 (6.67% of segment events)

   - Contention: 9 (0.08% of segment events)
   - Elapsed Contention: 00.00.00:00.1078

   - Waits: 2 (0.02% of segment events)
   - Elapsed Wait: 00.00.00:00.0474

   - Memory Events: 2217 (18.91% of segment events)

   - Interrupt Events: 177 (1.51% of segment events)
```

Figure 8-8-1.

Next, we click on the "Drill Down" button. Now we see the two work units running at the time of Segment #18 in the report, "APIVPZRO" and "APIVPONE", though APIVPZRO's event count was negligible. See Figure 8-8-2, below:

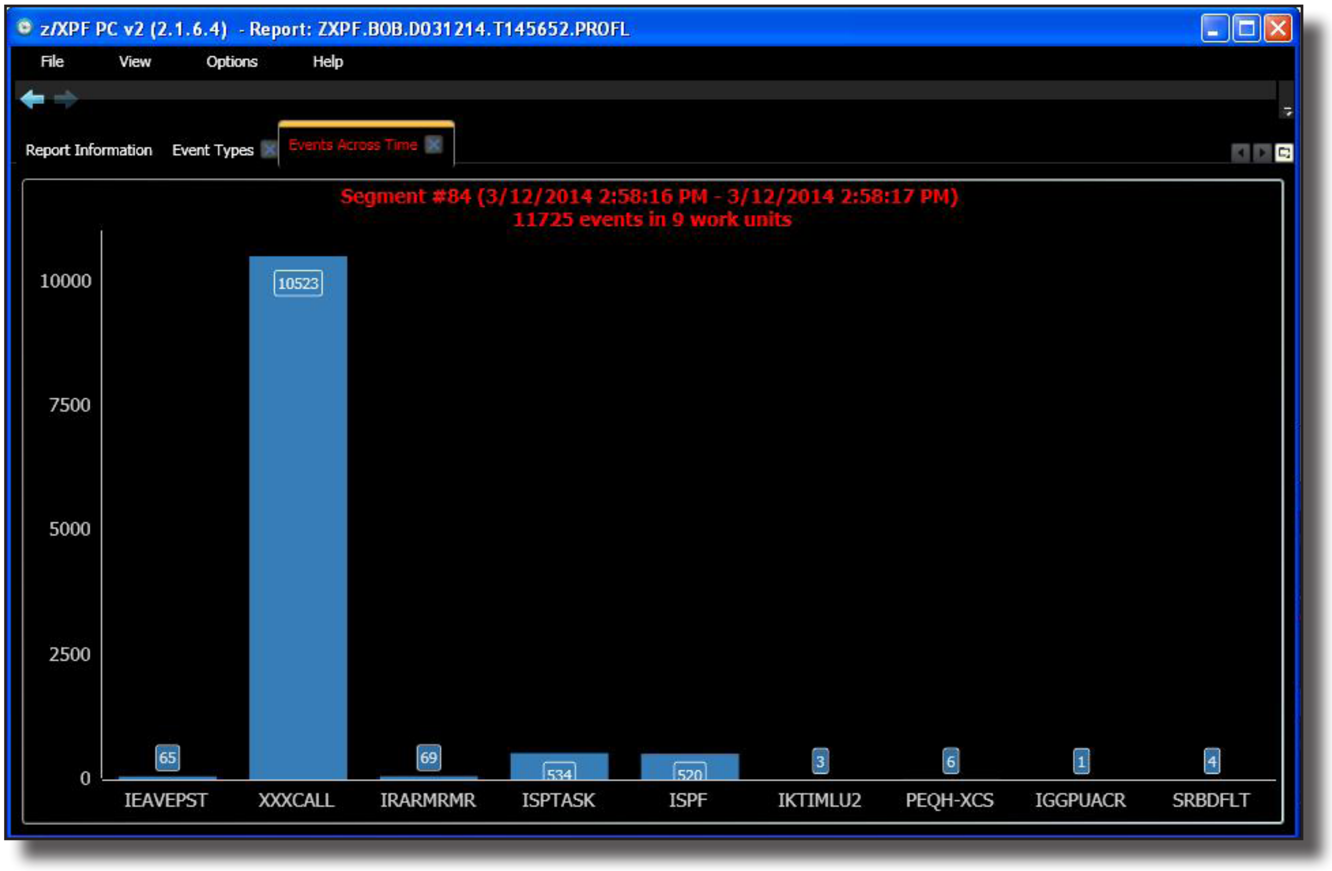
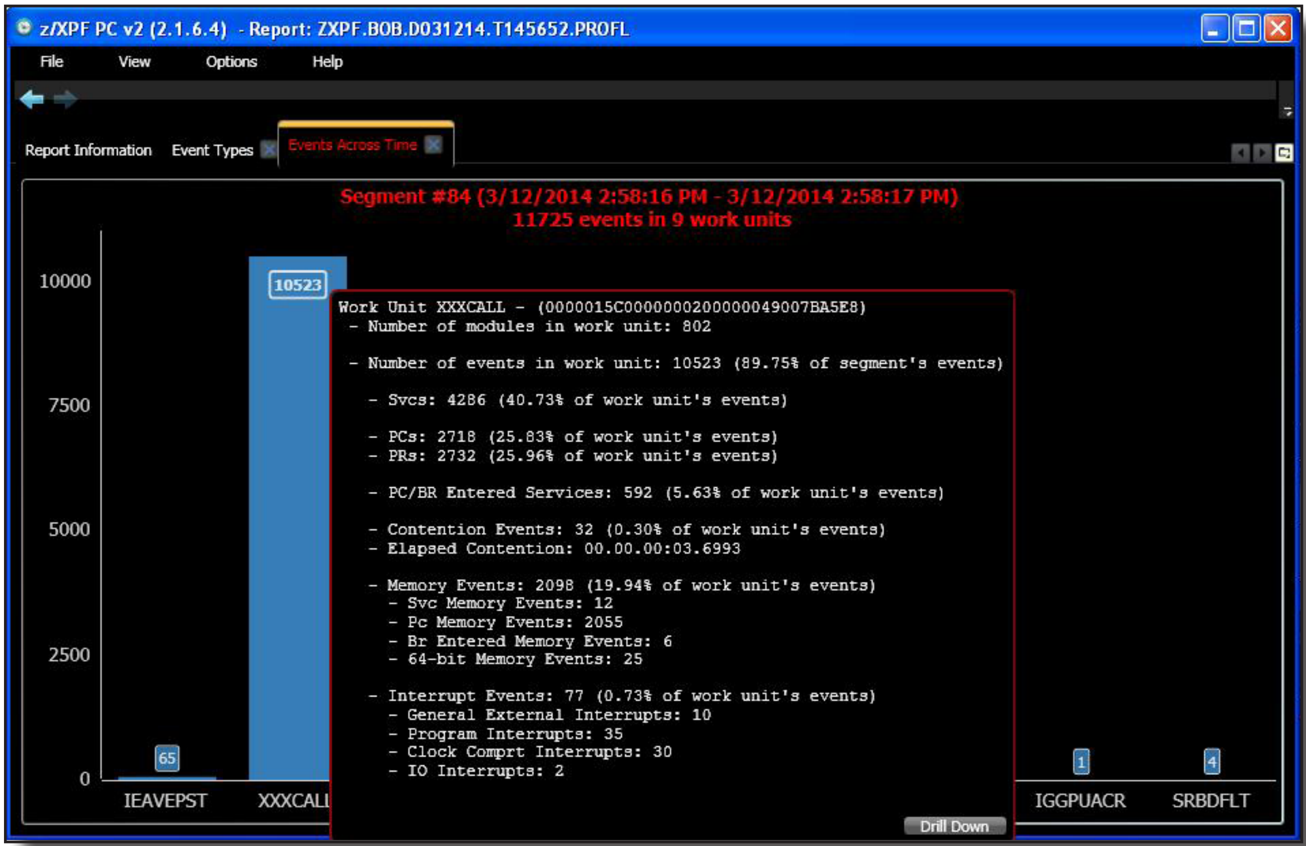


Figure 8-8-2.

Selecting XXXCALL gives us more information, and the opportunity to “drill down” again: Clearly, most of the events occurred in Work Unit APIVPONE. We’ll left-click to see Figure 8-8-3 below:


z/XPF PC v2 (2.1.6.4) - Report: ZXPF.BOB.D031214.T145652.PROFL
File
View
Options
Help
Report Information
Event Types
Events Across Time
Segment #84 (3/12/2014 2:58:16 PM - 3/12/2014 2:58:17 PM)
11725 events in 9 work units
10000
7500
5000
2500
0
10523
65
1
4
IEAVEPST
XXXCALL
IGGPUACR
SRBDFLT
Drill Down

```
Work Unit XXXCALL - (0000015C0000000200000049007BA5E8)
 - Number of modules in work unit: 802

 - Number of events in work unit: 10523 (89.75% of segment's events)

   - Svcs: 4286 (40.73% of work unit's events)

   - PCs: 2718 (25.83% of work unit's events)
   - PRs: 2732 (25.96% of work unit's events)

   - PC/BR Entered Services: 592 (5.63% of work unit's events)

   - Contention Events: 32 (0.30% of work unit's events)
   - Elapsed Contention: 00.00.00:03.6993

   - Memory Events: 2098 (19.94% of work unit's events)
     - Svc Memory Events: 12
     - Pc Memory Events: 2055
     - Br Entered Memory Events: 6
     - 64-bit Memory Events: 25

   - Interrupt Events: 77 (0.73% of work unit's events)
     - General External Interrupts: 10
     - Program Interrupts: 35
     - Clock Comprt Interrupts: 30
     - IO Interrupts: 2
```

Figure 8-8-3.

Let's drill again. Now we can see the modules running under Work Unit XXXCALL below, in Figure 8-8-4:

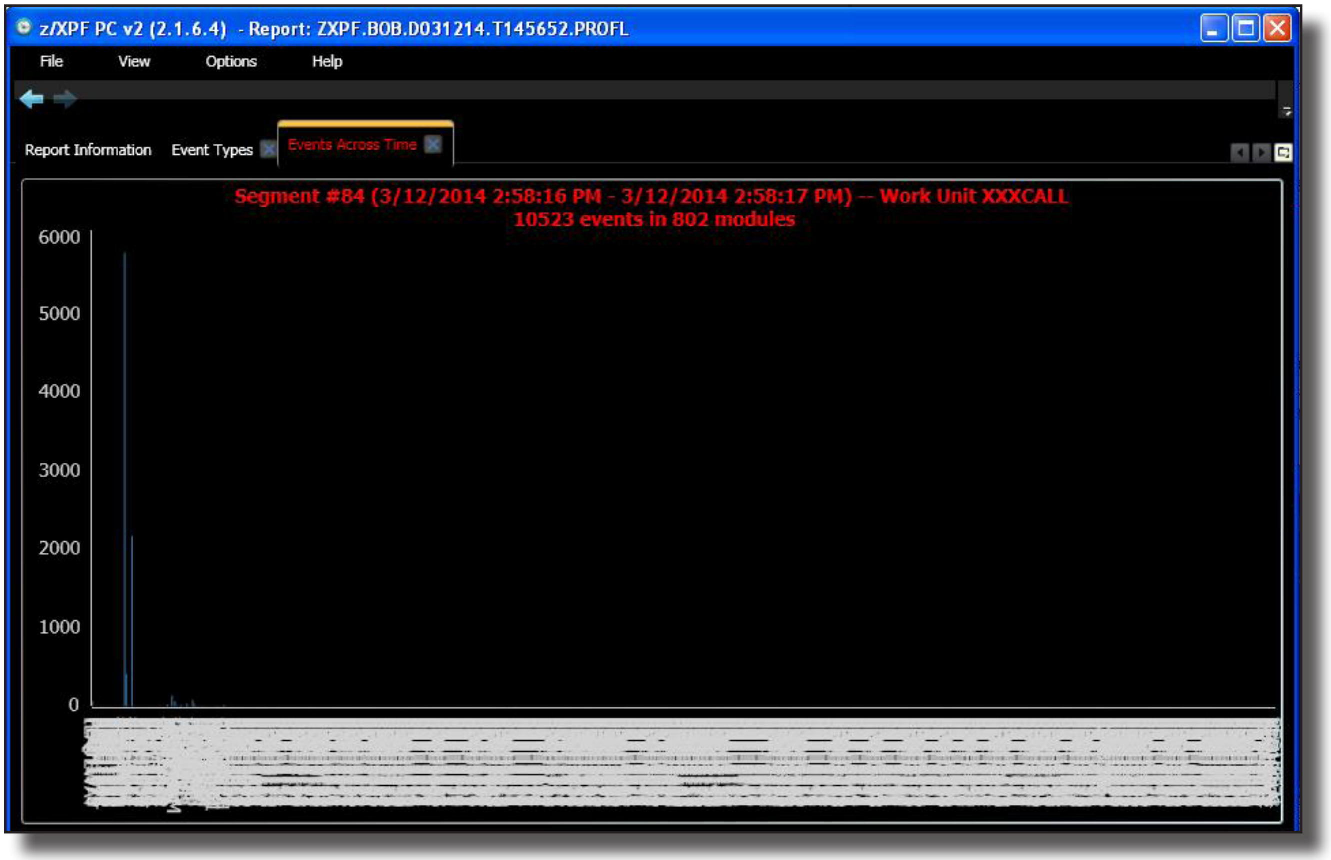

z/XPF PC v2 (2.1.6.4) - Report: ZXPF.BOB.D031214.T145652.PROFL
File
View
Options
Help
Report Information
Event Types
Events Across Time
Segment #84 (3/12/2014 2:58:16 PM - 3/12/2014 2:58:17 PM) -- Work Unit XXXCALL
10523 events in 802 modules
6000
5000
4000
3000
2000
1000
0

Figure 8-8-4.

Or, can we? z/XPF-PC is dutifully showing you ALL the 10,523 events across 802 modules within XXXCALL. The result is VERY cluttered! Let's clean it up.

Clearly, the left side of the screen has more activity (see the tiny blue "spike" in the graph?). If we left click and drag, z/XPF-PC will give us a more focused, and cleaner display. The intermediate step is to click and drag as in Figure 8-8-5:

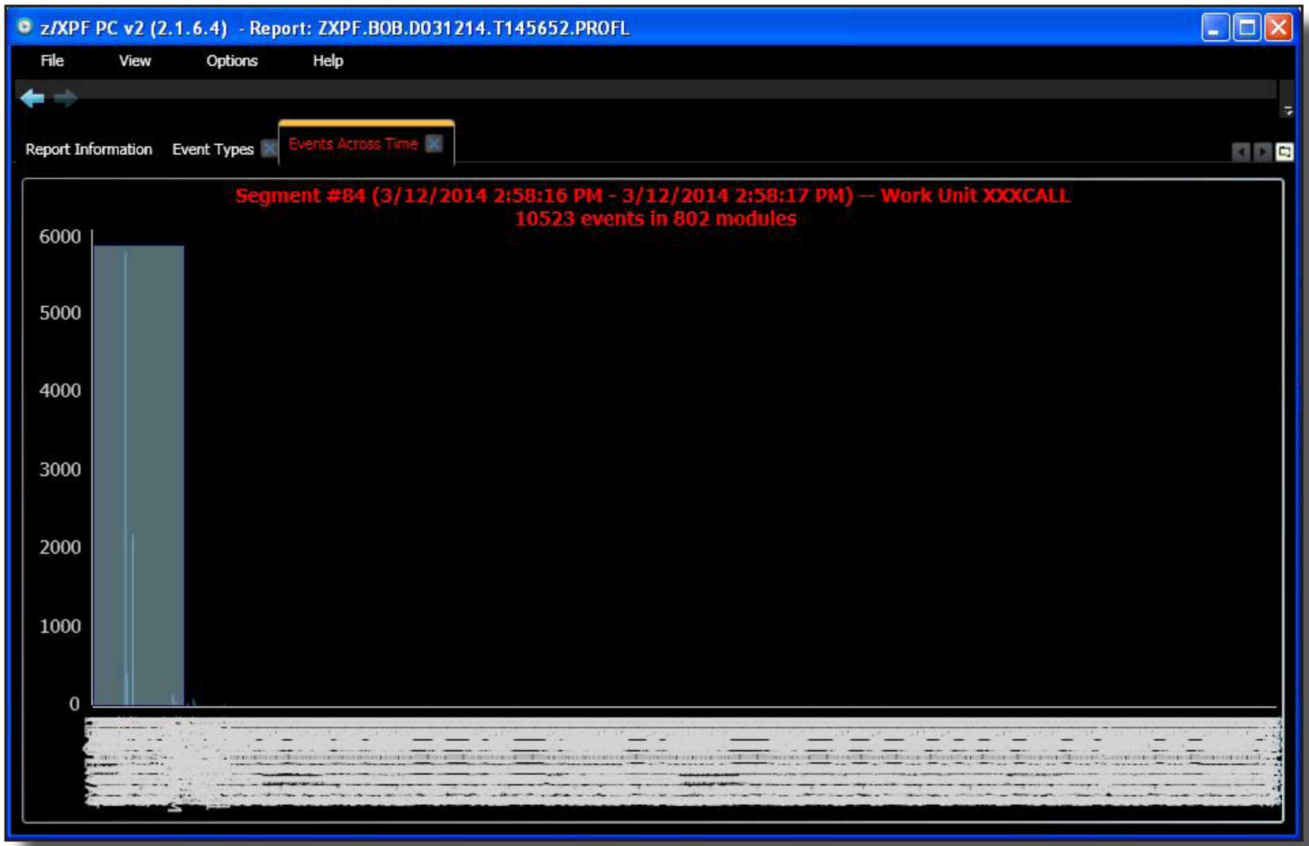

z/XPF PC v2 (2.1.6.4) - Report: ZXPF.BOB.D031214.T145652.PROFL
File
View
Options
Help
Report Information
Event Types
Events Across Time
Segment #84 (3/12/2014 2:58:16 PM - 3/12/2014 2:58:17 PM) -- Work Unit XXXCALL
10523 events in 802 modules
6000
5000
4000
3000
2000
1000
0

Figure 8-8-5.

Now the display is a good deal more intelligible. See Figure 8-8-6:

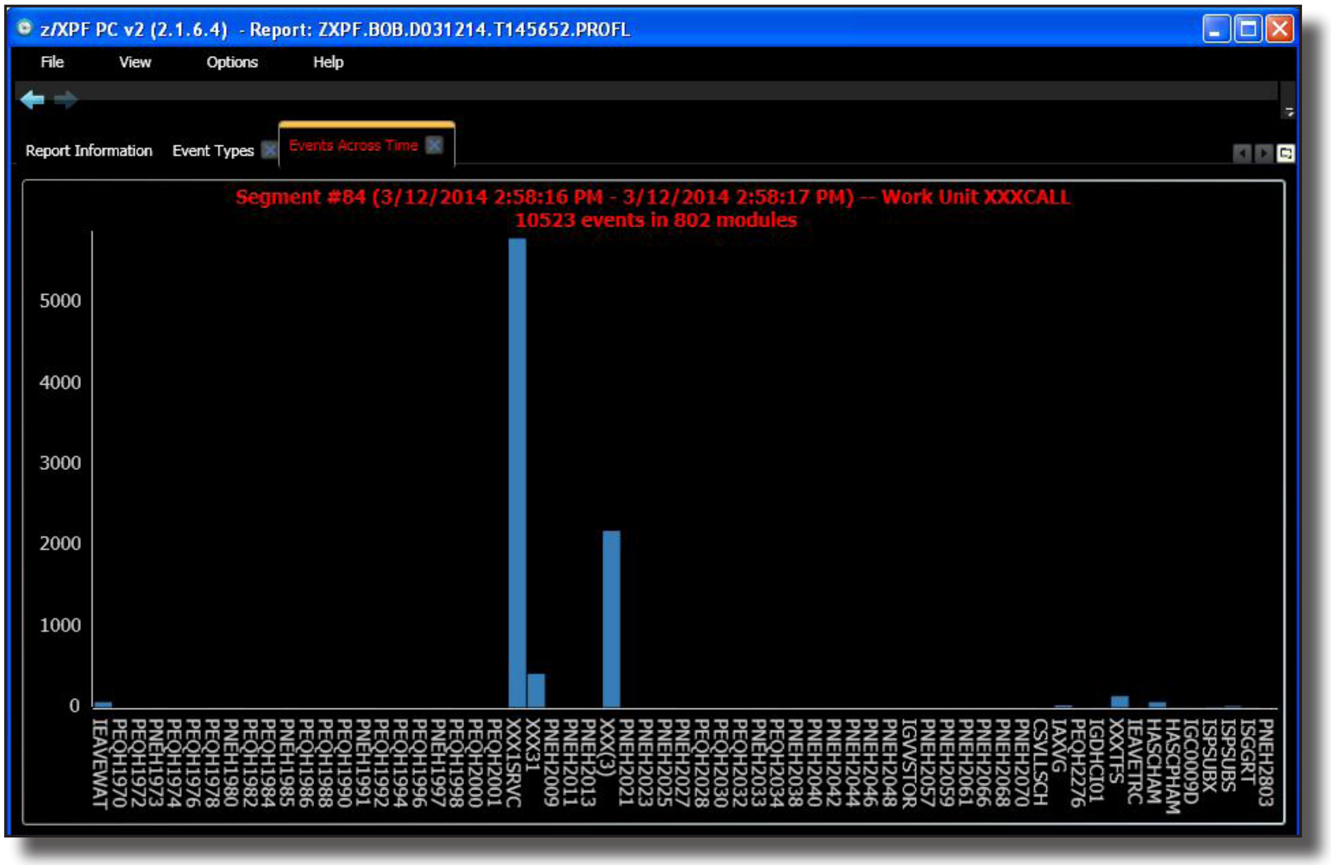


Figure 8-8-6.

Now, we see that the module XXX1SRVC shows the greatest number of events for the Work Unit XXXCALL within Time Segment #84. If we drill here, then z/XPF will drill again, and show us the PSW offset we're interested in. See Figure 8-8-7 below:,

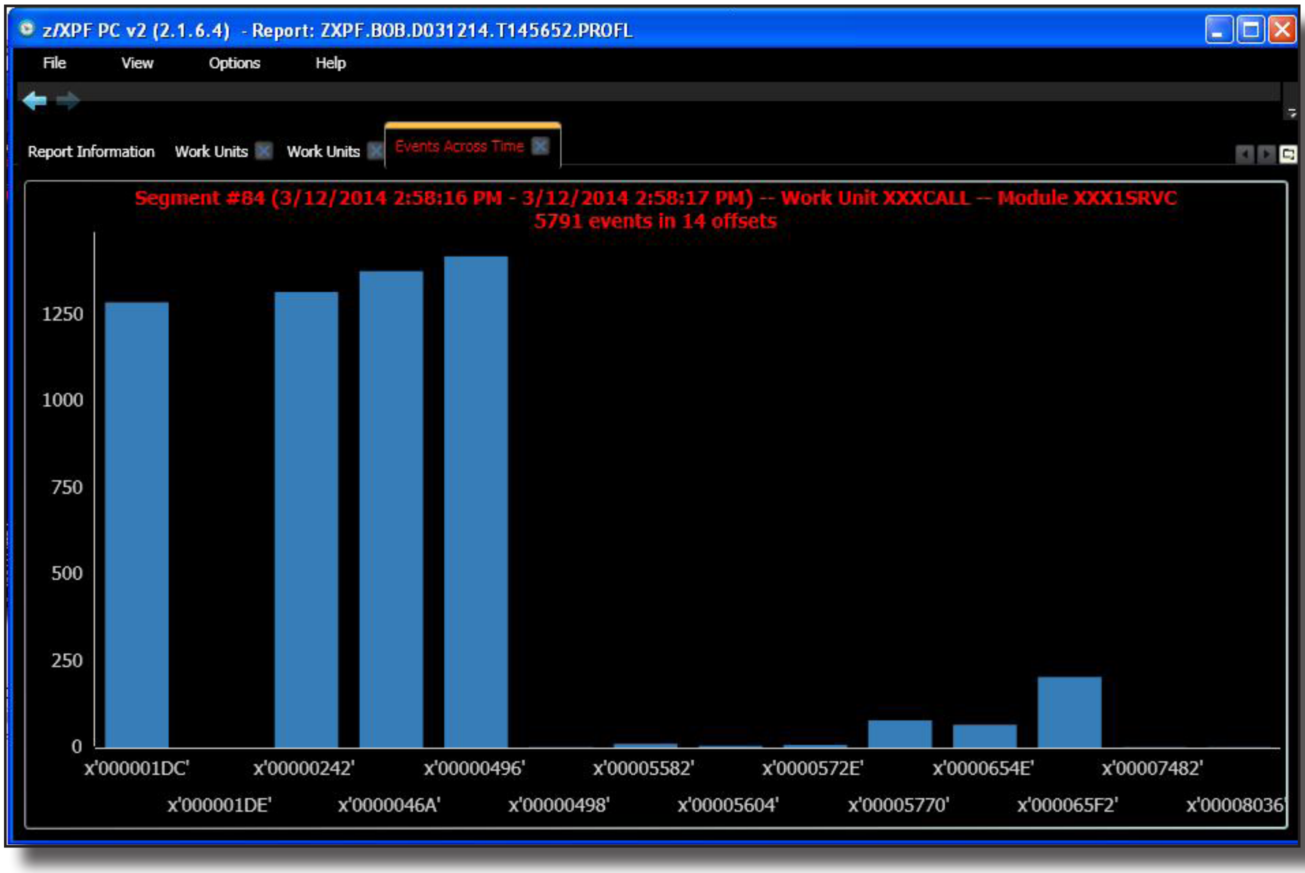
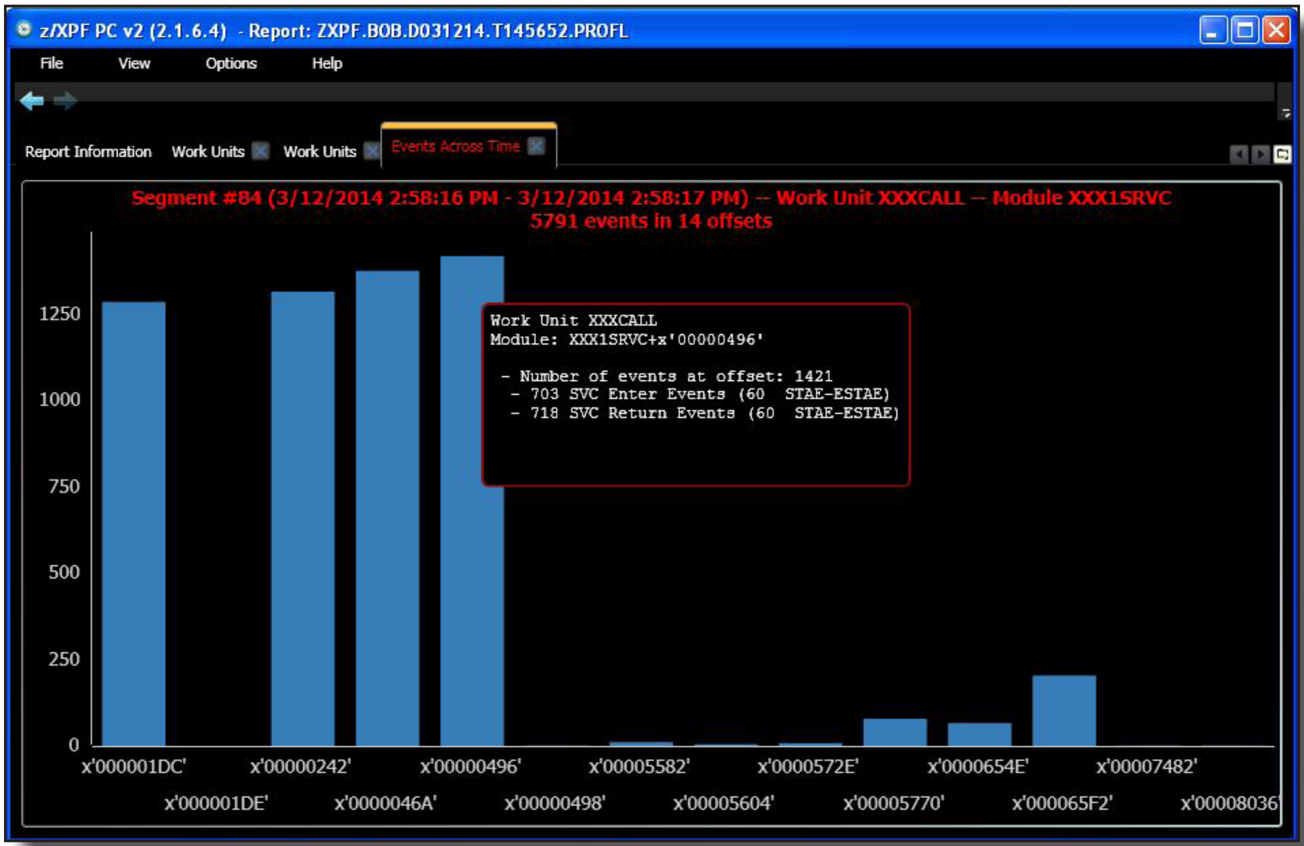


Figure 8-8-7.

Now, we HAVE skipped a step. We went directly from the module level display to the PSW offset, SKIPPING THE CSECT level. That is because in this particular case, z/XPF could not find a Binder Map for the module, so it cannot determine csect boundaries within the XXX1SRVC module. So, it appears that the offset X'496" into module XXX1SRVC is our "heaviest hitter" within Time Segment #84.

We cannot drill from here, but we can left-click and get a final level of information in this case. Please see Figure 8-8-8 below:



```
Work Unit XXXCALL
Module: XXX1SRVC+x'00000496'

 - Number of events at offset: 1421
  - 703 SVC Enter Events (60  STAE-ESTAE)
  - 718 SVC Return Events (60  STAE-ESTAE)
```

Figure 8-8-8.

That is as "deep" as z/XPF-PC can report. We have progressed step-wise from the Time Segment (in this case the entirety of the report), to the Work Unit, to the module to the the PSW offset. z/XPF-PC has displayed comprehensive information at each level, and you will have gained a good deal of information that may have been useful.

At this point you can use the left arrow (in the upper left corner, under "File") to back out successively until you reach another interesting screen or until you get to the initial display.

8-9 The Work Units Report

If you select the second tab in the initial display, you'll see something like this, in Figure 8-9-1 below:

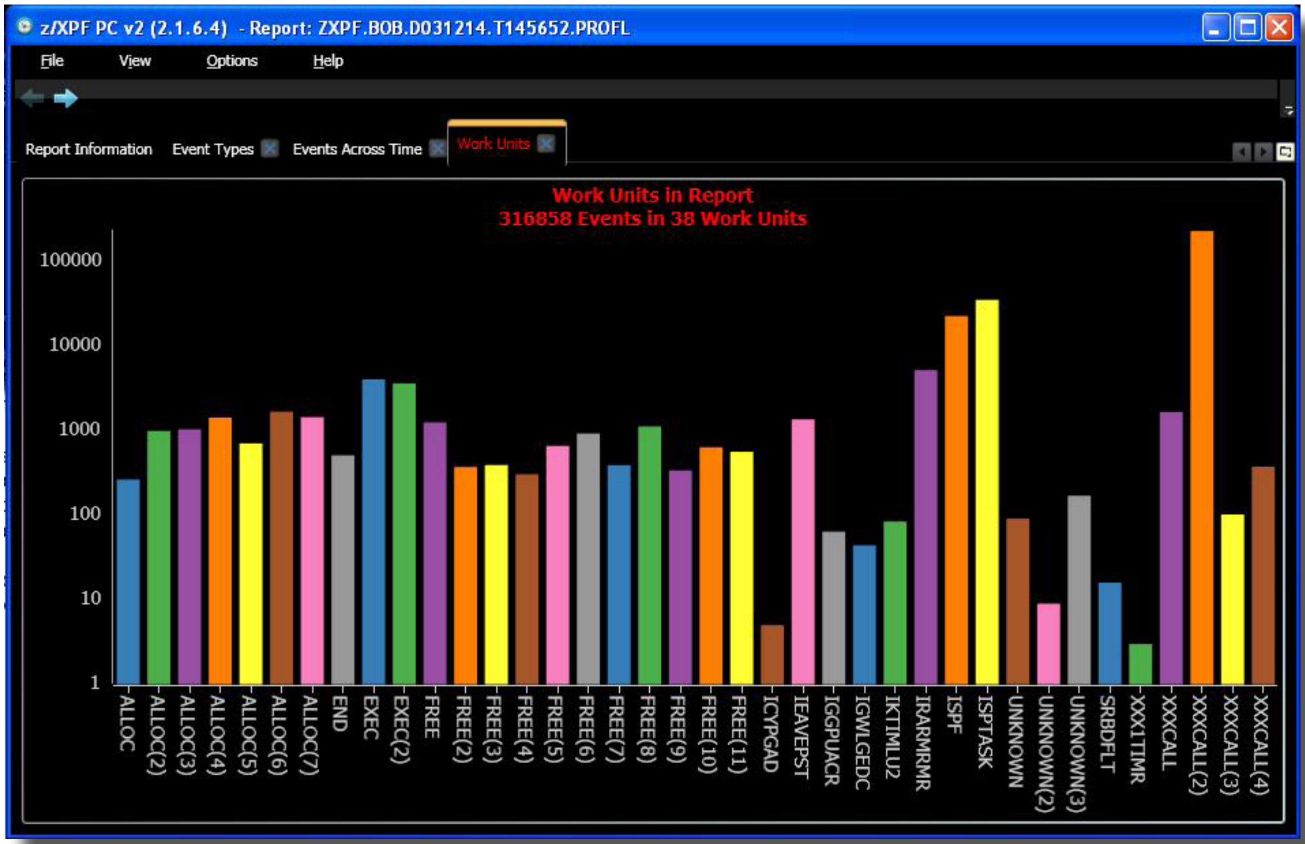


Figure 8-9-1.

Here are all the Work Units that were detected during the data capture. You can left-click/drill down into each of these Work Units. Some might be drawn to the "heaviest hitter", which is XXXCALL(2).

[Be advised: If you zoom into a very busy report it will appear as if "nothing is happening" for a while. The opposite is true! z/XPF-PC is driving your desktop machine very hard to perform your request and you may have to wait for the report to be displayed.]

8-10 The Event Breakdown Report

Selecting View/Event Breakdown yields a report that looks like Figure 8-10-1 below:

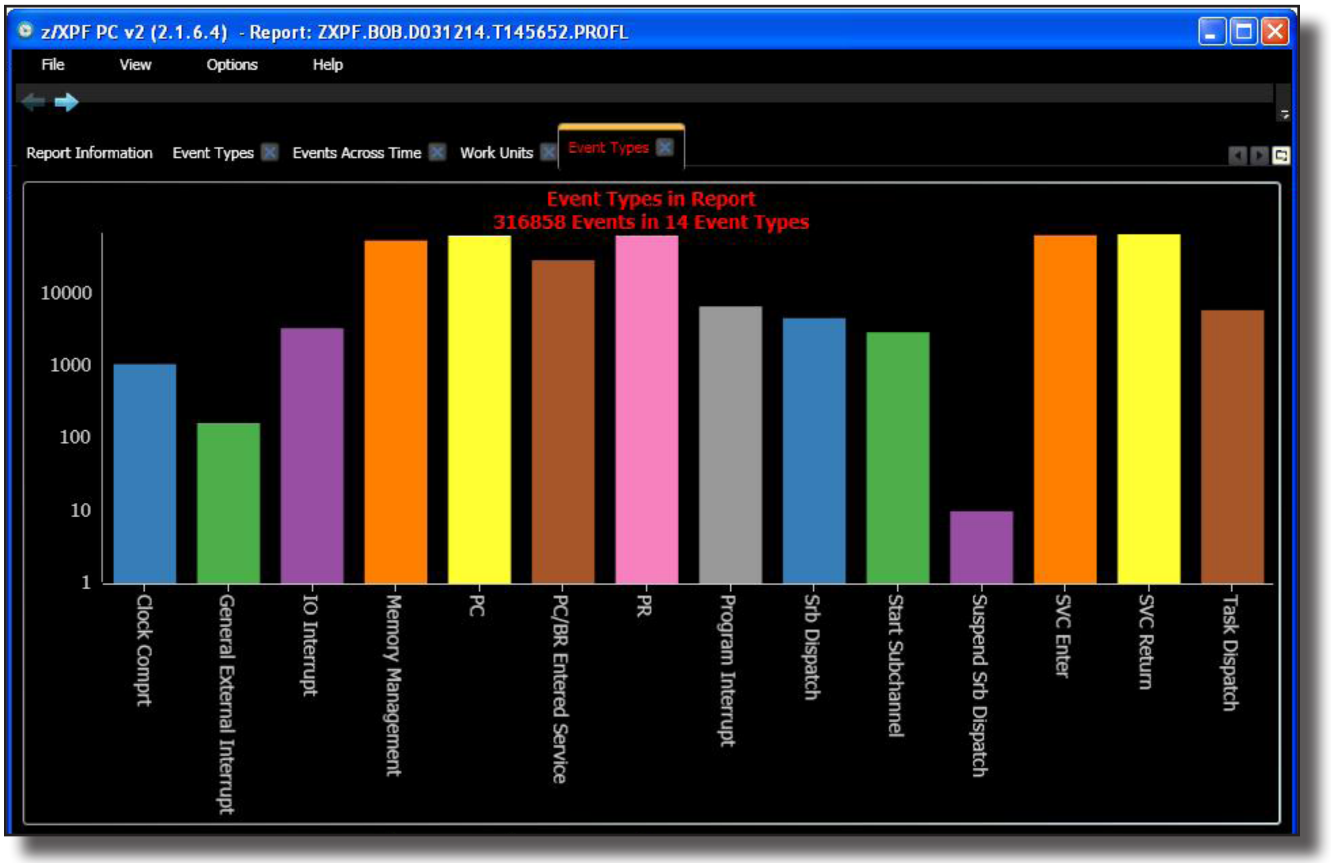
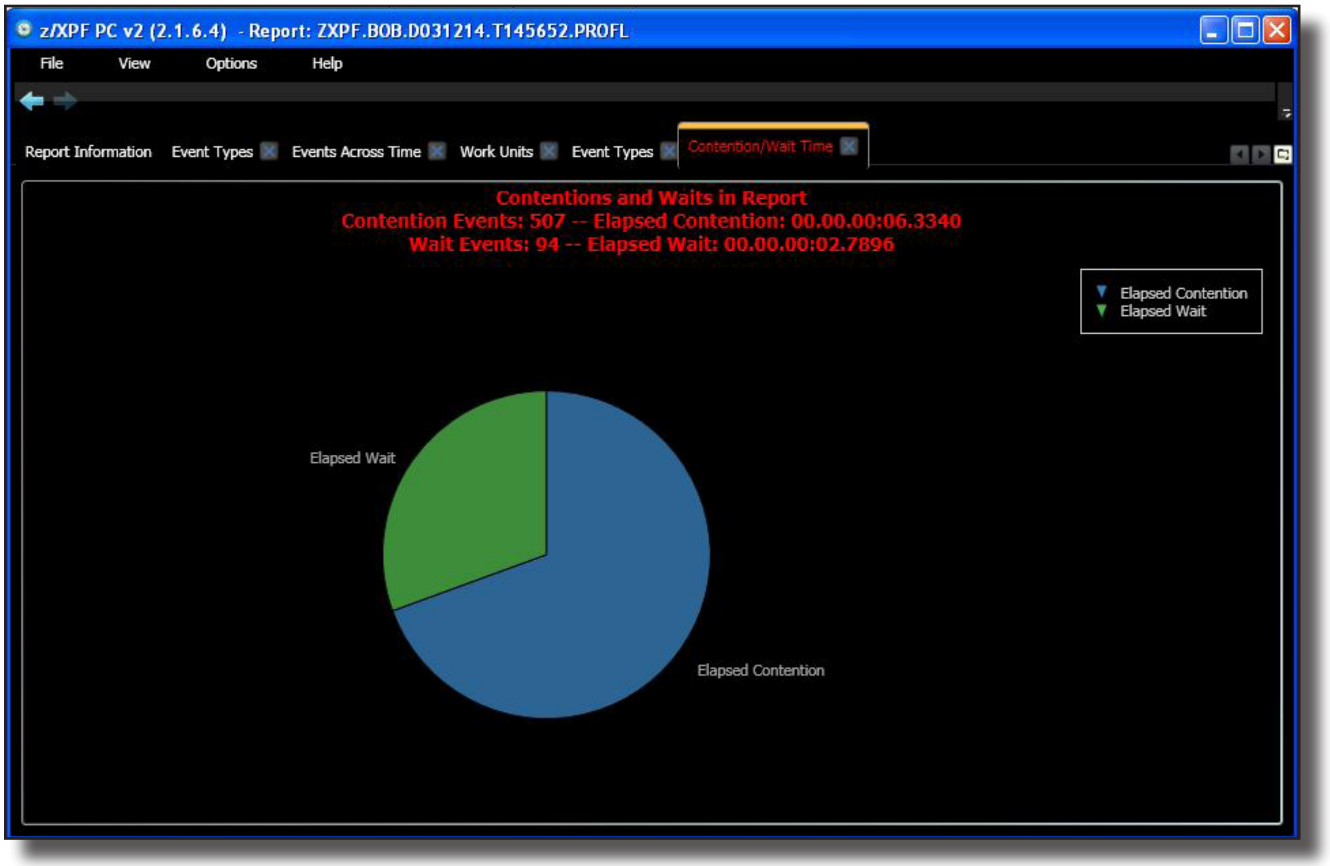


Figure 8-10-1.

This report delivers all the different kinds of events in the report categorized by type. Again, a left-click and drill down operation will reveal sub-categories.

8-11 Contention/Wait Time

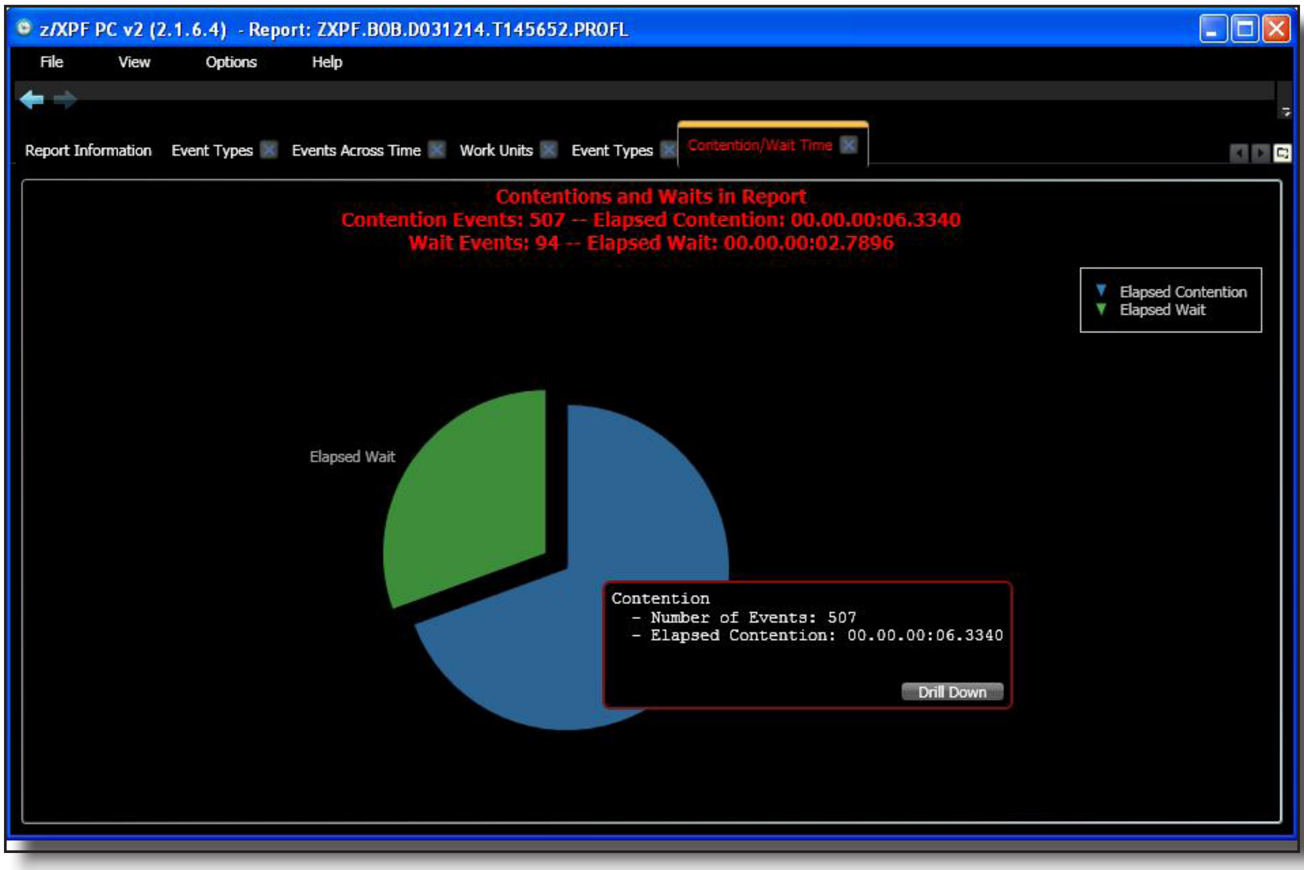
Selecting “View/Contention/Wait Time” in the menu bar at the top of the display,will show you the report below in Figure 8-11-1:



Contentions and Waits in Report
Contention Events: 507 -- Elapsed Contention: 00.00.00:06.3340
Wait Events: 94 -- Elapsed Wait: 00.00.00:02.7896

Figure 8-11-1.

If I click on one slice of the pie, it "floats" away from the whole, and gives me summary information and the opportunity to drill down. See Figure 8-11-2 below:



Contentions and Waits in Report
Contention Events: 507 -- Elapsed Contention: 00.00.00:06.3340
Wait Events: 94 -- Elapsed Wait: 00.00.00:02.7896

```
Contention
  - Number of Events: 507
  - Elapsed Contention: 00.00.00:06.3340
```

Figure 8-11-2.

This is cool enough that I should probably do a few more drills to show you z/XPF-PC's extreme level of "wonderfulness". See what happens when I drill into the large, blue piece of the "pie" in Figure 8-11-3:

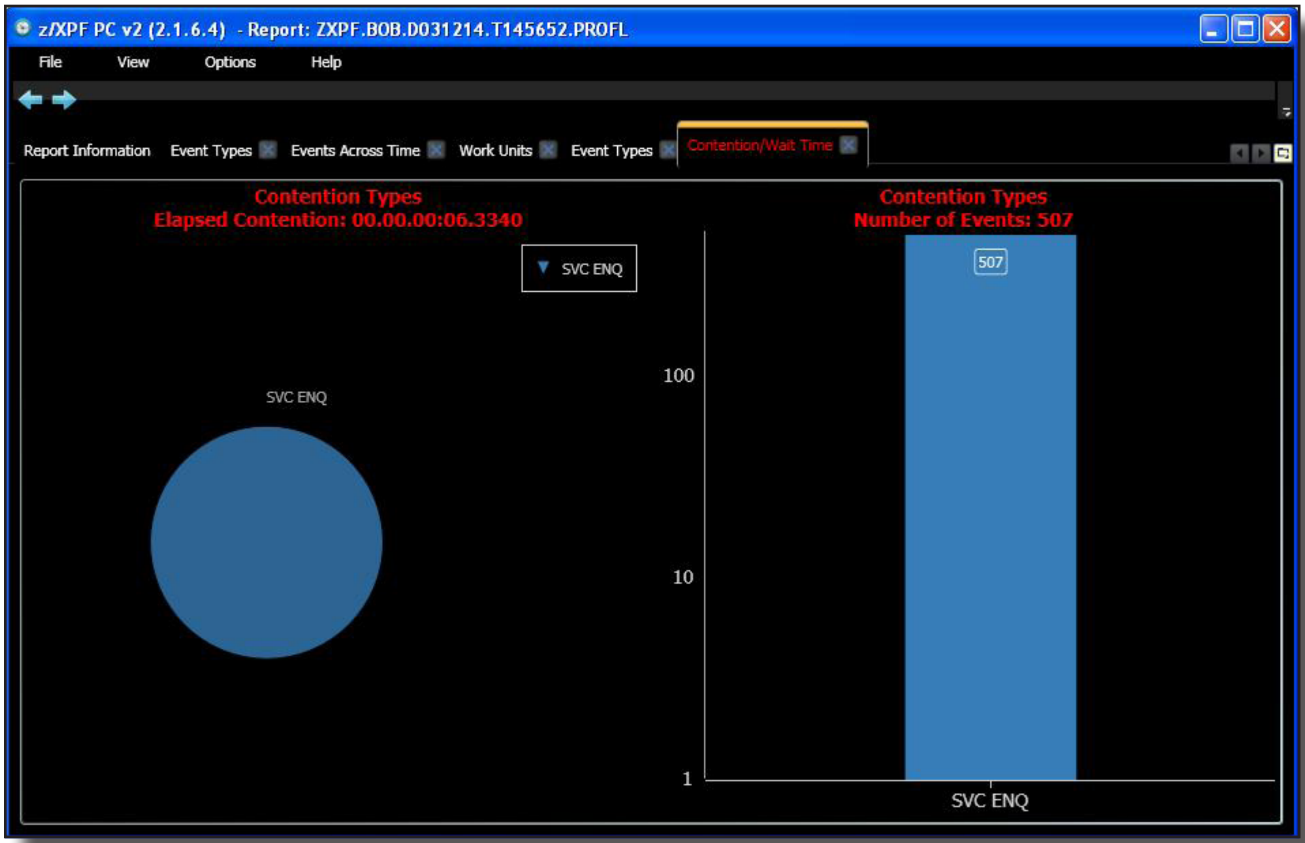


Figure 8-11-3.

I'll left-click and drill on the vertical bar on the right, to arrive at Figure 8-11-4, below:

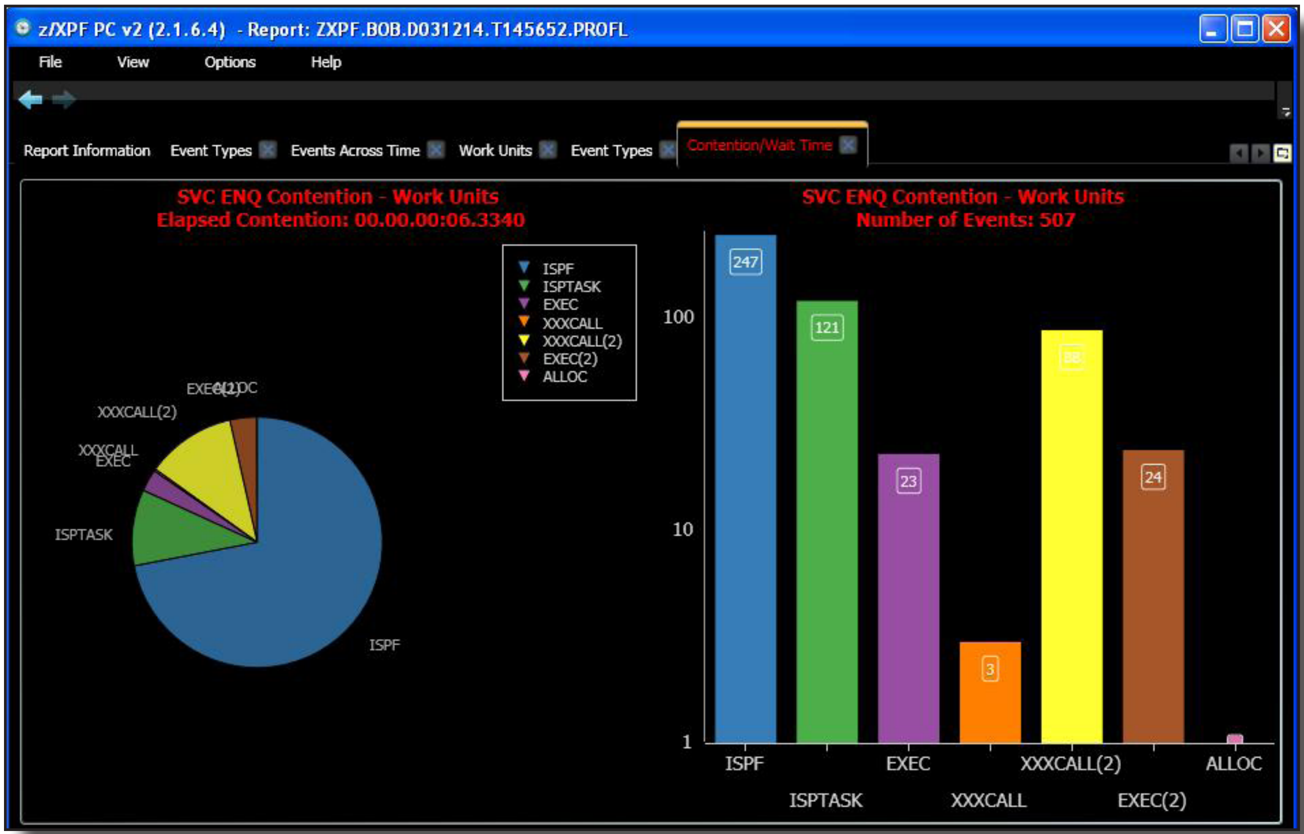


Figure 8-11-4.

8-12 SVC Breakdown

If I select "View/SVC Breakdown, I'll get a panel similar to Figure 8-12-1 below:

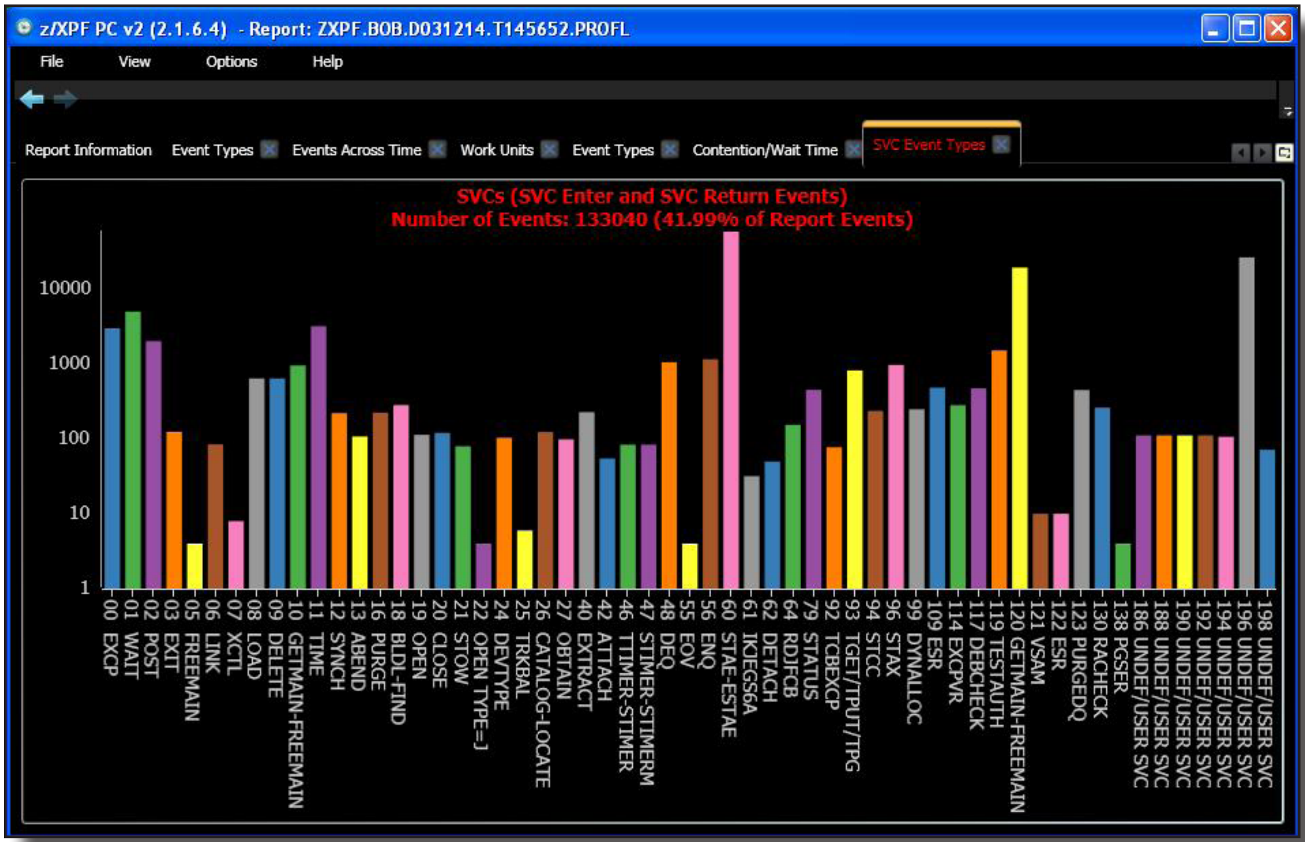


Figure 8-12-1.

8-13 PC/PT/PR Breakdown

Finally, if you select "View/PC Breakdown" (or PR or PT Breakdown), then z/XPF-PC will display Program Calls, Program Returns and Program Transfers. There's no need for me to go into each one separately because you get similar panels in each case. Here is a summary report for Program Calls (PC Breakdown), in Figure 8-13-1:

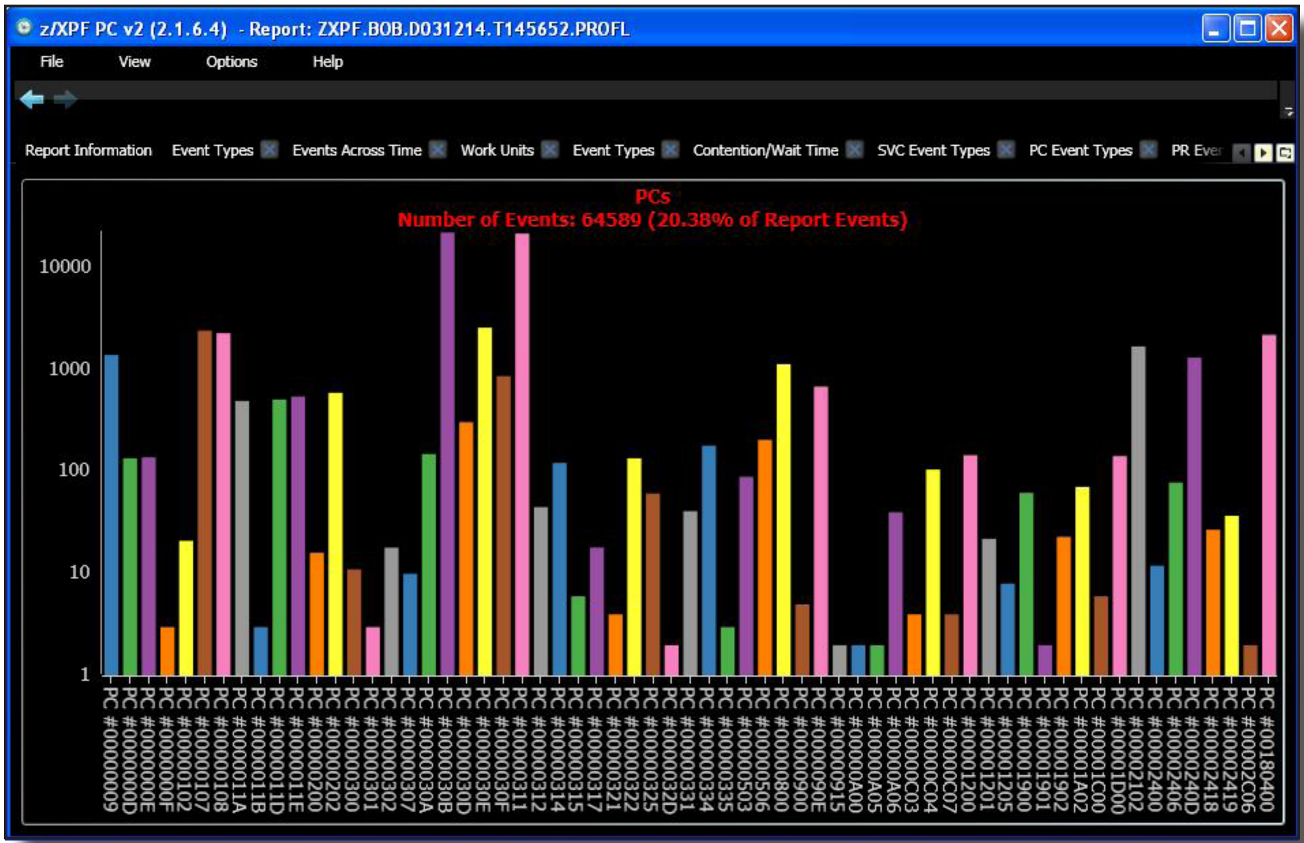


Figure 8-13-1.

Again, it's possible to drill down from any of these displays to deeper levels of the report.

Explore! Have fun. By now you should have a good idea how z/XPF-PC works. Bear in mind that we will continue to add functionality over time.

Index