colesoft™

Dump XDC®

Dump/XDC PRIMER

for Release z2.2 for z/OS

Bob Shimizu

z/XDC®

Contents

Revision date: 4th April 2025: Updated the Preface section.

dump/XDC® is a member of the XDC® (Extended Debugging Controller®) family of products

Preface

PROPRIETARY LEGEND

z/XDC® and its documentation (collectively, "Product"), including copies thereof, are the property of Mainchord, LLC DBA Colesoft ("Owner"). Use of the product is licensed from ColeSoft Marketing, Inc. ("Licensor").

The Product may be used only by those organizations that are licensed by Licensor for such use and only in the manner so licensed. The Product may not be published, reproduced, distributed or made available to third parties for any purpose without the expressed written permission of Owner or Licensor.
However, a reasonable number of copies may be made of the documentation (including the copyright notices thereon) as is necessary for the legitimate use of the Product within a licensed organization ("Customer").

Except as may be otherwise expressed in a signed agreement between Licensor and Customer, Owner
and Licensor make no representations or warranties, expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose, the warranty of freedom from rightful claims by way of infringement and the like, and any warranty as to accuracy.

WARNING! z/XDC® is a powerful tool for dynamically locating and correcting malfunctions in actively executing user programs and operating system routines. Accordingly, it is inherent in its design, that unless the use of this Product is properly controlled, then under certain conditions a malicious or careless user can use the Product to alter, subvert, counterfeit, damage or otherwise disturb the normal
execution of user programs or system routines including, under certain conditions, both its own and system security routines.

Therefore, even if advised of the possibility of loss or damages, under no circumstances shall Owner or Licensor be liable for any loss or damage whatsoever (including death) arising from the Product, whether such loss or damage be direct, indirect, consequential, special or otherwise. Further, neither Owner nor Licensor shall be obligated to indemnify in any manner against any person or organization for any loss of any kind or nature which the person or organization may experience, arising out of the use or misuse of the Product.

CONTACTING COLESOFT

The z/XDC® family of products is marketed by ColeSoft Marketing, Inc. with its principal office in Charlottesville, Virginia. If you would like more information, please contact ColeSoft Marketing as follows:

Phone:	**800-XDC-5150**
	540-456-8210
E-Mail:	sales@colesoft.com
Home Page:	http://www.colesoft.com

Our Technical Support contacts are:

Phone:	**540-456-8210**
E-Mail:	techsupt@colesoft.com
Home Page:	www.colesoft.com
FTP site:	ftp.colesoft.com

Our Customer Services contacts are:

Phone:	**800-XDC-5150**
E-Mail:	support@colesoft.com
Home Page:	www.colesoft.com

Our snail mail address is:

Address: **ColeSoft Marketing, Inc.**
414 3rd Street NE
Charlottesville, Virginia
22902
USA

ONLINE PRESENCE

ColeSoft Marketing maintains the following resources on the Internet:

[Home Page] ColeSoft's Home Page is www.colesoft.com. It provides the following services:

- General information about z/XDC
- E-mail links to both Marketing, Technical Support, and Customer Services
- FTP links for uploading diagnostic information and other files to Technical Support
- Online product delivery
- Links permitting existing customers to download a full set of z/XDC's documentation
- 24x7 self-service for temporary, short-term, license activation codes for use in D.R. tests and other emergencies
- Occasional blog posts publicizing newly implemented features and capabilities.

[YouTube] ColeSoft's YouTube page is at youtube.com/colesoftware. This is where you will find several
"how to" videos describing various aspects of using ColeSoft products. This is a wonderful resource, particularly for new Customers.

TRADEMARKS

TFS™, **XDC-TFS™**, **CDF™**, **XDC-CDF™, FASM™, base/XDC™, c/XDC™, asm/XDC™** and **dump/XDC™** are trademarks of Mainchord, LLC.

Extended Debugging Controller®, **XDC®**, and **z/XDC®** are registered trademarks of Mainchord, LLC.

Other brand and product names referenced in this document are trademarks or registered trademarks of their various holders. Use of their names herein is for identification purposes only.

The Problem With Reading Dumps

Dump reading is tedious. It can be very hard work. Even with the use of IPCS it's still a challenge to translate object code back into a source language. So, generations of programmers have sighed, rolled up their sleeves and went to work with IPCS. And, to their credit, generations of programmers have gotten good at using IPCS. They're doing a lot of translation in their heads. That's mental "overhead".

New z/OS programmers have it even worse! Not only are they learning an extremely intricate architecture, but they are learning Assembler (or C) language as well. Then, when confronted with an abend and a dump to examine they face even more hard work to understand what's going on. It's yet another level of abstraction for them to overcome.

What if this wasn't necessary any longer? What if we could make veteran IPCS users much faster? What if new programmers could reduce their reliance upon IPCS and *use one command language for both debugging AND dump-reading?*

There are two great human questions: "How come?", and "What if?"

dump/XDC began with, *"What if you could use the z/XDC command language against a dump?"* Implementing that idea took our Peter Morrison several years!

What Is dump/XDC?

First up: **dump/XDC : IPCS :: z/XDC : HLASM**

dump/XDC works within IPCS as a VERB EXIT to create a z/XDC-like environment within IPCS. dump/XDC allows the user to access z/XDC's familiar commands – and enjoy the benefits of z/XDC's power - all within a dump environment. You may also use IPCS commands in a d/XDC session if you care to.

Note: If you use IPCS commands from within a dump/XDC session, we post-process the output, and anything that looks like a legitimate address to us give you the ability to use our point-and-shoot commands, just as if you were in z/XDC itself. Nice.

dump/XDC journals the command you enter so that the state of the dump/XDC session is preserved. When you come back for another session your csect and dsect maps are replayed so that you don't have to duplicate your efforts. That's convenient.

dump/XDC allows you to use a large majority of z/XDC commands, with some obvious limitations. z/XDC's rich groups of LIST commands, the FORMAT, DISPLAY, MAP and DMAP commands all work under dump/XDC. However, because d/XDC works with "dead storage" you can't issue TRACE, TRAP, HOOK or GO.

However you CAN run a dump/XDC session in tandem with a live z/XDC session, which DOES allow you to do "what if" scenarios on the live copy of your code.

Further, as you work in a specific dump, dump/XDC is journaling your MAP and DMAP commands

(and others). If you suspend working on a dump and come back to it later, these commands are PLAYED BACK, so you only have to set up MAP and DMAP commands ONCE. Nice.

We foresee two scenarios:

1. Dumps that occur in a local environment, where the user has access to all the libraries and architecture in that local environment, and;
2. dumps that are taken outside the local environment (such as from your customer).

When you get such a dump from an outside source, you can no longer be sure that the local environment matches the remote environment. Now you have another level of abstraction to work with.

To help you with external dumps, we introduce the XDCMMEF free-ware utility program (together with full documentation and JCL examples) which you may download from the Cole Software website. XDCMMEF gathers information that is very useful to d/XDC in conjunction with a dump. It may include information about the various libraries in use at the time the dump was created.

In practice, your downstream customer might report a problem and send you a dump. You could ask the customer to run XDCMMEF and send you its output along with the dump. Then, dump/XDC will have far more information to work with, such as the location of modules. This information will help with dump/XDC's MAPPING processes.

What releases of z/OS Are Supported?

dump/XDC supports all z/OS releases from V1R13 to V2R5 and beyond, just like the z/XDC base product. That said, while dump/XDC will work on any dump within the current range of z/OS levels, *it is best that the dump be examined on the same or later release of the operating system that the dump was originally created on.*

Why This Book?

This dump/XDC Primer is designed to help you begin to use and understand d/XDC more quickly. It was written using the z/XDC Help Subsystem, a rich resource of comprehensive information on the z/XDC product line. This derivative work is NOT a substitute for reading the Help Subsystem itself. It is certainly not complete. It may even be inconsistent with the Help Subsystem.

All the information you need to truly understand d/XDC can be found under Help DUmps. I have cited specific portions of this part of our Help Subsystem at the end of each discussion unit. I have shown the minimum abbreviation possible to access each Help panel by capitalizing the essential characters. An example would be "Help DUmps EXPecteduses".

In writing this book, I relied heavily on our engineering staff. The errors herein are mine, and mine alone.

Let's get to work.

What Dump Types Are Supported by dump/XDC?

When dump/XDC gets control in a dump it analyzes the dump header to determine what type of dump it is, and whether dump/XDC can support it. dump/XDC supports these dump types:

- SVC dumps;
- SYSMDUMP dumps;
- Operator-initiated dumps;
- A dump produced by the SDUMP(X) macro;
- SLIP command dumps;
- Dumps created by the IEATDUMP macro (transaction dumps);
- A stand-alone dump.

What Dump Types Are Not Supported by dump/XDC?

dump/XDC cannot work with ICPS Active dumps. This is a use of IPCS that allows the user to look around a current, executing system. To our minds, our base product z/XDC already does a superior job of that!

For more information on this topic, please see Help DUmps Dumptypes.

What Languages Does dump/XDC Support?

Like the parent product z/XDC, dump/XDC supports z/OS-based Assembler, IBM XL C, C++ Metal C, SPC and Dignus' Systems/C - so long as the appropriate ADATA/SYMDATA/DWARF data is available. In the C environment, d/XDC allows you to use the LIST VARS command to examine your C programs variables. *IPCS cannot do this*.

How Do You Set Up dump/XDC?

dump/XDC requires the presence of a dump to begin with (duh). If you are the one initiating the dump there are certain options that allow d/XDC to do its job properly. These options may be specified in various PARMLIB members (IEADMRxx), on the SDUMP(X) macro, in the Operator CHNGDUMP command or in the Operator DUMP command.

dump/XDC will be "happiest" if the DUMP command specifies the PSA, CSA, LPA, LSQA, RGN, SUM, SWA, TRT, ALLNUC and GRSQ

If you really must see how to run dump/XDC you may skip ahead to the section, How to Invoke dump/XDC Interactively.

The CAS and the CXU

dump/XDC refers to “the place in the dump where the abend occurred” as the “Current Execution Thread.”

There are two terms in play here:

1. The Current Address Space (CAS), and;
2. The Current eXecution Unit (CXU).

dump/XDC first determines the CAS, and then the CXU within it. The CXU is a Request Block linked to a Task Control Block or a Suspended Service Request Block.

Determining the CXU is a step-wise process. Once the CXU is determined, the CAS is ignored.

dump/XDC can often determine the CAS and CXU by itself, but you may also manually specify the CXU yourself in one of two ways: You may use the “CU” shortcut command or you may specify the IPCS Literal “DXDCUCXU”.

For more information, please see Help DUmps EXEcutionthread.

The CU shortcut command

dump/XDC introduces a new two-character shortcut command “CU”. You may enter this in column one of SOME dump/XDC displays to designate the Current Execution Unit. Thus, when dump/XDC cannot determine the CXU by itself (as in the case of a stand-alone dump), you may select the CXU manually.

The CU shortcut command may be used in the output of these z/XDC commands:

- LIST DXDC: Shows a great deal of useful information on your dump.
- LIST TASKS: You may set the CXU to the newest Request Block in the nominated Task Control Block.
- LIST RBS: You may set the CXU to the nominated Request Block and the Task Control Block that owns it.
- LIST SSRBS: You may set the CXU to the nominated Suspended Service Control Block.

Once the CU shortcut establishes the CXU, dump/XDC will internally specify literal equates DXDCU-CAS or DXDXUCSU with one or the other set up to “NO”.

In operation, if a CXU is found (or if you manually specify it), that will over-ride any previous specification of CAS.

You may also set the CAS and CXU manually with the IPCS literal equates DXDCUCAS and the DX-DCUCXU, respectively. HOWEVER, these literasl must be defined prior to opening the dump/XZDC session. That’s the next topic.

For more information, please see Help SHortcutcommands Cu.

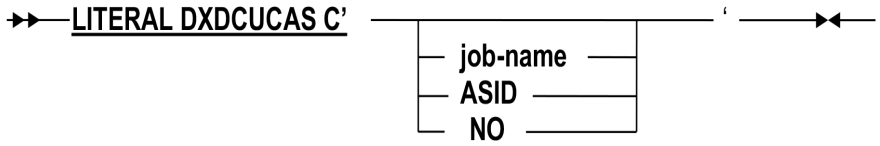
Specifying the CAS With an IPCS Literal Equate

Here are some examples:

LITERAL DXDCUCAS C'job-name' The job-name within quotes will be upper-cased for processing and designated as the CAS.

LITERAL DXDCUCAS C'asid-identifier' The address space within quotes (a hexadecimal number) will be upper-cased for processing and designated as the CAS.

Here is a rail-road track diagram for d/XDC's literal equate DXDCUCAS:



Use an IPCS literal equate to manually specify a Current Address Space for a dump under d/XDC.

C'job-name'	Pass a job name to d/XDC. This will become the Current Address Space for further processing in d/XDC. Job-name must be unique or an error message will be displayed.
C'ASID'	Pass an Address Space Identifier to d/XDC. It must be a hexadecimal value not wider than two bytes beginning with a decimal value (from 0-9). It may consist of numeric digits, alphabetic characters (including the hex digits A-F) or national characters ("@", "#" or "$"). This will become the Current Address Space for further processing in d/XDC.
C'NO'	The same as not specifying the CAS value at all.

Remarks: The entire parameter string must be enclosed within a SINGLE set of tic-marks. Under many circumstances, d/XDC is able to determine the Current Address Space (CAS) or Current eXecution Unit (CXU) automatically. However you may also directly specify a CAS with this literal equate.

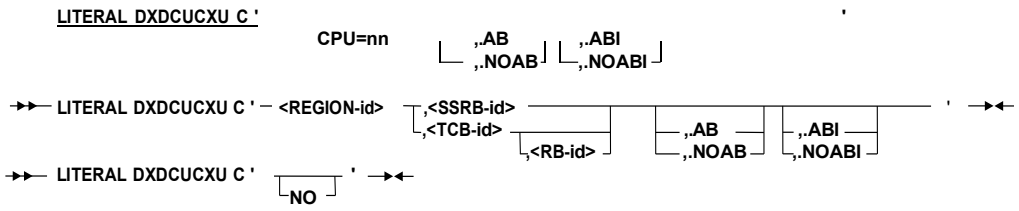
Examples: LITERAL DXDCUCAS C'JES2' ← The Current Address Space is set to JES2.
LITERAL DXDCUCAS C'FRANK2' ← The Current Address Space is set to FRANK2.
LITERSL DXDCUXAS C'0ABCD' ← The Current Address Space is set to the ASID value of X'ABCD'.

More Info: Enter Help Dumps EXEcutionthread Changing. See also Help DUmps Literals.

Figure 1: Rail-road Track Diagram for Literal DXDCUCAS

Specifying the CXU With a Literal Equate

Here's a rail-road track diagram of the IPCS command LITERAL DXDCUCXU:



Use an IPCS literal equate to manually specify a Current eXecution Unit (CXU) for a dump under d/XDC.

CPU=nn Invokes a two-hex-digit CPU identifier. *This parameter is supported only for stand-alone dumps because CPU status records exist only in stand-alone dumps. Its usage is also mutually exclusive of other parameters in this literal.*

<REGION-id> The unique region identification (which determines the home address space). Use a hex number (starting with a number (could be 0) from 1 to 0FFFF) to represent an ASID or a 1-8 character unique jobname (starting with an alphabetic (A-Z) or national character (@,#,$)). May also be a TSO userid or the name of a Started Task.

<SSRB-id> The hexadecimal address of a Suspended Service Request Block that resides in Common Storage. It must resolve to a 31-bit value.

<TCB-id> The hexadecimal address of a Task Control Block (TCB) within the dump. It must be less than 16M. It must resolve to an address in the 24-bit Private Area or in the 24-bit System Nucleus.

<RB-id> The hexadecimal address of a Request Block in the RB chain. If omitted then the default is the newest RB in the Request Block chain (the equivalent of "-1" as below).

If the RB's location is given as a hex address, it must be greater than 64K-1 (to distinguish the parameter from an ASID) and it must resolve to a number less than 16M (because RBs are below the 16M line). It must also resolve to a value within the 24-bit Private Area – AND – it must be preceded with a TCB-identifier.

- You may also specify the RB's position from the NEWEST RB in the chain by specifying a "-" (dash or negative) and an integer "nnnn" where "nnnn" may not exceed the number of RBs in the chain. An example could be "-2", which would be the second from the last RB (second newest Request Block).
- You may also specify the RB's position from the OLDEST RB in the chain by specifying a"**#**" (pound or hashtag symbol) and an integer "nnnn" where "nnnn" may not exceed the number of RBs in the chain. An example could be "#3", which would specify the third from the oldest RB in the chain.

.AB Causes the product to determine if the nominated CXU is in an ESTAE-type (or FRR) abend recovery routine. The CXU will be set to the address space, Task and RB for which that recovery routine is running.

.NOAB The opposite of .AB. No examination for an abend routine is done.

.ABI For "Abend Information". If .AB is in effect, and dump/XDC has detected that the routine is an ABEND handling routine, then the PSW and register values are taken from the RTM2WA (if an ESTAE-type routine) or the SDWA (if an FRR routine).

.NOABI Regardless of whether dump/XDC detected that the routine is an ABEND handling routine, the PSW and register values are left alone.

'NO' or ' ' This undoes any previous statement of DXDCUCXU, reverting to dump/XDC's automatic resolution of the CXU (if one existed in the first place).

Remarks: The entire parameter string must be enclosed within a SINGLE set of tic-marks. All parameter values are separated from one another by commas.

Under many circumstances, d/XDC is able to determine the Current Address Space (CAS) and Current eXecution Unit (CXU) automatically. However you may also directly specify a CXU with the IPCS literal equate. It will over-ride any automatic choice that d/XDC makes and will be saved in the dump so that you may return to the dump later and work on it again.

Once a CXU is established the CAS setting is ignored. The Current Address Space is the "home" address space set by the CXU.

Folks, there is a much easier way to change the CXU inside of z/XDC itself. Just issue LIST TASKS, and then LIST RBS and place the "CU" shortcut command wherever you like.

If, at a later time, you wish to revert your CXU setting you may specify "LITERAL DXDCUCXU C'NO' or "LITERAL DXDCUCXU C' ' " (a null parameter value).

Examples: LITERAL DXDCUCXU C'ABCD,123458' The CXU is changed to the address space named "ABCD" and the Task Control Block located at hex address "123458".

LITERAL DXDCUCXU C'ABCD,00123458,-2' The CXU is changed to the address space named "ABCD", the TCB is set to "00123458" and the Request Block is set to the 2nd newest RB in the chain.

More Info: Enter Help DUmps EXEcutionthread Changing. See also Help DUmps LiteralsHelp SHortcutcommands and Help COmmands SEt Current.

Figure 2: LITERAL DXDCUCXU

dump/XDC and IPCS Literals

dump/XDC makes use of more IPCS Literal Equates than just the DXDCUCAS and DXDCUCXU literals stated above.

The process of troubleshooting using a dump is a long, complex affair that is often conducted in several sessions over hours or even days. dump/XDC supports this and preserves the "state" of each dump so that you may pick up later where you left off. To do this, dump/XDC uses many IPCS Literal Equates. They are created by the IPCS LITERAL command and deleted by the DROPSYM command. For the definitive treatment on d/XDC's use of literal equates, please see Help DUmps Literals.

Some literals may be set by the user. Some may be set by the user or by d/XDC. Finally, some literals are set by d/XDC internally.

All user-settable literals will be of the form "LITERAL DXDCU<base-name><numeric suffix>".

Here is a table of d/XDC's user-settable Literal Equates, and what they do.

DXDCUCAS	Identifies the Current Address Space
DXDCUCXU	Identifies the Current eXecution Unit
DXDCUDSLIBn	One or more literals that link one or more dsect libraries to the dump. Allows z/XDC's DMAP command to work.
DXDCULDDNXTn	One or more literals that link library DDNAMES within a dump to library definitions occurring within Module Mapping Extraction Files received from the customer that provided the dump.
DXDCUMMEFDSNn	One or more literals that link one or more Module Mapping Extraction Files (from the XDCMMEF utility) to the dump. Allows z/XDC's MAP command to work.
DXDCUOPTIONS	This literal contains a string of user-selectable processing options. See Help DUmps Startingdumpxdc Processingoptions for more information.
DXDCUPROFILE	This literal contains the name of the initial session profile that z/XDC is to load at the start (or resumption) of a d/XDC session. See Help DUmps MIscellaneous PROfiles for more information.

dump/XDC Journals and Replays Commands For You

Every time you return to a d/XDC session on a particular dump, some commands that you set previously will be replayed for you. Here is a list of them:

- DEFINE
- DELETE
- DMAP
- DROP
- EQUATE
- LIBRARYLISTS
- MAP
- SET MAPLIBS
- SET QUALIFIER
- SET AUTOMAP
- SET CXDC
- USING
- The creation of any watch windows you create during your dump/XDC session

You may research each of these z/XDC commands by entering "Help Commands <pick-a-command>"

The XDCMMEF Utility

Cole Software offers the XDC Module Mapping Extraction Facility (XDCMMEF) utility program, along with full documentation and JCL examples, right on our website. Thus anyone – even non-z/XDC customers - may use this utility.

Why? Because sending the output of XDCMMEF along with a dump will make things much easier for the dump/XDC user.

When z/OS loads a program into storage, the layout information stored on DASD as part of the load module or program object is NOT loaded. The XDCMMEF utility collects that information and writes it to a dataset for use by the MAP command.

Imagine one of your own customers reporting a problem and sending you a dump. You may ask them to run XDCMMEF and send that to you as well. Now you'll be better equipped to solve your customer's problem.

XDCMMEF extracts load module information and writes it to a dataset. This information is very useful to d/XDC. It does NOT include the entirety of load modules. It also compresses the information gathered to reduce file size (and transmission time).

XDCMMEF obtains:
- Information about the system nucleus;
- Information about the PLPA;
- Information about the linklist;
- A module's ESD map (the layout of all csects within a module);
- Information on up to 50 user libraries (or concatenations).

The XDCMMEF utility should be run on the same CPU and on the same OS level as when the dump occurred.

Please note that an appropriate Non Disclosure Agreement (NDA) should exist between the two parties that share this information. ColeSoft Marketing assumes NO liability with regard to the transmission of XDCMMEF output between third parties.

For more information, please see Help DUmps Mappingdata, and Help UTilities Mmef.

The Define command

Most of the time dump/XDC can automatically determine the contents of a XDCMMEF dataset. It learns the name of the XDCMMEF dataset, where its dsect libraries are and where to find the JOBLIB, STEPLIB, TASKLIB and ISPLLIB libraries for the dump. When it CANNOT find these values you may specify them with three Define commands

1. DEFINE MMEFDSN - Locate the XDCMMEF file to be used along with a specific dump.
2. DEFINE DSECTLIB - Locate dsect libraries
3. DEFINE DDNTRANS - Locate JOBLIB, STEPLIB, TASKLIB and ISPLLIB.

Because you may not need to use the DEFINE command, I won't document it here. Instead, see

Help COmmands DEFine.

dump/XDC and Control Blocks

dump/XDC requires the presense of these control blocks in order to do its job.

ASVT	the Address Space Vector Table
CPPSA	The Prefix Storage Area for each CPU in the system that was active at the time of the dump.
CVT	The Communications Vector Table
ECVT	The Extended CVT
GDA	Global Data Area (related to storage mapping)
PCCAT	Prefix Storage Area (physical CPU communications area) Addresses pointed to by the CVT
PSA	Prefix Storage Area, the one at location 0, which will be the one for the CPU active at the time of the dump.
RCE	The Real Storage Manager control area (related to storage mapping)

It would also be good if dump/XDC has access to these “nice to have” control blocks:

RTCT	Recovery Termination Control Area (which contains a lot of useful information).
RTSD	An extension to the RTCT that also contains useful information.
TCBs	Task Control Blocks
RBs	Request Blocks
SSRBs	Suspended Service Request Blocks

If these “nice to have” control blocks are absent, dump/XDC may be limited in its analysis of the dump.

For more information, please see Help DUmps Contents.

How To Invoke dump/XDC Interactively

There is a fantastic resource in the z/XDC Help Subsystem that will give you step-by-step instructions on how to start dump/XDC. Please see **Help DUmps Startingdumpxdc**. I begged Dave Cole and Peter Morrison for this section and they delivered!

dump/XDC runs as an IPCS VERB EXIT and can use Literal Equates to manipulate the dump for processing.

dump/XDC supports IPCS in 3 environments:

1. IPCS under TSO
2. IPCS under ISPF
3. IPCS in batch (with some special considerations)

A normal installation of z/XDC will ensure that z/XDC's modules will be accessable to IPCS.

Note: *You you must start a version of IPCS that matches the version of z/OS that the dump was created on.*

Starting dump/XDC interactively

dump/XDC actually runs as a VERBEXIT to IPCS. This relationship is how you can use both z/XDC AND IPCS commands in the d/XDC environment. To keep IPCS out of dump/XDC's way, we ask you to invoke the VERBEXIT with the parameters "NOTOC NOTERMINAL PRINT".

IPCS is supposed to preserve this state, but it doesn't always do so. This creates a problem for dump/XDC. So, as of maintenance level **PEM-2204A** we have introduced a REXX procedure called DXDC". This procedure sets up "NOTOC NOTERMINAL PRINT", calls IPCS and on exit preserves this state.

IF you have applied maintenance to z/XDC at PEM-2204A (or later), then *the DXDC REXX procedure needs to be placed somewhere that all users may access it*. That's a job for your friendly neighborhood Spider-man (I mean "systems programmer").

Drilling down on the DXDC Process

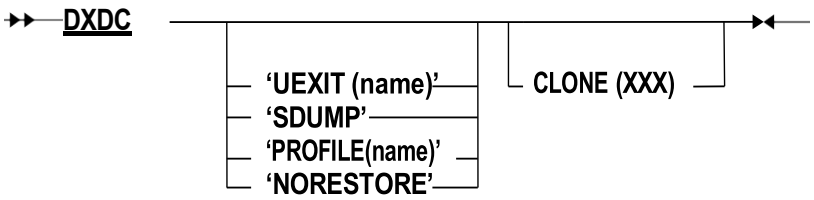
> Frank had this to say to one of our users: "dump/XDC is started by 2 procs, DXDC and DXDC9. They can be found in the DBCOLE.XDCZ22.XDCCLIST library and need to be made available to you either in a global SYSPROC/CLIST concatenation or in your own personal libraries. Once they are made available to you (and by extension IPCS), you can then issue the DXDC command in a IPCS session to start dump/XDC."

For now, you may continue to invoke dump/XDC with the VERBEXIT statement, but we are deprecating it in favor of the "DXDC" procedure, and at some point I will remove the VERBEXIT description.

Now, back to starting dump/XDC interactively. My reference here is Help DUmps Startingdumpxdc.

1. If it doesn't already exist, allocate the IPCSPRNT DDNAME to SYSOUT, DUMMY or to some FB133 sequential dataset. It will be needed by IPCS's PRINT setting.
   - The precise syntax is, "TSO ALLOC FILE(IPCSPRNT) DUMMY".
2. dump/XDC runs under IBM's IPCS as a verbexit. So, first start IPCS.
3. Next, you can run IPCS either from within ISPF or from TSO READY. This example will use the ISPF approach.
4. From ISPF issue the local menu choice to navigate to IPCS's initial panel. If you don't know how to navigate to this panel, ask your local mentor or Systems Programmer.
5. Then issue "IPCS". *You must start an IPCS version that matches the version of z/OS that the dump was originally taken on.*
6. When you press Enter, you will get to IPCS's Primary Options Panel. Select Option "0" (zero) to open IPCS Default Values panel.
7. Set the Scope variable to "BOTH". In the SOURCE line, enter the dataset name of your dump.
8. Finally, go to Option 6 ("COMMAND") and enter the d/XDC procedure, which is simply, "DXDC".

Here is a rail-road track diagram explaining DXDC:



Use the DXDC REXX procedure to establish a d/XDC session against a dump.

DXDC	This invokes d/XDC.
UEXIT(name)	Pass a user exit to d/XDC. Exits may be written to do further dump analysis. If not specified, d/XDC will attempt to load a sample load exit named "XDCXITS".
SDUMP	If d/XDC, %DXDC, or z/XDC itself abends, the system will create an SVC dump.
PROFILE(name)	Nominate a z/XDC profile other than the "factory default" as your favored profile for this d/XDC session.
NORESTORE	d/XDC stores information about a dump for use in a subsequent dump-reading session. This parameter will NOT reload this stored information.
CLONE NAME	If you have different versions of z/XDC on your system (and you probably don't) then you can identify these variants with a three-character "Clone name". An example would be "X1A" if (for some reason) you really wanted to run the old version Z1.10 of z/XDC. Usually in your world the Clone name is and always will remain "XDC". Ignore this parameter.

Remarks: This is how you "launch" d/XDC running IPCS under TSO READY, or ISPF. Parameters such as UEXIT, SDUMP or PROFILE must be enclosed with sinqle quotes. You may specify any combination of parameters, again enclosed in single quotes.

One must also allocate the DDNAME IPCSPRNT to a dummy file.

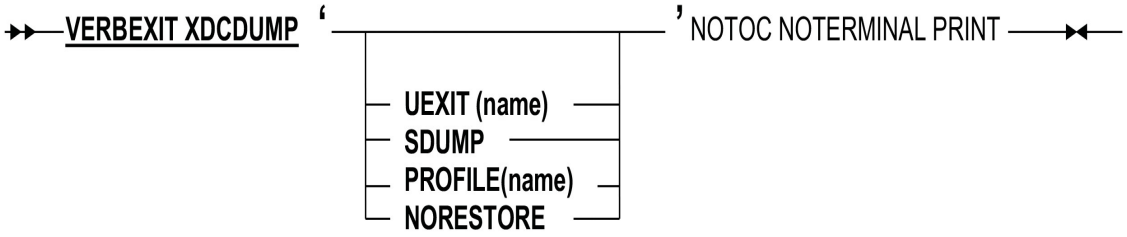
- When using d/XDC under TSO, you may issue "TSO ALLOC FILE(IPCSPRNT) DUMMY". IPCSPRNT may also be allocated to SYSOUT.
- You may also add this ddname to your ISPFPROC.
- If running IPCS in batch, add this JCL statement: "//IPCSPRNT DD DUMMY".

Examples: %DXDC Invoke d/XDC without any additional parameters.
%DXDC 'PROFILE(BOB1)' Invoke d/XDC and open the session using z/XDC's BOB1 Profile.

More Info: Enter Help Dumps Startingdumpxdc Tsoready. See also IBM Publication SA22-7594-04 (MVS Interactive Problem Control System (IPCS) Commands.

Figure 3: The DXDC command

Here's a rail-road track diagram of the VERBEXIT command, for those who aren't at Maintenance level PEM-2202F.



Use the IPCS VERBEXIT command to establish a d/XDC session against a dump.

XDCDUMP	This invokes d/XDC.
UEXIT(name)	Pass a user exit to d/XDC. Exits may be written to do further dump analysis. If not specified, d/XDC will attempt to load a sample load exit named "XDCXITS".
SDUMP	If d/XDC, the verbexit, or z/XDC itself abends, the system will create an SVC dump.
PROFILE(name)	Nominate a z/XDC profile other than the "factory default" as your favored profile for this d/XDC session.
NORESTORE	d/XDC stores information about a dump for use in a subsequent dump-reading session. This parameter will NOT reload this stored information.
NOTOC	Tells IPCS that d/XDC will not produce a table of contents.
NOTERMINAL	Prevents messages from being written to the terminal.
PRINT	Causes the messages from IPCS's responses to be written to the print dataset allocated to d/XDC.

Remarks: This is how you "launch" d/XDC running IPCS under TSO READY, or ISPF. The NOTOC, NOTERMINAL and PRINT parameters are mandatory. These parameters ensure that IPCS gets "out of the way" of /XDC. Parameters such as UEXIT, SDUMP or PROFILE must be enclosed with sinqle quotes. You may specify any combination of parameters, again enclosed in single quotes.

One must also allocate the DDNAME IPCSPRNT to a dummy file.

- When using d/XDC under TSO, you may issue "TSO ALLOC FILE(IPCSPRNT) DUMMY".
- You may also add this ddname to your ISPFPROC.
- If running IPCS in batch, add this JCL statement: "//IPCSPRNT DD DUMMY".

Examples: VERBEXIT XDCDUMP NOTOC NOTERMINAL PRINT ← Invoke d/XDC without any additional parameters.
VERBEXIT XDCDUMP 'PROFILE(BOB1)' NOTOC NOTERMINAL PRINT ← Invoke d/XDC and open the session using z/XDC's BOB1 Profile.

More Info: Enter Help COmmands Command-name. See also IBM Publication SA22-7594-04 (MVS Interactive Problem Control System (IPCS) Commands.

Figure 4: d/XDC's VERB EXIT command

If you'd rather start dump/XDC from the TSO READY prompt, Dave has created a Help page for you. Please see Help DUmps StartingdumpXDC Tsoready.

How To Invoke dump/XDC In Batch

You may submit IPCS as a batch job. You set it up, run it and then have a look at the results after the batch job completes. There is sample JCL to do that in the IPCS User's Guide.

IPCS is a TSO command, so you actually run the TSO TMP program as your batch jobstep program.

TSO in batch reads from SYSTSIN. So, in the SYSTSIN stream, you have the IPCS command followed by IPCS subcommands. One of them is END and that will end the IPCS batch job.

TSO in batch writes all line mode messages to SYSTSPRT. So, you normally see that allocated to

SYSOUT.

When dump/XDC detects that it is running under IPCS in batch, it tells XDC to work in line mode (by supplying a UCI exit), and picks up its input using a TSO macro that reads the next line in SYSTSIN, and writes all output in line mode to IPCS (or the DXDC Process), with all dump/XDC output bracketed by a header and trailer (making it easy to find). So, to invoke dump/XDC you use an input stream with the IPCS VERB EXIT subcommand to run d/XDC, followed in the input stream by the dump/XDC commands themselves. To terminate d/XDC and re-enter IPCS, you use the z/XDC END command.

Putting it all together, here is some sample JCL. We will assume that the user has already used normal IPCS to make a specific dump current (this information is stored on the user's IPCS dump index dataset), and that dump/XDC is in the LPA or linklist:

```
//IPCSJOB JOB 'acctinfo','IPCSU1 output',MSGLEVEL=(1,1),
// MSGCLASS=A,CLASS=J,NOTIFY=IPCSU1
//*
//IPCS EXEC PGM=IKJEFT01,DYNAMNBR=20
//IPCSDDIR DD DSN=IPCSU1.DUMP.DIR,DISP=(SHR)
//IPCSPARM DD DSN=DEPTNUM.IPCS.PARMLIB,DISP=SHR
//SYSPROC  DD DSN=SYS1.SBLSCLI0,DISP=SHR
//         DD DSN=<xdc-procedure-library>,DISP=SHR
//SYSUDUMP DD SYSOUT=A
//IPCSPRNT DD DUMMY
//SYSTSPRT DD SYSOUT=A
//SYSTSIN  DD *
PROFILE MSGID
IPCS NOPARM
<IPCS subcommands>
DXDC <operands>
<dump/XDC commands>
END /* exit dump/XDC */
<other IPCS subcommands>
END /* exit IPCS */
/*
```

If z/XDC is NOT installed into LINKLIST and the PLPA, and you want to use dump/XDC in batch, a // STEPLIB statement will be required. Here is a sample, which is based upon the default installation names:

```
//STEPLIB DD DISP=SHR,DSN=DBCOLE.XDCZ22.LINKLIB
          DD DISP=SHR,DSN=DBCOLE.XDCZ22.LINKLIBE
          DD DISP=SHR,DSN=DBCOLE.XDCZ22.XDCPLPA
```

Note that this sample JCL has been taken from the IPCS User Guide with appropriate changes to show how you would use dump/XDC in batch.

For more information, please see Help DUmps Startingdumpxdc Batch.

How To Use dump/XDC

Once dump/XDC is up and running, it's just like using z/XDC itself! You can use z/XDC's powerful LIST commands, FORMAT, DISPLAY, FIND, MAP and DMAP. We intend the interface to be indistinguishable from a "normal" z/XDC session.

Here, just for fun, is the output of the WHERE command in my dump/XDC session:

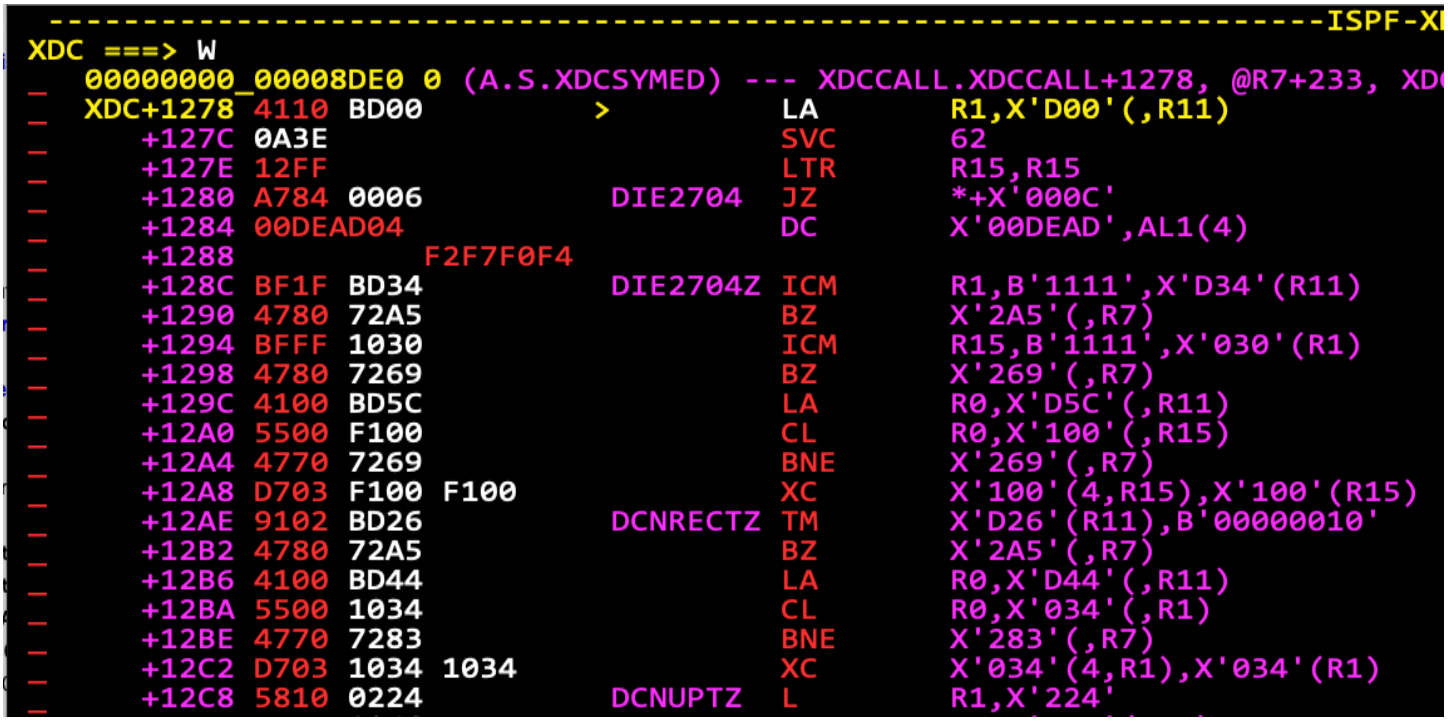


Figure 5 - Output of the WHERE command under d/XDC

This looks like a typical z/XDC session, doesn't it? Actually I was surprised, because I expected to be in the XDCSYMED program (the code I use to teach all my z/XDC classes).

A quick question to Frank yielded this: "The dump you are looking at is an intentional abend that I set up in XDCSYMED, but XDCSYMED *is protected by XDC as a recovery environment* (via XDCCALL). When the dump actually took place, execution had already gone into XDC via XDCCALL, so when you isue a WHERE command, it's in XDCCALL. For dump/XDC to be effective, just like XDC, the user has to have some understanding of the environment and context of what they are looking at.

With dump/XDC you can, and should, change the "Current eXecution Unit" to reflect that you want to be looking at the XDCSYMED TCB. Once that is done, a WHERE will work as you currently expect it to."

This was a perfect opportunity for me to demonstrate the CU Shortcut command.

First, I issued LIST TASKS:

Figure 6: Output of the LIST TASKS command

You can see above where d/XDC "chose" XDCCALL as the Current Execution Unit by yellow highlighting. I was tempted to put the CU shortcut right next to the XDCSYMD Tack in TCB#5, but Frank pointed out, "That would be a mistake. Instead issue LIST RBS TCB#n (or use the "L" shortcut command) to see the Request Blocks under the task. Then put your CU on the proper RB."

There are two ways to see the RB structure in this situation. The easiest way is to put an "L" next to the desired task (always my first choice). The harder way is to issue "LIST RBS TCB#5 (at least in THIS address space). So, I put an "L" next to the desired task:

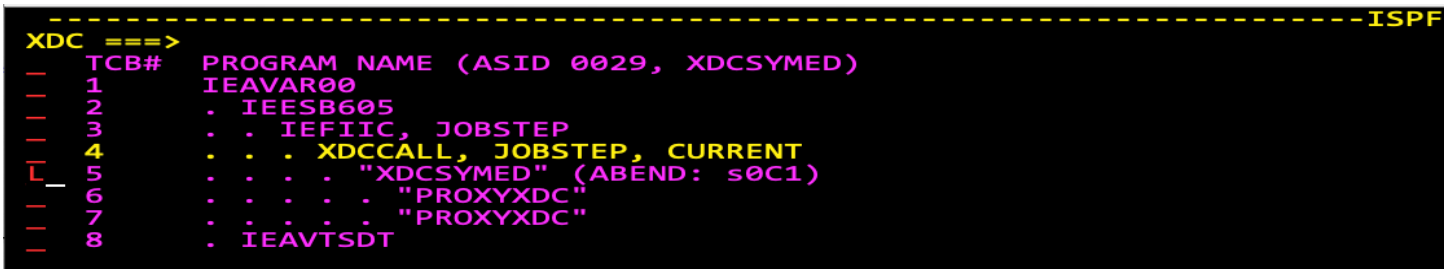


Figure 7: Using "L" to see the RB structure

Now I FINALLY get to use the CU shortcut. I put it next to RB#1,

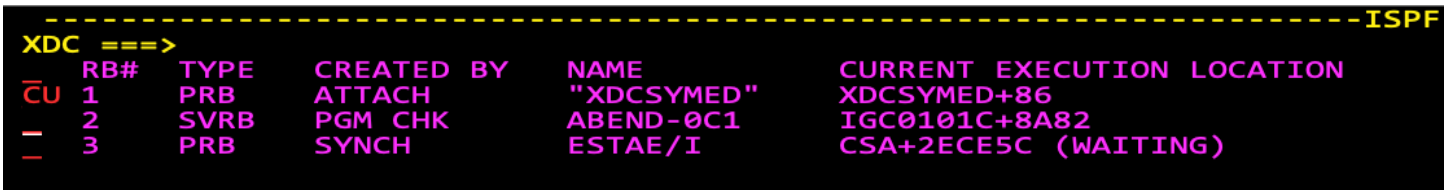


Figure 8: Using the CU shortcut command.

dump/XDC reports the change of the Currect eXecution Unit:

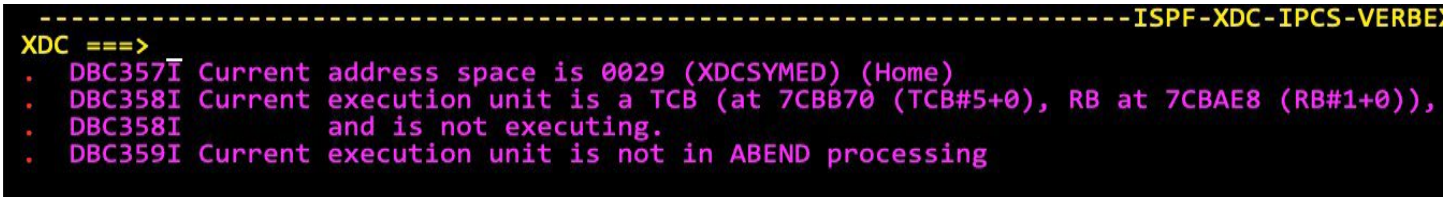


Figure 9: Output of the CU shortcut command.

z/XDC needs to know where the ADATA resides for any program. In my world I have default settings that always map XDCSYMED for me. You probably don't have that arrangement, so I'll show you the SET MAPLIBS command, which establishes where z/XDC should find the ADATA for my copy of XDCSYMED. Your settings in your environment will be different, but:

```
 ------------------------------------------------------------------------ISPF-XDC-I
XDC ===> SET MAPLIBS DBCOLE.XDCZ22.XDCADATA_

   MAPPING INFORMATION IS AVAILABLE FROM THE FOLLOWING LIBRARIES AND DATASETS:
   #1    (PDS) DBCOLE.XDCZ22.XDCADATA

   THE FOLLOWING MAPLIB LISTS ARE SAVED:
       DEFAULT
```

Figure 10: Output of the SET MAPLIBS command

Now I issue my MAP command:

```
 ------------------------------------------------------------------------ISPF-XDC-IPCS-VE
XDC ===> map xdcsymed. sys1.csw.linklib uselds_

   READING THE MODULE MAP FOR XDCSYMED FROM SYS1.CSW.LINKLIB(XDCSYMED).
           (READING ESD INFORMATION FROM THE LOAD MODULE. METHOD=BSAM.)

   READING THE CSECT MAP FOR XDCSYMED.DBCSYMED FROM DBCOLE.XDCZ22.XDCADATA(DBCSYMED).
           (READING ADATA RECORDS FROM A SYSADATA FILE. METHOD=BSAM.)

   The following maps have been built:
   MAP TYPE          LOCATION          NAME
_  LOAD MODULE       0000EF88          XDCSYMED
_  CSECT (ADATA)     0000EF88          DBCSYMED
```

Figure 11: The MAP command

NOW let's try a "Where" command:

```
 ------------------------------------------------------------------ISPF-XDC-IPCS-VERBEXIT----------------------
XDC ===> w
_  00000000_0000F00E 8f (A.S.XDCSYMED) --- XDCSYMED.DBCSYMED+86, @R15+86, XDCSYMED+86, PRIVATE+D00E
_  DBCSY+86>         0026              +38                                                  *..*
_       +88                  +         DS    0F                 ALIGNMENT                        01-#ENTER
_       +88                  +SYMEDRSA DC    (72)X'00'          LOCAL SAVE AREA                  01-#ENTER
_       +88o@R13     00000000 0000B468 00000000 00000000  00000000 00000000 00000000 00000000  *...................
_       +A8o         00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *...................
_       +C8o         00000000 00000000                                                         *........*
_       +D0                  +E0283END DS    0H                                               01-#ENTER
_       +D0 A7F4 0004        +         J     over0283        Skip entry literal pool    03/16 PRF 01-#ENTER
_       +D4 00000000         o                                   *....*
_       +D8                  +         LTORG ,               Literal pool for entry code  03/16 PRF 01-#ENTER
_       +D8                  +over0283    DS    0H                                      03/16 PRF 01-#ENTER
_       +D8 47F0 D078                     B     OLDIOZ             SKIP SOME JUNK        05/97 X33 08440000
_       +DC                               TITLE 'OLDIO -- I/O Routines - 24-Bit Interfaces'  07/04 Z16 08540000
_       +DC                  +            @TITLE '[ALVL=3] OLDIO -- I/O Routines - 24-Bit Interfaces'   X01-00000
_       +DC                  +                                                            09/98 X34
_       +DC                               @OLDIO MF=LIVE,          THIS IS FOR REAL      05/97 X33*08630000
_       +DC                                    R13=SYMEDRSA        "USING SYMEDRSA,R13" EXSTS 05/97 X33 08730000
_       +DC 0000F07C         +@EXLST      DC    A(EXL24)                                   05/97 X33 01-@OLDIO
_       +E0                  +            DS    0F                 ALIGNMENT              07/04 Z16 01-@OLDIO
_       +E0                  +OLDIO       DS    0F                 INTERNAL NAME          07/04 Z16 01-@OLDIO
```

Figure 12: The Where command after changing the CXU and mapping

Success! Now (unlike in IPCS) I can SEE my source code. You can do exactly the same in C language as well, AND you can (unlike in IPCS) SEE your variables too. Nice.

Hey, I almost forgot to show you where dump/XDC detects the ABEND. From the output of my "WHERE" panel I issue "F -2":

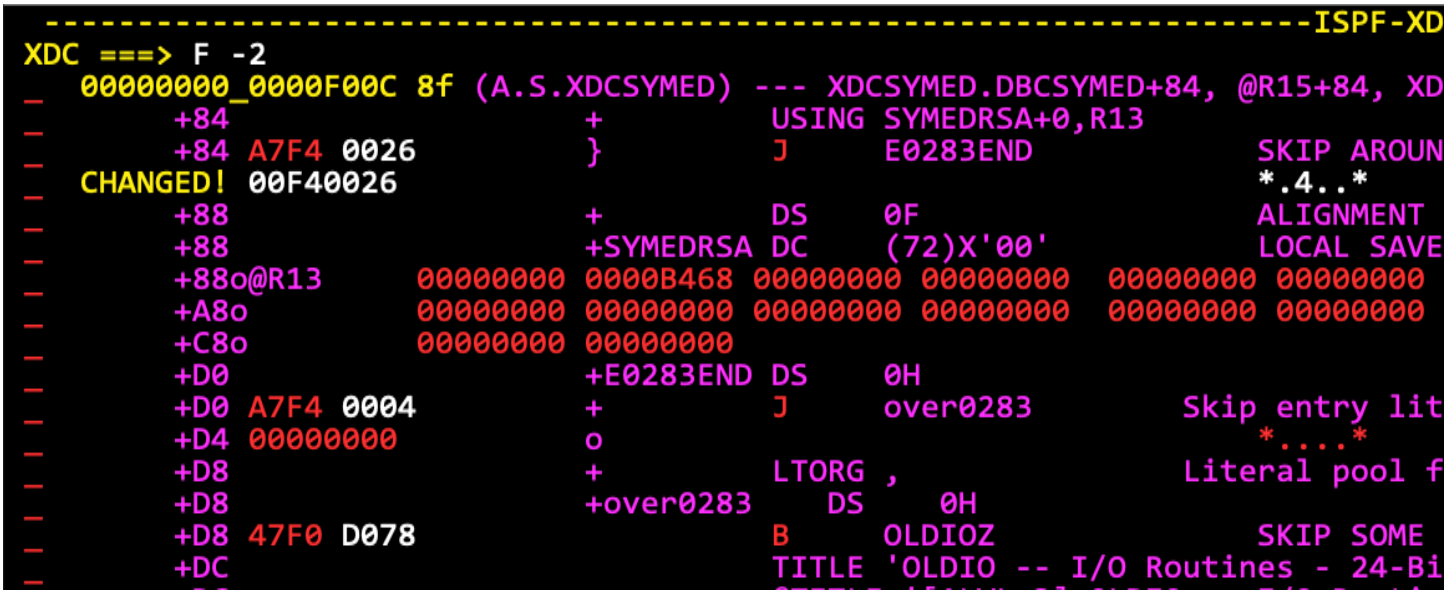


Figure 13: Showing the ABEND in XDCSYMED.

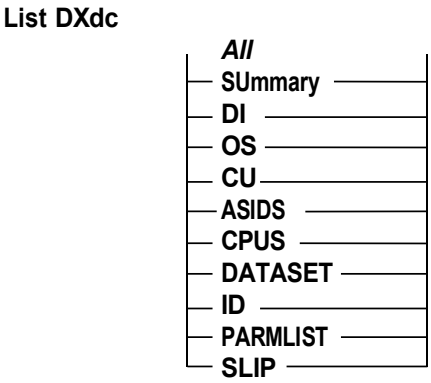
The clue is the "CHANGED" word. It clearly shows where Frank zapped in X'00' to create the S0C1 abend that caused this dump.

The LIST DXDC Command

The LIST DXDC command will show you a great deal of information about your dump.

The default behavior for LIST DXDC is to show EVERYTHING, but you may qualify that with additional parameters if you wish. This will “pare” the output to your immediate needs. For more information please see Help Commands LISt DXdc.

Here is a rail-road track diagram for LIST DXDC:



Display information about the current dump environment in a d/XDC session.

All	Default. Displays the entirety of the command, and all of the attendant parameters (CPU, ASID, SLIP). This is the default behavior.
Summary	d/XDC displays a terse output of this command.
DI	The Dump Information Section will be displayed, including the dump title, DSN, original dump DSN, IPCS type and whether the dump is from the host system or not.
OS	The Operating System Information Section will be displayed, including the OS name and version, the System name (from SCVTPNAM) and the clone name from (ECVTCLON) and the SYSPLEX name.
CU	Information about the Current Execution Unit and Current Address Space will be displayed
ASid	Displays the address space identifiers (ASIDs) and the integer they have been assigned from the Address Space Vector Control Table.
Cpu	This displays all the CPUs seen in the dump. The Current eXecution Unit (CXU) is highlighted. **Important:** The output of the CPU parameter allows you to use the “CU” shortcut, which will change the Current eXecution Unit to whichever CPU you enter the CU next to. The line showing the CXU is always high-lighted.
Dataset	Information about the dump dataset will be displayed. Information includes the DSN, block-size and LRECL, the number of blocks and records, etc.
Id	Displays the dump’s CPU id and timestamp information in both raw-hexadecimal and human-readable formats.
PARMLIST	The SDUMP Plist Section (if available) will be displayed.
Slip	If this is a SLIP-type dump, d/XDC shows the SLIP command text. For a PER dump, (IF, SBT, SA, ZAD) d/XDC shows hardware related PER information (if available). For COMP d/XDC shows the ABEND and reason codes. For MSGID you will see the message text For ERROR no results are shown.

Remarks: The default action of this command is to show ALL. Alternatively, one may be able to issue LIST DXDC Summary to see only the summary output. Note that each of these parameters may be prefixed by “NO”, such as “NOASID” and so forth. I don’t know why our users would go to the trouble of typing that, and if I added it to this “railroad track” the result would have been ugly.

Examples: LIST DXDC SU ↵ See summary info.
L DX ↵ Show ALL information.

More Info: Enter Help COmmands LISt DXdc.

Figure 14: Rail-road track diagram of the LIST DXDC command

Here is an example of the output of LIST DXDC:

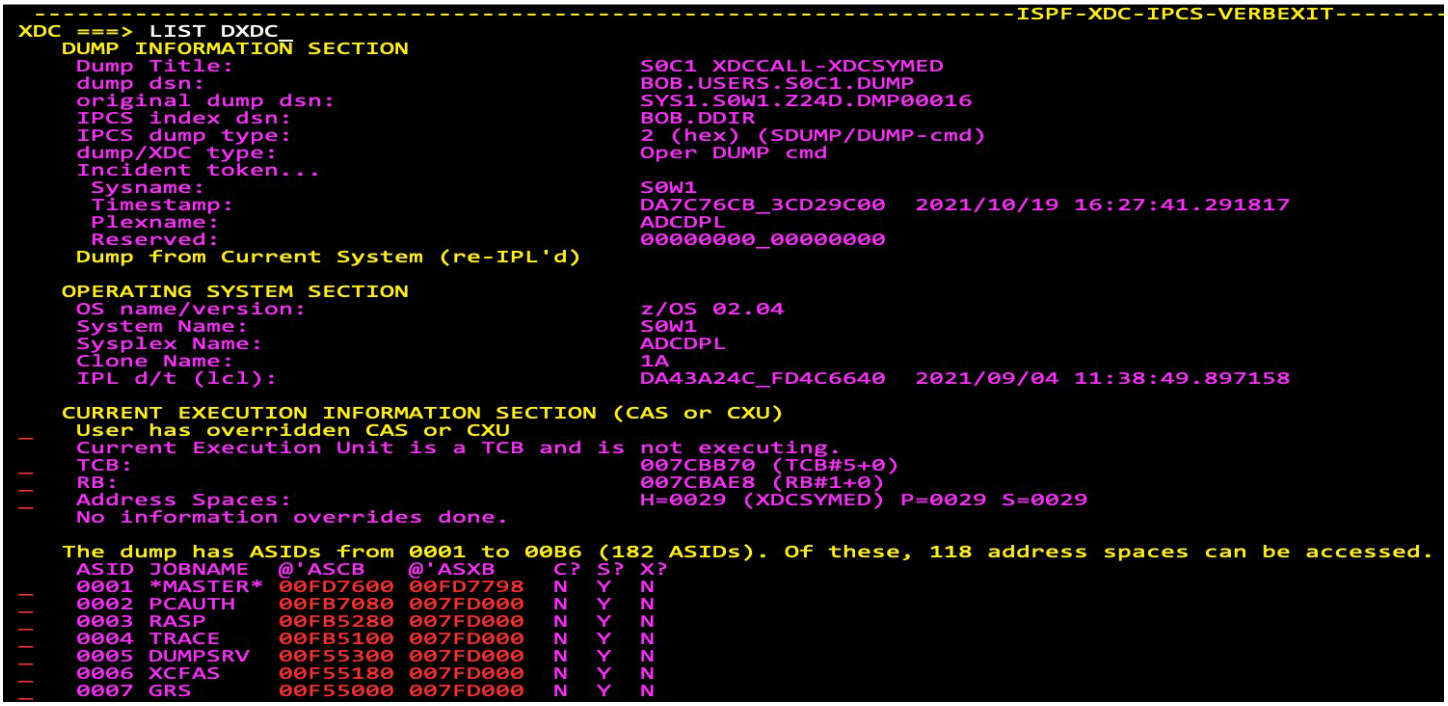


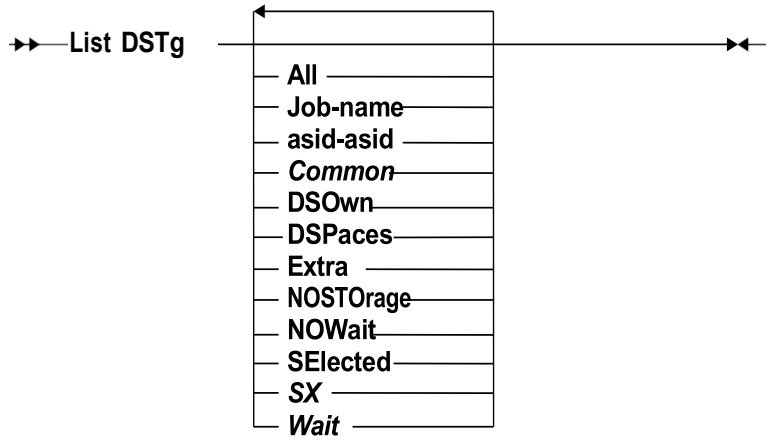
Figure 15: Output of the LIST DXDC command

There's a lot more to the output of this command, but here's a portion of it. Notice the "red portion" of the display in the bottom of the screen. This is an opportunity to use z/XDC's point-and-shoot commands to investigate any desired storage (for example, "D ", or "D-space" for a display command).

For more information on LIST DXDC, please enter HELP COMMANDS LIST DXDC.

The LIST DSTG Command

List DSTG displays storage allocation ranges and dataspaces that are present in the dump. The output of this command can be rather verbose, so there are many parameters that will allow you to "tune" the output of this command, showing you only what you need to see. In other words if you issue LIST DSTG without any parameters, you're effectively issuing "LIST DSTG ALL", and you're going to be paging down for a while!



Display storage allocation ranges and dataspaces that are present in a dump.

Omitted	The default output is "COMMON" and "SX".
All	Shows storage allocation ranges and dataspaces present in the dump.
Job-name	When given a job-name (such as "JES2", or the name of a batch job or a TSO userid) d/XDC will display storage owned by that address space. This parameter will also accept a hexadecimal ASID number instead of a job-name.
asid-asid	Displays storage allocations for a range of address space identifiers. Typically these will be hexadecimal numbers but they may also be decimal numbers (expressed with a suffix of "N"). Example: "LIST DSTG 001E-001F" Hint: You can use z/XDC's LIST ASID <jobname> command to get the address space identifier. In my world, "LIST ASID JES2" returns X'001E'.
Common	All Common Storage allocations present in the dump will be displayed.
DSOwn	All address spaces present in the dump that own one or more dataspaces in the dump will be displayed.
DSPaces	Displays the names (and storage ranges) of dataspaces owned by a particular job name (such as JES2) or ASID (such as X'001E').
Extra	If a Current eXecution Unit exists in the dump, then this parameter will display storage allocations owned by the CXU's home, primary or secondary address spaces is not chosen by any other parameters.
NOSTOrage	Suppresses the extent-by-extent display of storage allocations, showing only summary information.
NOWait	If dump/XDC's information gathering routine has not completed when this parameter is entered, then this command will fail and display an error message.
SElected	Displays storage allocations for address spaces in the IPCS verbexit's "address space selection" option. Typically this will be "ASAX1" in cases wherein the ASXB control block can be accessed. When I tried this command I saw MASTER, CONSOLE and the address space containing the CXU (ASIDs X'0001", X'0009" and in my case X'0029).
SX	Produces output that combines the parameters "Selected" and "Extra" which are otherwise mutually exclusive. Think of it as "Selected-eXtra".
Wait	Will pause the command's execution while dump/XDC's asynchronous information gathering routine runs and until dataset information becomes available.
Remarks:	This command will display ONLY the storage allocations that are present in the dump. If a jobname exists with a name that can also be resolved to an ASID (Jobname=001E exists but the ASID 001E ALSO exists), then the ASID for that job-name must be specified. The arrow above the railroad track pointing left means that you may enter more than one of the parameters at a time. If no parameters are given, the default actions are ALL, DSOWN and SX.
Examples:	L DST ↵ See all ranges of storage extents and dataspaces for all ASIDs on the system in numeric order by ASID. L DST DSPACES JES2 ↵ Show dataspaces owned by the job-name JES2. LIST DSTG COMMON ↵ Show storage extents for all common storage allocations.
More Info:	Enter Help COmmands LISt DXdc.

Figure 16: Rail-road track diagram of the LIST DSTG command

Peter stressed to me that the "All" parameter doesn't show ALL storage in any particular system, *ONLY that which is present within the dump*. OK, that makes sense.

For more information, please enter Help COmmands LISt DSTg.

Mapping CSECTs and DSECTS in dump/XDC

z/XDC's MAP and DMAP commands may be used in dump/XDC. When building maps, z/XDC gathers information from a number of sources. These may be ESD records (External Symbol Directory), Assembler ADATA or SYM records, and C language DWARF datasets. When these "building blocks" are unavailable, the type of map(s) desired cannot be built.

The output of XDC's Mapping Extraction Facility program gathers ESD information, producing juicy wholesome goodness in dump/XDC's displays. As the vendor, you will have to provide your ADATA, SYMDATA and DWARF/SOURCE files.

There are special considerations which apply when you use MAP and DMAP in d/XDC. For the best advice, please see Help DUmps MAppingdata.

Using Watch Windows and Profiles under dump/XDC

You can also create/delete Display Windows within the terminal screen, just as you can in z/XDC. Normally, PFK13 performs a "SET WINDOW CREATE" command. PFK1 performs a "SET WINDOW DELETE command" Both of these command work with the position of the cursor on the screen to create or delete a Watch Window.

We intend to make dump/XDC as natural an interface to the user as z/XDC itself. *It all just works.*

You may also access the Profile Facility, which interprets your symbolic actions in the Profile Menu to execute SET commands on your behalf. Such SET commands allow you to change dump/XDC's default "worldview" of your dump, alter PFkey settings, view storage in EBCDIC or ASCII and on and on.

You may also influence this by invoking z/XDC's PROFILE READ command to load your own favorite profile. You may also specify your profile with an IPCS literal equate of the form DXDCUPROFILE. For more information please see Help Dumps MIscellaneous PROfiles.

Remember that when using dump/XDC you're looking at "history" so you can't perform TRACES, or set breakpoints. But you may run multiple sessions: A dump on one side and a live z/XDC session on the other. One of d/XDC's design points is to run a dump/XDC session in tandem with z/XDC. We hope our customers will exploit this.

Here is a suggestion. Just as you can do in a regular z/XDC session, this command string would be a very useful way to begin: "LIST BEA;;LIST TASKS;;LIST RBS". The two semi-colon command delimiters will create a blank line between these command's output for readability.

This command string shows the Breaking Event Address (how did we branch here?), the Task structure and the Request Block structure for the current Task.

Thus, in one command sequence you would know a LOT about your current environment. You might also choose to use the WHERE command (also available as the "W" shortcut command) to automatically FORMAT the command just after the abend occurred.

Here's a screen shot from my own dump/XDC session:

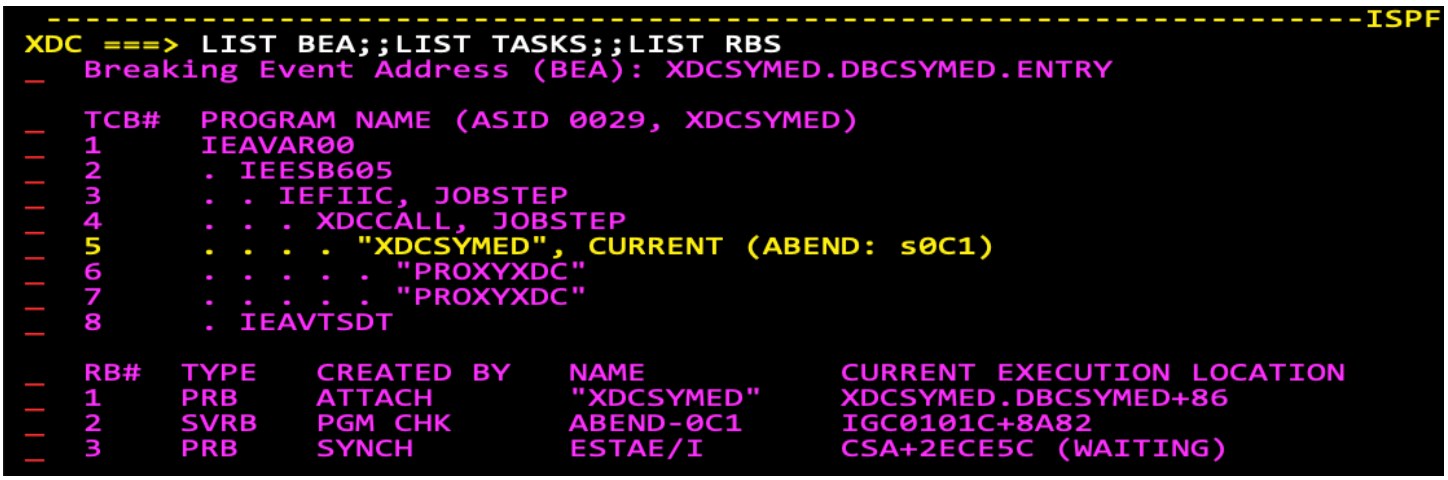


Figure 17. Three Useful z/XDC Commands "Stacked"

Issuing IPCS Commands from within your dump/XDC Session

During your dump-reading session you may wish to enter additional IPCS commands. You can END the d/XDC session and then enter IPCS commands on the IPCS command panel (or the IPCS line-mode prompt under TSO), OR you can issue these commands within the d/XDC session.

To do this issue IPCS <IPCS-command-name>. dump/XDC will port the command to IPCS on your behalf. In the example below I've issued IPCS 'verbx vsmdata' inside my d/XDC session:

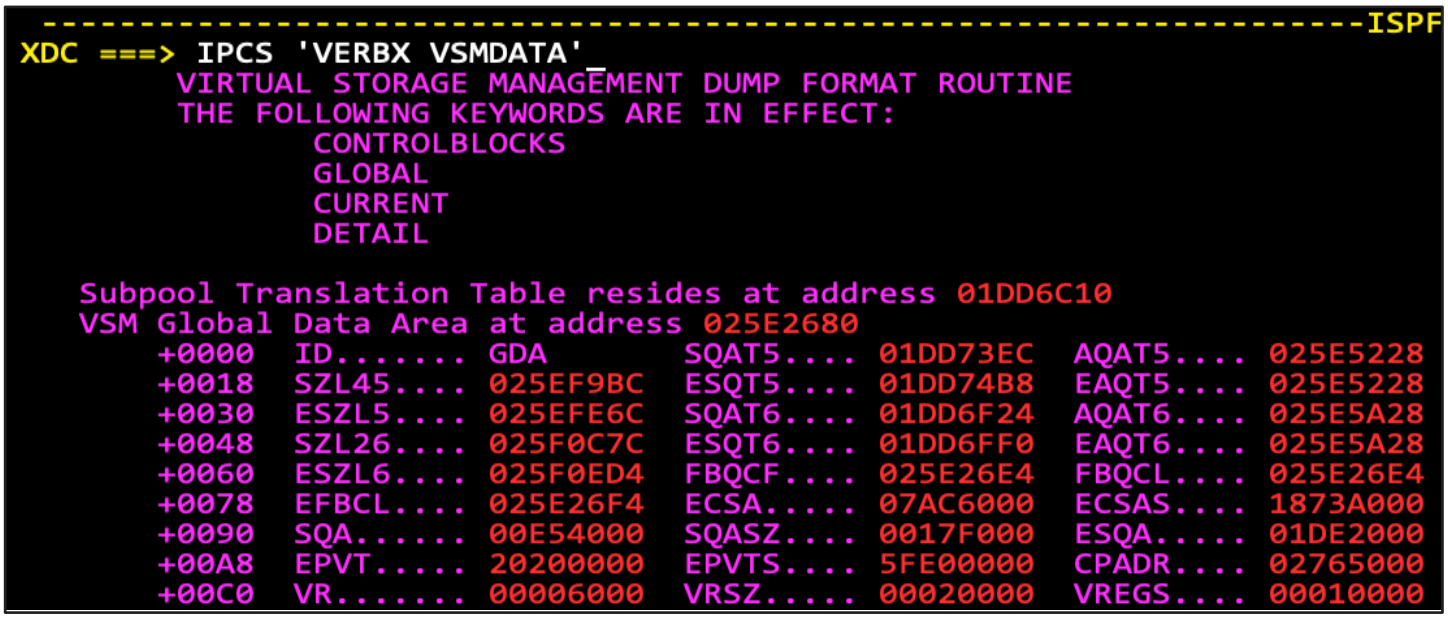


Figure 18. invoking the IPCS Verbexit command inside the d/XDC session.

Please notice the red hexadecimal numbers above. This is an opportunity to use z/XDCs "Point and Shoot" commands. So, if you want to, try putting a "D " (d-space") on top of any of these addresses. You'll see that storage.

Here's another advantage to using dump/XDC: ANY IPCS command that you issue within dump/XDC will receive our post-processing. That means that anything we display that looks like a legitimate address will allow you to use point and shoot commands.

For more information on point-and-shoot commands, please see Dave Cole's presentation on these commands at our website: https://www.colesoft.com/training/training-videos

Being able to issue IPCS commands from within a d/XDC session is a good thing, and being able to use VSMDATA is also a good thing - BUT - z/XDC's **LIST SUBPOOLS** command is a FAR better way for you to view this information. To get an idea how much more powerful LIST SUBPOOLS is, please enter HELP COMMANDS LIST SUBPOOLS. You're welcome.

You May Customize dump/XDC's Actions

dump/XDC makes a lot of intelligent choices on your behalf - so that you don't have to. However, if d/XDC isn't giving you quite what you want, then you may specify options to improve d/XDC's actions.

You can do this by stating an ICPS literal equate of the form DXDCUOPTIONS. These options are "words" separated from one another by blanks or commas. The maximum width of this literal equate is 256 characters.

Some of the words you may use in DXDCUPOPTIONS include:

ASRTCT	Only address spaces on the dump that are also listed in the RTCT will be made available.
ASAX1	Only address spaces where d/XDC can retrieve the ASXB will be made available.
ASALL	All address spaces in the dump will be made available.

There many more options that you may use but let me warn you that most of them are already set to useful default options that you probably won't want to change. The complete list of them is beyond the scope of this Primer, but are excellently discussed in Help DUmps Startingdumpxdc Processingoptions.

Error Level vs. Retry Level In dump/XDC

For decades, z/XDC has kept a lot of information on two distinct environments:

1. The Error Level environment (the current Request Block, the PSW, the registers and a slew of control blocks and other information) relating to the state of the program at the time of the error (an abend, a breakpoint or the end of a Trace command).

2. The Retry Level is all the same information that will become relevant when the job resumes. In a z/XDC environment this might be when the user issues a "GO" or a Trace command.

Under z/XDC the retry level can be a completely different place from where the error occurred. For example, a non-authorized program getting an abend on a GETMAIN. The retry level will be after the

GETMAIN SVC in the RB level that issued the GETMAIN, whereas the error level could be the abend in the GETMAIN SVC itself.

These differences are mainly to prevent system integrity issues that have nothing to do with z/XDC itself.

Under dump/XDC, execution cannot be resumed. As a result, the Error Level means the Request Block in which the error has occurred (that hasn't changed), but the Retry Level will now be the same as the Error Level, with the PSW "backed up" by the value of the Error Level's PSW's Instruction Length Code field.

For more information, please see Help Executionlevels, and Help DUmps EXEcutionthread Erroran-dretrylevels.

dump/XDC And Lock Analysis

The base product z/XDC offers the LIST LOCKS and SET LOCKS commands. This comes from our product's recent ability to debug SRB-mode code.

dump/XDC also performs lock analysis in a dump at two levels:

1. Locks held by the Current eXecution Unit holds (if any), and;
2. Locks held by all address spaces and CPUs in the entire system.

Locks are beyond my own understanding so it would be vastly better to see Help DUmps MIscella-neous Locks.

dump/XDC and SRB-Mode Programs

dump/XDC fully supports SRB-Mode programs. If a dump is taken while a CPU is in SRB-mode, even though there is no control block in z/OS that indicates that the CPU is in SRB-mode, dump/XDC correctly handles it.

How to End a dump/XDC Session

You end a dump/XDC session with the END command (or PFKey) just as you do with z/XDC. You may still issue the NOASK paramater if you like. When you end the session, it's over.

However, dump/XDC saves a lot of things in the IPCS index so that you can pick up where you left off whenever you return to your dump. Upon re-entry into a d/XDC session d/XDC will "replay" lots of commands automatically. For example, if you MAP'd your source or used the DMAP command for data structures these will be shown to you without additional effort on your part. Nice.

dump/XDC's Current Limitations:

- d/XDC will NEVER be able to support the LIST ENQ command. The relevant control bocks are Object Code Only from IBM and the API that z/XDC uses for the LIST ENQ command cannot be used against a dump. However, if the proper information has been gathered, IPCS has good reporting capabilities in this area.
- The ability to display access lists associated with a unit of work (the LIST ACCESSLISTS command) is not yet included. This is planned for a future release.

How To Get More Information

The entirety of dump/XDC's documentation may be found under Help DUmps. You may also enter Help Whatsnew Z22 Dumps.

I prefer to use LIST HELP DUMPS. That shows me all the subtopics within that portion of help. Put an "L" next to any subtopic and drill down to see the next level of subtopics. When there's only one line displayed, place an "H" next to that and press Enter. Now you have quickly drilled down to what you need to read.

It's easy to start reading z/XDC's huge Help Subsystem: Start a debugging session and enter the Help command. There are many useful hyperlinks within the Help Subsystem that will allow you to navigate around inside Help DUmps. There is a useful discussion on using z/XDC's Help System in the z/XDC Primer, available at our website under Support/Documentation.

How to Access our Support People

Of course, we are always glad to help you directly. You may contact us at:

- bshimizu@colesoft.com I serve as your First Level support rep. I also make sure that we keep engaged with you on your problem until it is resolved. You may reach me directly at (800) XDC-5150. If you're outside the continental US the number is 001-928-771-2003.
- support@colesoft.com Both Dave and Frank monitor this email address.
- frank@colesoft.com You may reach Frank directly.
- dbcole@colesoft.com You may reach Dave directly.

Our office telephone number is (540) 456-8210. Ask for any of us.

Support is generally available 08:00-17:00 Eastern time, but we tend to monitor email off-hours and weekends as well. Reach out and you may be happily surprised to hear right back from us.

Attribution:

Peter Morrison created dump/XDC. He is a long term “power user” of the z/XDC product who is based in Sydney, Australia. He has been a close friend to the company over many years and is now one of our senior engineers. He has concluded a three-year project in creating dumpXDC and now faces more work as he perfects it. Peter, well done indeed, sir!

Frank Chu is well-known to our user community. Frank has learned z/OS architecture under Dave Cole’s tutelage and now contributes heavily to the internals of the z/XDC product family. He’s a vital “go-to” person in our enterprise and wears a lot of hats at Cole Software. I depend upon him a great deal. You may consider Frank as my “Imperial Truth-Sayer”, who takes exception with me whenever I’m about to tell a “lie” about our product.

Dave Cole is the Senior Architect and the original author of the z/XDC product line. z/XDC was Dave’s idea, and for over thirty years he has been the prime mover of its growth and increasing depth. He is a well-respected member of the z/OS community, and his exchanges with colleagues and customers continues to drive the enhancement of the product. He is one of the most capable programmers I’ve ever known and we at Cole Software are all in his debt. Thanks, Dave.