colesoft™

ZXDC®

RAILROAD TRACK REFERENCE

2024 Edition Including Z/XDC, c/XDC and Dump/XDC)

Bob Shimizu

ZXDC®

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 The Product may not be published, reproduced, distributed or made available to third parties for any purpose without the expressed written permission of Owner or Licensor.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.

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.

Table of Contents

Why this booklet?

This edition of the Railroad Track Reference is current to May of 2024.

"This is NOT by any means a replacement for z/XDC's very robust Online Help Subsystem." It is as correct as I could make it, but it is certainly not complete.

Many of the LIST and SET commands do not appear in this book. I've made the judgment that you won't need them most (or all) of the time. Any editorial mistakes I have made are my fault alone. I have also made the editorial decision not to discuss some of the parameters in some of the commands. If you want the FULL story please check out Dave Cole's masterful treatment in z/XDC's Online Help Subsystem. It is the reference from which this book was drawn. You can also learn a good deal about z/OS itself, because Dave HAS to explain these concepts in order for you to understand z/XDC, c/XDC and dump/XDC.

z/OS is an intricate environment to navigate. It takes years for an engineer to get traction on this platform, and it may be a decade or more before one can demonstrate mastery of only one aspect of it. z/XDC has evolved to meet your needs. It too HAS to be intricate in order to deliver its full value. This leads to astonishment for the new user, which is a bad thing. Hence, this book.

I, Bob Shimizu, am no engineer. I AM (perhaps by default) your product educator. It's my job to make the product approachable and easier to use in the early stages. Thereafter, you'll know where to concentrate your efforts in order to REALLY learn z/XDC's robust command structure. I also do my best to show you what's new in z/XDC. In a perfect world you'll see me every few years so that I can show you what's been added recently.

You may always access our engineering people by sending mail to support@colesoft.com. Please CC me (bob@colesoft.com) and I will track your problem along with you to completion. It's our job to give you the "smoothest ride" possible.

If you DON'T consult z/XDC's Online Help Subsystem you're going to miss a lot of what z/XDC offers.

Then, why provide this booklet at all? That's a fair question. I believe that this Railroad Track Reference is easier to access than z/XDC's Online Help Subsystem. I've tried to completely document what you need and only what you need.

This document has been updated to the current level of the z/XDC product: Release z2.2. Dave has decided NOT to increment version numbers of the product – at least for now. There's some internal reason for this that I'm not privy to. However, we offer multiple updates to the product several times per month, and Dave quietly introduces new features and enhancements very quietly – without fanfare.

So, if your systems people keep z/XDC updated regularly, you will get all the latest functionality that Dave and his team have added.

While I'm on the subject, if you want to find out what your maintenance level is, start a z/XDC session and issue:

LIST XDC ← This gives you a LOT of information about the current state of your copy of our product.
HELP MAINTENANCE <yyyy> (where <yyyy> is any four digit year). ← Learn what our team has done to the product for any maintenance release.

In recent years, Dave introduced c/XDC and dump/XDC, new Features for the product:

- **c/XDC** allows you to debug – with source code support - IBM XL C/C++, Metal C, SPC and (soon) Dignus' Systems/C programs in the z/OS environment. This is a major new feature within the z/XDC suite of debugging tools. Your installation may purchase c/XDC Features in any granularity to meet your organization's needs.

- **dump/XDC** allows you to use z/XDC OR c/XDC interactively in a dump. dump/XDC runs as a VERBEXIT to IPCS, which means that you get BOTH the power of XDC and IPCS when looking at a dump. You can bring in your ADATA source code to map the object code in a dump. AND you can bring your DWARF files into dump/XDC as well so that you see your C source as well. IPCS can't do that trick.

This edition of the Quick Reference has been totally stripped down to its essentials. I will no longer discuss the basic concepts of the z/XDC and c/XDC Features, because those concepts can be found in the z/XDC and c/XDC Primers. I have also taken out the archaic and vestigial commands that I feel either are underused or not used at all. For example, in these modern times there's no need for SET CONSOLE. So, many inconsequential LIST and SET commands will no longer appear.

You can thank me for this book, but the true honors go to our development team:

- Dave B. Cole is our Founder and Sr. Architect. Dave has made z/XDC his life's work. He is one of the best programmers and writers I've ever known. His hard, methodical work and unstinting devotion to quality has made my job teaching z/XDC and selling it a relatively easy one.
- Frank Chu, who has become a very knowledgeable developer during his years with us. Frank came to us years ago knowing PC security and networking. Since then he's amassed a great deal of knowledge on z/OS internals and of course z/XDC itself. Frank has done a good deal of independent design work on z/XDC while simultaneously maintaining our network and serving as our Second Level Support Representative, Systems Programmer – as well as the company chef! While writing this book I consulted with him heavily.
- Mike Lewis is the original designer of the TFS fullscreen support that brought the old DBC product into the era of full-screen operation. This was a huge contribution in itself. He has a concentration in VTAM and z/OS internals and returned to ColeSoft some years ago, where he built XDCCDF into a fully-functional Server Address Space. He is also the prime mover behind c/XDC and will be integral in newer product offerings.
- Peter Morrison has come aboard over the last couple of years. Peter is an excellent architect who has worked deep in z/OS (and predecessors) for decades. Peter has been invaluable in creating a new product we hope to release in the near term.
- Cleon Chick has come to us recently as has William Cole. These younger men are learning the product and filling in wherever they can.
- Chris Craddock has joined us on an informal basis to create a REAL GUI for z/XDC. Chris, thanks!

It is and has been an honor and a privilege to work alongside these men for over thirty years. Thanks guys!

ColeSoft's viability and continued success rests entirely on our relationships with our loyal customers, some of whom have been with us for decades. It is you - the world's highly sophisticated mainframe software engineers - that we are here to please. Some of you have even become friends, and we are enriched by that as well. The greatest thanks are due YOU. Thanks for your loyalty, your kindness and your friendship.

One thing won't change at ColeSoft or with Bob Shimizu: ***If you EVER have a question about z/XDC we will be delighted to hear from you***.

Cole Software offers full support for z/XDC. You can contact us via our web site, or at any of our telephone numbers. Both the c/XDC Primer and the z/XDC Primer are available on our documentation page, along with the "formal" documentation. We also offer free, Internet-based education classes for any level of user expertise. We look forward to hearing from you!

Contact: www.colesoft.com, info@colesoft.com, bshimizu@colesoft.com. (540) 456-8210 (Head office) (800) XDC-5150 (Bob)

Bob Shimizu
Partner, ColeSoft Marketing, Inc.
bshimizu@colesoft.com
(800) XDC 5150

How To Use This Booklet

This booklet is designed to help the z/XDC user who needs a quick reminder of the forms and uses of z/XDC commands. It is also for those who would like a desk reference on z/XDC. All the information herein is taken from the z/XDC User Guide and z/XDC Commands Reference.

Due to the highly complex nature of the z/OS environment, z/XDC is a sophisticated product rich in features, and it is recommended that the user study the aforementioned documentation for a more complete understanding of the z/XDC product. Therefore, this booklet is NOT to be considered the definitive resource on the z/XDC product.

Notational Conventions

Command names are printed **bold** in **CApital** letters to indicate the minimum abbreviation of the command name. Lower case words are parameters; information that is to be furnished with the command itself (such as an address expression).


Indicates the beginning of a command.


Indicates that the command syntax is continued on the next line.


Indicates that the command syntax is continued from the previous line.

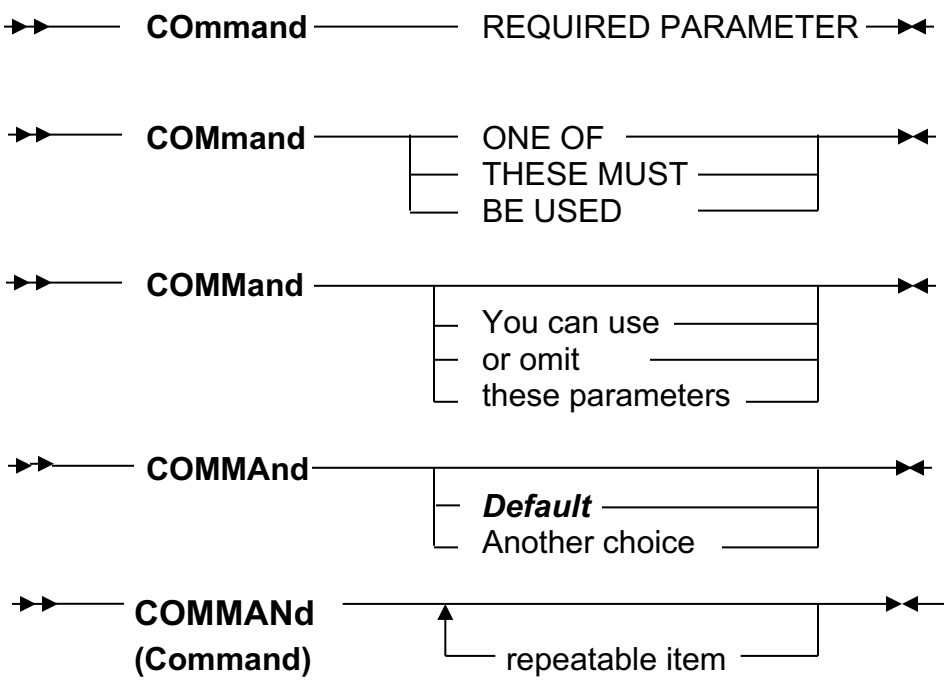

Indicates the end of a command.


Indicates that a parameter may be repeated numerous times.

- Commands and parameters that appear in upper case must be typed exactly as shown.
- Acceptable abbreviations of commands and their parameters appear in upper case.
- Default values are shown bold and italicized.
- Parameters that appear in lower case letters are optional and not required.
- Parameter groups when specified may be delimited by either a comma “,” or space “ “.
- Required parameters appear on the same horizontal line as the command.

NOTE: <u>Command names that are underlined</u> are important z/XDC commands you are likely to use in every session.

Syntax Examples



XDC Command Syntax

When issuing a z/XDC command, only as much of the command's name need be given as is necessary to make it distinguishable from other z/XDC commands. For example, OFF may be abbreviated to "O" but DMAP can only be abbreviated to "DM" (because of the DELETE and DROP commands).

In some cases ambiguous abbreviations have a "preferred" meaning. For example, F means FORMAT, not FIND (The minimum abbreviation for Find is "FI").

z/XDC command names may be preceded by blanks. Command parameters (if any) may be separated from the command name by one or more blanks. Subsequent parameters in the same command may be separated from each other by single blanks or commas interchangeably.

No single z/XDC command string may exceed 255 characters.

Multiple z/XDC commands may be typed on a single line by separating the commands with semi-colons (";"). All such commands are processed in order by XDC one after the other.

Special character strings are used to delimit a command or commands "associated" with a breakpoint definition. When the breakpoint is reached or the trace is performed, then another or many commands are processed in association with the breakpoint or trace. An associated command COULD include a "Script", which allows many commands to be performed en-masse. The syntax for associating commands with a breakpoint or trace is:

<AT/TRACE-command> 'cmd1;cmd2;cmd3' ← The command string is enclosed within single quotes, and commands are separated with semi-colons. The command delimiter ";" may be used twice to create a blank line in the resulting display, for easier reading.

If XDC encounters an error while processing a z/XDC command, processing is aborted for the remainder of the string, and no more XDC commands in that string are processed. An asterisk appears under the trailing end of the "offending" parameter, and a message appears.

You may use the Retrieve key (typically PF12) to recall the last command and fix the problem. OR, you may tab down next to the error message, enter an "H" and press the Enter key. You'll go straight into the Help Sub-system, where Dave Cole attempts to fully inform you of what went wrong, how to fix it, and offers some useful examples. The "RETRIEVE LIST" command will show you all the commands that you have entered in the current session. I really like this feature.

A Cross-reference of z/XDC's commands by Function

z/XDC's commands fall into a number of categories, and this section should help you to understand what each of them do. The categories are, "Administrative", "Environment-Examination", "Symbol Manipulation", "Traps and Traces", "Environment Alteration", "Cross-Address Space", and "Help". The very coolest, most useful commands appear bold and underlined for easy recognition. We expect that you'll use those commands in nearly every session. Consequently, not each and every z/XDC command will appear in this list – only the ones that you're most likely to need. Also, some of the commands may appear in more than one category. We don't want you to miss something useful because we only put it in one place!

Administrative commands

These commands help you to understand and configure the information that z/XDC presents to you. These commands also help you to create the environment that you prefer for different debugging scenarios. Chiefly, this section contains the SET commands, and the Profile commands.

Command	Description
XDCCALL or XDCCALLA	Establish a z/XDC session. XDCCALL begins a non-authorized session. XDCCALLA establishes an APF-authorized session.
ENd	Terminate the current, or ALL z/XDC debugging sessions.
List Help	An "index" function of the z/XDC Online Help Subsystem.
PROfile	Directly invoke the PROFILE facility's panel structure.
PROfile REAd	Summon a previously-defined profile for use in the debugging session.
PROfile RESet	Summon the system default profile for use in the debugging session.
PROfile Save	Store the current Profile for use in a later debugging session.
REAd	Cause z/XDC to process a list of pre-defined z/XDC commands; a "macro" function.
RETRIEVE LIST	Show a list of the nn previous commands you issued.
SCanlog	Search the z/XDC session log for strings of characters.
Set ACCessto	Establish z/XDC's access to a dataspace.
Set LOCks	Control system locks when z/XDC resumes a suspended SRB-mode program.
Set ILC	A default behavior. z/XDC will prevent the user from overtyping opcodes with greater widths than the existing opcode.
Set Maplibs	Establish z/XDC's access to ADATA side-files that contain source statements for z/XDC's MAP command.

Set NOIlc	Do not allow the user to alter instruction length when overtyping ALC mnemonics. Converse of Set ILc.
Set REFRprot	Directly control the way in which refreshable modules are loaded.
Set TRace	Control how z/XDC presents the working window. Also controls the scope and action of the Trace command.
Set Format	Influence how z/XDC looks at your “world”.
Set WIndow CReate/Delete	Dynamically create or delete a “watch” window on the display panel. Can be accessed via PFKey setting only.
Set Zap	Control whether the user may write to store-protected storage, or not. Default is “normal”.
TSO	Pass a TSO command through z/XDC.

Environment Examination, or “Looking Around” commands

These commands allow you to investigate the environment your program executes in. Chiefly, these are the Display, Find, Format and List commands.

Name	Description
Where	Show the current execution location, with or without ADATA in the display.
Map	Searches for ADATA or DWARF data (for c/XDC) and will overlay memory with your source code.
?	Enter the “**?**” line command in column one of any display terminal to see the list of shortcut commands that are permitted “here”. To see all of them, enter Help SHortcutcommands.
Display	Obtain a “dump” of raw memory. “**D**” is available as a shortcut command.
Down	Navigate down in the session log any distance, or number of commands.
EWhere	Format (disassemble) the memory pointed to by the Error-level PSW.
FInd	Search memory for character or hex strings. Skip areas of memory. Stop at the nth “hit”, etc.
Format/ “F”	Cause z/XDC to dis-assemble object code into instructions. “**F**” is available as a line command.
“L”	A context-sensitive “List” shortcut command. A great time-saver. Experiment with it and you’ll be glad that you did.
List ACCesslists	Show the details of any access list (PASN-AL or DU-AL).
List AMode	Interrogate whether the PSW is executing in 31-, 24- or 64-bit mode.
List Aregs	View the entirety of the Access Registers.
List ARn	View a single Access Register. z/XDC will “decode” the register for you.
List ASID	List the Address Space Identifier of an address space.
List BRanches	Display the contents of z/XDC’s Branch History Table.
List BReakpoints	Obtain a global display of breakpoints, Hooks and Trace Watches for the debugging session.
List CAChe	Interrogate z/XDC’s ADATA Cache mechanism.
List CRegs	View the entirety of the Control Registers.
List CRn	View a single Control Register. z/XDC will “decode” the register for you.
List EARegs	View the entirety of the Error-level Access Registers.
List EARn	View a single Error-level Access Register. z/XDC will “decode” the register for you.
List ENQ	Display a list of system and user Enqueues, including contention amongst enqueues.
List EPSw	Show the Error-level Program Status Word.
List EQuates	Show all currently defined (and built-in) Equates for the debugging session.

List ERegs	Show the entirety of the Error-level General Registers.
List ERn	View a single Error-level General Register. z/XDC will “decode” the register for you.
List ENQ	Display a list of system and user Enqueues, including contention amongst enqueues.
List EStaes	Display a list of currently-active STAE control blocks for any task in any accessible address space.
List FEAtures	Display hardware/software features in the current environment that z/XDC detects and may use.
List FIxed	Display memory “decoded” as a fixed point number
List FLoat	Display memory “decoded” as a floating point number.
List FRegs	Show the entirety of the Floating Point Registers.
List FRn	Show a single Floating Point Register. z/XDC will “decode” it for you.
List Help	A way to l “index” into the z/XDC Online Help Subsystem.
List HOoks	Show a list of currently defined HOoks.
List LKedmap	Display the internal structure of load modules or program objects. Like a “List Binder” command.
List LOCks	Display all locks held by the SRB-mode program you are debugging with z/XDC.
List LStack	Display individual “state” entries from the Linkage Stack. Alias: List LSEs.
List MAPlibs	Display overview information on groups of ADATA MAPLIBs defined by the user.
List MAPS	Display the list of currently defined csect and dsect maps.
List MEmoryobjects	Display information about above-the-bar memory objects in any accessible address space.
List OPErands	Displays the operands of an Assembler machine instruction OR a High Level Language statement (as in IBM C/C++/Metal C).
List PGms	Display the list of load modules in memory for any task, or locate a module via its virtual address. See also Help Commands LISt PGms Report, for a discussion on what’s displayed.
List PSw	Display the state of the Program Status Word. Try the “F” parameter. z/XDC decodes the PSW!
List RBs	Display the Request Block structure under any task in any accessible address space.
List REFrprot	Display how refreshable modules are being loaded into memory. See also the Set REFRprot command.
List Regs	Display the entirety of the General Registers
List Rn	Display a single General Register. z/XDC “decodes” it for you!
List RSa	Interrogate the Save Area chain.
LIST SSRBS	Display Suspended SRBs.
List SUbpools	Query memory allocation for the address space. Optionally, qualify by subpool ID or addr-expr.
List TAsks	Query the task structure for any accessible address space.
List TIOt	Query the task I/O table for any task in any accessible address space.
List Variables	See a variable, or variables in the C environment.
List Vstack	Display the current pool of Global, Local and subroutine variables (for c/XDC).
List XDc	Show the current state of your copy of the product. Very useful for reporting problems to us.
List XMs	Display the cross-memory characteristics of the current error-level and retry-level environments.
SWitch	When z/XDC is working on multitasking programs, switch to another sub-task.
Scanlog	After the appropriate Format command, look for text within your source listing.
Up	Navigate upward (or back in history) in the session log.
Verify	Compare the contents of memory with another memory location, or a supplied value. Handy for use with the Zap command.
Where/ “W”	Disassemble where the PSW points to now. Equivalent to “Format PSW!”, but much faster! “**W**” is available as a line command! Very cool command.

Symbol Manipulation, or “Mapping-related” commands

z/XDC is an object code debugger, which means it works just fine on OCO code. However, it’s often more convenient to see object code in the same format as originally written. z/XDC can do that, too. z/XDC also lets you view dsects, so that you can see the names of the fields, and the values in those fields.

Name	**Description**
DElete CAChe	Clear z/XDC’s cache of ADATA, which can become very large during the session.
DElete EQUates	Purge one or more currently defined Equates. The default removes all equates
DElete MAPLibs	Remove elements (or all) of the list of the maps of ADATA information that z/XDC has access to.
DElete MAPs	Remove previously defined dsect, csect, source level, and/or module maps from storage.
DMap	Load or copy a dsect into memory. See DMap Clone and DMap Load forms of the command. Also, see the Using command.
DRop	Delete the origin address for a dsect.
EQuate	Create an Equate at a specific address for a given length. Force z/XDC to interpret the memory pointed to by the Equate in various ways.
Map (or “M”)	Cause z/XDC to load a Link-Edit Map, SYM records or ADATA for the current module.csect. “**M**” is available to you as a line command!
Set Maplibs	Establish z/XDC’s access to side-files containing source statements for z/XDC’s MAP command.
“Q”	A line command that will issue a Set Qualifier command for you. A very convenient command.
Set Qualifier (or”Q”)	Establish a “default” module.csect. This allows you to reference labels as “.labelname”. “**Q**” is available as a line command.
USing	Establish an origin address for a dsect. The Autoclone parameter allows you to create lists of pointers with equate names automatically assigned to each element of the chain.

Traps, and Trace-type commands

z/XDC allows you to stop programs wherever you like (and under which conditions) in four ways. These are referred to as “traps” or “breakpoints”. z/XDC also allows you to trace program execution in seven ways. You can impose counting, or value-testing conditions on the trap or trace, and you can associate other z/XDC commands with a trap or trace (or GROUPS of commands in a z/XDC script). Finally, you can create a Trace Watch, which allows you to “fire and forget” a conditional statement that will be evaluated at every trap and trace.

Name	Description
ADeferred/ “A”	Creates a permanent “deferred” (or “scheduled”) breakpoint. Great for situations in which a module will be dynamically loaded some time in the future.
AT	Create a single, permanent breakpoint. “**A**” is available as a line command.
DElete HOOks	Remove existing Hook definitions.
GO	Release execution of the program until the next breakpoint, trace or abend is reached
HDeferred	Create a deferred HOOK. That is, create a z/XDC session wherever you like, no matter what other ESTAEs/ESPIEs may be in place. A very cool command.
HOok/ “K”	Create a z/XDC debugging session wherever you like, regardless of pre-existing ESTAEs or ESPIEs. Also available via the #XDCHOOK macro. “**K**” is available as a line command.
List Branches	Display the contents of z/XDC’s Branch History Table.
Off /”O”	Turn off a breakpoint, or various breakpoints or all breakpoints. “**O**” is available as a line command.
STep	Execute High-Level Language statements. Really important for C/C++/Metal C users!
TDeferred	Create a transient, one-time-only scheduled breakpoint.
Trace	Trace execution of the program in seven different ways. A very cool command.
Trap/ “T”	Sets a transient, one-time-only breakpoint. “**T**” is available as a line command.
T BY	Trace to the next successful branch instruction (or an EXECUTE of one).
T SB	Trace to BEFORE the next storage alteration instruction.
T SA	Trace to AFTER the next storage alteration instruction.
T BNC	Trace within the program (do not follow calls to other routines). This is fairly recent.
Trace Watch	Sets a conditional statement that is not tied to any breakpoint or trace, but is evaluated at all breakpoints and traces.
“X”	A shortcut command that “disables” or “enables” a breakpoint definition without permanently removing it.

Environment Alteration commands

Z/XDC allows you to alter registers, memory, opcodes and mnemonics, etc. with its Zap command. But it does much more than that! Basically, any change your program can do programmatically, you can do interactively.

Name	Description
COPy	Move the contents of memory about. Sort of an “MVCL” command.
DElete	Delete maps, modules, equates, hooks, maplibs, the maplib cache. Very general, very powerful.
ENd	Terminate the current, or ALL z/XDC debugging sessions.
FREEmain	Releases a chunk of storage that was previously GETMAIN’d.
GETMain	Obtains a specified amount of memory from a specific subpool.
LOad	Loads a module into storage.
“J”	A line command that forces the PSW to point to a specific instruction.
POst	Post an Event Control Block with a given code.
Read	Execute a script; a list of z/XDC commands contained in a sequential dataset. Very handy – a “macro function”!
Set PSw	Alter the Program Status Word in very interesting ways.
Switch/ “S”	In the LIst TAsks display, can be used to SWITCH to debug another task in the address space. “**S**” is available as a line command.
Verify	Compare the contents of memory with another memory location, or a supplied value. Handy for use with the Zap command.
“Z”	A line command that can be used to overtype existing values on the screen. This is huge!
Zap	Allows you to alter memory, registers, etc. Is available from the command line only.

Cross Address Space commands

z/XDC allows you to investigate other address spaces via Foreign Address Space Mode (a limited subset of z/XDC’c commands work in this environment). Z/XDC also allows you to fully control batch programs via the Cross Domain Facility (all of z/XDC’s commands work in “CDF”).

Name	Description
DISC	In Cross Domain Facility, sever the connection between the batch program and your TSO or VTAM logonid (You may in most cases prefer to use “DISC;GO”).
List LOCks	Display all locks held by the SRB-mode program you are debugging with z/XDC.
List ACCesslists	Show the details of any access list (PASN-AL or DU-AL) for tasks, SRB-routines, address spaces system-wide!
List AREGs/ List ARn	Query the Access registers, as a set or as individuals.
List DSPaces	Display a list of dataspaces owned by a specified (and accessible) address space.
Set AUth / Go Nowhere	This sequence of commands (if allowed by your security system) will change an unauthorized session in one address space to APF-authorized
Hook	Creates a new debugging session “nested” below the current one. Think of Hook as a “super-breakpoint”. See Help Hook for more information.

List ENQ	Display a list of system and user Enqueues, including contention amongst enqueues.
List MEmoryobjects	Display information about above-the-bar memory objects in any accessible address space.
List SSCt	Display the Subsystem Control Tables in the system.
List SSRrbs	Display all suspended SRBs within an address space.
Set ACCESSTO	Establish access to a dataspace.
SET ASID	If your debugging session is running APF-Authorized, you may issue SET ASID <name/number> to connect to another running address space. In this environment List, Display, Format and Hook all work. Some other z/XDC commands do not. Hook is HUGE. You can FORCE a debugging session in another address space, then go into that address space later with Cross Domain Facility. Now ALL your z/XDC commands will work.
SET ASID HOME	Returns you to your home address space. SET ASID <blank> will do the same thing.
SET LOCKS	Control locks. This pertains to z/XDC's debugging of SRB-mode code

c/XDC-Related Commands

c/XDC was engineered to look as natural as possible for a C-centric programmer. The "master reference" may be found in Help DEbugging C within the product. You may also access the c/XDC Primer on the colesoft.com website.

Here is a short list of c/XDC commands that you'll find most useful. They are listed in no particular order.

PROFILE RESET CWIDE	Change "the world" in a way that makes debugging in C as easy as possible for a C-centric programmer
LIST VARS	Display the entirety of C variables, or any single C variable
STEP	Navigate a C program, statement by statement. Step Into,Over or Out of a function call.
LIST VSTACK	Display the Stack Frame. See Global, Subroutine or local Pools of variables as they are created and discarded.
LIST SIZEOF	Find out how much storage variables consume.
Trap or AT	Set permanent, transient and deferred breakpoints in your C program
cxdchook()	Create hooks in your C program (dynamic hooks work, as well)
GO (and GO REMOVE)	Execute the C program to the next breakpoint (or hook)
LIBRARYLISTS	Create linkages between C object code and DWARF libraries (Avoid if you can. These days you almost never need it).
END	Terminate the debugging session.
LIST LKEDMAP CFRIENDLY	Examine the output of the Binder and its effects on your C program.
FORMAT	Examine your object code. If DWARF is available and AUTOMAP is in effect you will see your source code.

Please bear in mind that ALL of z/XDC is available to you in the C debugging environment. So, the more facile you are with z/OS architecture and our base product z/XDC, the more you will able to comprehend and achieve.

dump/XDC-Related Commands

dump/XDC runs as a VERBEXIT to IPCS, which gives you both IPCS AND XDC commands in the same environment. In dump/XDC you have access to ALL of z/XDC's commands EXCEPT those which alter things. You can't trace the code. You can't set breakpoints or HOOKS. You can't issue ZAP. If you think about it, this all makes intuitive sense. You're looking at "dead history".

The "master reference" may be found at Help DUmps within the product. You may also access the dump/XDC Primer on the colesoft.com website.

SET CURRENT (or the "CU" shortcut command)	dump/XDC often locates the problem spot in your dump automatically – most of the time. You can change that whenever you like.
List DSTG	See the layout of storage in the dump
List DXDC	Show the characteristics of this particular dump
List DSTG	Display storage layout within the system from which the dump was derived.
XDCMMEF	A freeware program available at colesoft.com. This extracts useful information from the system on which the dump was generated, giving you a very clear idea of the foreign system's layout (not a dump/XDC command).
DEFINE	A series of "helper" commands for working with the output of XDCMMEF.
LIST LOCKS	Useful if your dump includes Service Request Block-mode programs.
IPCS	Use "IPCS" to preface any IPCS commands you want to issue from within dump/XDC.

Help Commands

Every bit of z/XDC's documentation is included in the Online Help Subsystem. This is a "tree" structure of ISPF panels going down and out many levels. You can easily find explanation for z/XDC's messages, and commands, but you can also learn about z/OS internal structures and strategies for approaching specific debugging scenarios. Help is a very large subsystem, but this table of keywords may help you to orient yourself. This is not a complete list. Only the most "interesting" sub-topics appear here, and all of these keywords below relate only to the Help Subsystem itself, NOT this booklet.

Name	Description
LISt HELP	A handy way to "index" into the Help system. You can navigate by "levels" very easily.
Help	Access z/XDC's main Help panel. You'll see a list of referential (highlighted) and tutorial (non-highlighted) topics. You may use "S" and the Enter key to select a sub-topic. The unique name of each help panel is shown with the minimum abbreviation capitalized. You may "select" your way to a topic of interest, or you may invoke the panel name specifically. Your PFKeys change during the Help session, and these may help you to navigate around. When you're ready to leave Help, you may enter the "END" command, or merely type another z/XDC command. Either will return you to your debugging session.
"H"	A shortcut command. If you get a z/XDC message, you can enter an "H" next to the message and z/XDC will display the explanation of the message.
Help COmmands	Accesses a complete, authoritative reference on all z/XDC commands. Syntax, explanation of all parameters and examples are available here.
Help XDCSrver CDF	Gives you information on how to deploy z/XDC in the batch environment
Help SHortcutcommands	Gives you a complete list of all of z/XDC's line commands.
Help SCRIPTS	Access a library of useful "scripts" that you can use via the z/XDC "READ" command.
Help DEBUGGING	Learn how to use z/XDC in a wide variety of environments. Required reading if you want to gain full command of the z/XDC product.
Help MESSAGES	A reference on each of z/XDC's messages. Remember, you can also use the "H" line command.
Help MULTITASK	How to use z/XDC in multitasking environments.
Help WHATSNEW	A way of tracking features that are/were introduced in various z/XDC releases. It's an online "Release Guide of sorts.
Help SUPPORT	If you cannot find what you need to know in z/XDC Help Subsystem, here is how you can reach us directly. We'll gladly help you find your way!
Help MAINTENANCE <yyyy>	See what our engineering people have been doing for any given year. Can be VERY interesting.
LIST HELP online	The LIST HELP command is now available on our website. See this link.

The REALLY SHORT list of z/XDC commands

Here are the commands that you can't live without, grouped by function.

Administrative:

XDCCALL	Begin a z/XDC session.
END	Terminate a z/XDC session.
PROfile	Fiddle with your session characteristics (PFkey settings, z/XDC's default actions and MUCH MORE).
Set	Influence how z/XDC interprets your environment.

Seeing your code and local environment:

Where	Display the current executable statement or instruction within your program.
Equate	Create equates at any place in storage. Equates can map a singled address, or may span storage.
Format <addr-expr>	dis-assemble anywhere.
Display <addr-expr>	obtain a "dump" of memory wherever you like.
List Regs	See the registers. You may see ALL types of registers including control registers.
List PSw Format	Decode the Program Status Word
List TAsks	See the Task Structure in the address space.
List RBs	See the Request Block structure for any task.
List Subpools	See the organization of memory
List PGms	See all the programs running under any task.
List Variables	Examine variables in the C programming environment
List Vstack	Display the current pool of Global, Local and subroutine variables (for c/XDC).
FInd	Search memory.
Set MapLibs	Prepare to load ADATA source listings for your program (somewhat obsoleted by AUTOMAP)
Map <addr-expr>	Load maps for your program. We support ADATA and DWARF (for C).
Set Qualifier <addr-expr>	Establish a "default" module.csect.
SCanlog	After the appropriate Format command, search for text strings in your source listing.
DMap	Load dsect maps for your program (or a control block).
Using	Establish an origin address for a dsect.

Trapping and Tracing:

AT <addr-expr>	Create a permanent breakpoint.
AD <module.csect+offset>	Set a deferred breakpoint in a module that isn't loaded yet.
T	Trace a single instruction step.
T BY	Trace to the next successful branch statement.
T BNC	Trace within the current program. Do NOT follow branches into subroutines.
HOok <addr-expr>	Create a z/XDC session wherever you like! Think of Hook as the Super-breakpoint.
STEP	Follow the execution of a C program.
Off	Delete a breakpoint.
GO	Let your program execute to the next breakpoint, or abend.

Changing Things:

REAd <dsn>	Run a z/XDC "Script" of commands.
COPY	Move the contents of storage elsewhere.
Zap	Change things
GEtmain	Obtain a "chunk" of storage
LOad <module>	Load a copy of a module into storage.
POst (addr-expr>	Post an Event Control Block.
Jump	The "J" line command. Force the PSW to point wherever you like.
Set PSw	Alter the contents of the Program Status Word.

Other Address spaces:

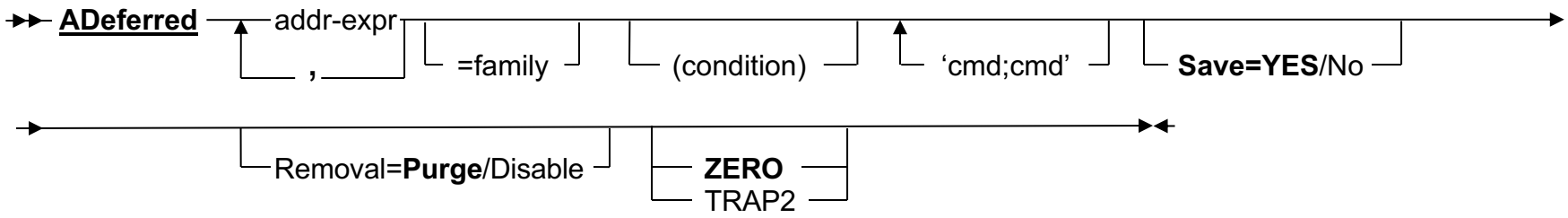
Set ASid <jobname>	Switch to another address space and look around.
Set ASid Home	Switch back to your home address space.
Cross Domain Facility	Not a command. "CDF" is how you work on a batch job. See the section!
DISC;GO	How to terminate a CDF session without your TSO session locking up.

Help System:

Help Debugging	Learn how to deploy z/XDC in different situations.
Help COmmands <name>	Research the particulars of any z/XDC command.
Help Scripts	See the READ command scripts we've provided in the product.
Help SUpport	Contact us when you get "stuck".

An alphabetic list of z/XDC's commands

Here is a pared-down list of z/XDC and c/XDC commands in alphabetic order. NOT ALL the commands appear in this book. For a complete reference on our commands, start a z/XDC session and enter Help COmmands.



Create a deferred breakpoint in the desired module (permanent for the life of the session or until manually removed).

addr-expr Address expression(s) that names a module, a module.csect and (if desired) the offset where the deferred breakpoint is to be placed. The "base" of the addr-expr MUST be the primary name of a load module. Separate multiple address expressions with commas OR spaces.

=family A breakpoint family may be specified (a series of breakpoints named alike). The breakpoint will then be assigned to that family of breakpoints.

(condition) A conditional expression may be specified to further control under which situations the breakpoint is to be taken.

'cmd;cmd' An additional z/XDC command (or commands) that are to be executed when the breakpoint is taken. New for z/XDC Release z2.1, you no longer may use colons to associate commands with traces or breakpoints. Instead you enclose the command string within single quotations and separate commands within a string with a semi-colon. See Help Whatsnew Z21 Syntaxchange! for more information.

Save= When a conditional expression or an associated command string is stated, z/XDC normally saves them for us as defaults in subsequent break-pointing commands. Omitting this parameter defaults the action to Save=Yes. Save=No turns this off.

Removal= Default action is "Purge". This would certainly apply to transient breakpoints (which are automatically deleted when executed once). "Disable", on the other hand would deactivate the breakpoint (changing the op-from X'00' or X'01FF' back to its original value). This retains the breakpoint, and in the LIST BREAK command's output you will see that the breakpoint is "DISABLED". You can enable or disable a breakpoint using the "X" shortcut command next to any breakpoint. PURGE will cause the breakpoint to be automatically deleted when it is executed. In this case it would be just as easy to create a TRANSIENT breakpoint.

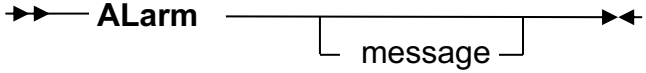
ZERO This will create a "classic" breakpoint of X'00'. It will cause a S0C1 abend as z/XDC has done for decades.

TRAP2 This invokes z/XDC's TRAP2 support, in which case z/XDC will create a X'01FF' at the breakpoint and will use the TRAP2 facility to create breakpoints.
Note: for more information on z/XDC's TRAP2 support, enter "Help Whatsnew z22 Trap2".

Remarks: This command creates a breakpoint in a module that will be loaded into memory sometime in the future. This is a "scheduled" breakpoint. A breakpoint created by ADeferred remains in existence for the life of the debugging session or until it is explicitly removed. If you desire a one-time-only deferred break-point, see TDEFERRED.
This command may not be used in Foreign Address Space Mode.

Examples:
AD MYPGM+3DC ← Create a Deferred breakpoint at offset "+3DC" into module MYPGM whenever it loads.
AD MYPGM.MYCSECT+3DC ← Create a Deferred breakpoint at offset "+3DC" into module MYPGM and csect MYCSECT whenever it loads. You would have had to issue a DMAP command for MYPGM prior to this so that z/XDC can "find"csect MYCSECT.
AD =5 MYPROG+3DC +4 +8 (R1,EQ,00) :L REGS:L AREGS ← OK, this is a toughie. Set a Deferred breakpoint into Family "5" at MYPROG+3E8 but only if Register one is equal to all zeroes. If and when this Transient Deferred breakpoint fires, then issue "LIST REGS and LIST AREGS.

More Info: Enter Help COmmands ADeferred. For conditions see HELP Breakpoints Conditional and Help Commands Syntax Breakpoints Conditions.



Ring the terminal bell for up to 5 seconds, optionally displaying a text message.

Message Any arbitrary text up to 68 characters (I actually counted them).

Remarks: Alarm will display a “Flower Box” of asterisks, containing the optional message text, and prompting you to press the Attention Key to stop the alarm.

For decades XDC had NO Alarm command. Using it would generate DBC007E COMMAND NOT RECOGNIZED causing the terminal bell to ring anyway (cute). I guess Dave decided to formalize it.

Examples: AL ← You see the Flower Box with the prompt to end the alarm with your Attention Key.
AL what a fun command! ← Displays the Flower Box with “what a fun command!”
ALARM Gawd, you’re ugly! ← generates an error because the single quotes aren’t balanced.

More Info: Enter Help COmmands ALarm

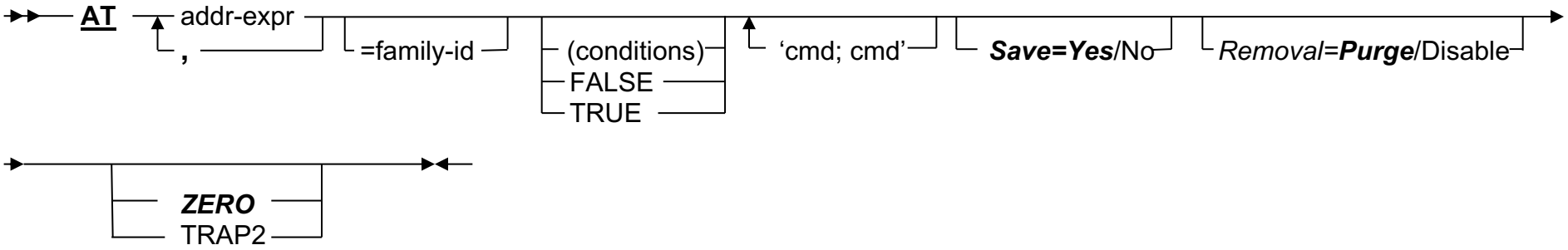
Introduce a single line of comment into the z/XDC session log.

Comment text Any amount of text that will fit into the command line

Remarks: Put an asterisk into the command line and type whatever you care to. If you're the sort who preserves z/XDC session logs for historical purposes, this may be useful. If you want to create a BLOCK of commentary, please use the COMMENTARY command.

Examples: * Bob, when you review this session log, remember to bring home some milk.
* I can't think of anything else to put in here.

More Info: Enter Help Commands ASterisk. See also Help Commands COMMENTAary.



Set a permanent breakpoint at a specified location in virtual memory in order to stop program execution at that location.
Short-cut equivalent: "A"

addr-expr	Specify the location where you wish to place the breakpoint. You may specify multiple address expressions separating them with a comma OR a space. Multiple breakpoints created at the same time will be assigned to the same family (a numeric value).
=family-id	Assign this breakpoint to a particular family. This parm may appear anywhere in the command string, prefaced with the equal sign ("**=**") and a family number ranging from 1-999.
(conditions)	Specify a Counting or Value-Testing conditional statement (See Conditional Expressions in this booklet or type Help COmmands Syntax Breakpoints CONditions). **FALSE** will bypass the trap, or cause the Trace to advance one instruction. **TRUE** will cause the trap to be accepted or the Trace operation will stop. FALSE/TRUE were introduced in order to cause a TRACE WATCH to stop program execution – which it couldn't do originally.
'cmd'	Specify other z/XDC commands that you would like to have executed WHEN the breakpoint is reached. Enclose the command string within single quotations. Separate individual commands within a string using a semi-colon.
Save=	When a conditional expression or an associated command string is stated, z/XDC normally saves them for us as defaults in subsequent breakpointing commands. Omitting this parameter defaults the action to Save=Yes. Save=No turns this off.
Removal=	Default action is "Purge". This would certainly apply to transient breakpoints (which are automatically deleted when executed once). "Disable", on the other hand would deactivate the breakpoint (changing the op-from X'00' or X'01FF' back to its original value). This retains the breakpoint, and in the LIST BREAK command's output you will see that the breakpoint is "DISABLED". You can enable or disable a breakpoint using the "X" shortcut command next to any breakpoint. PURGE will cause the breakpoint to be automatically deleted when it is executed. In this case it would be just as easy to create a TRANSIENT breakpoint.
ZERO	This will create a "classic" breakpoint of X'00'. It will cause a S0C1 abend as z/XDC has done for decades.
TRAP2	This invokes z/XDC's TRAP2 support, in which case z/XDC will create a X'01FF' at the breakpoint and will use the TRAP2 facility to create breakpoints. Note: for more information on z/XDC's TRAP2 support, enter "Help Whatsnew z22 Trap2".

Remarks: This is one of z/XDC's core commands; one that you will use in almost every session. TRAP is an alias of this command. You can also specify "ATX" to create a "hard-to-remove" breakpoint (that must be OFF'd explicitly). This command may not be used in Foreign Address Space Mode.

Examples: **AT R14? +4 + 6 .JOBTOP1 +2:LREGS:F R15?** ←Set six breakpoints, then execute a List Registers and a Format at the location pointed to by the contents of R15 whenever the breakpoint(s) is/are taken.

AT myprogram.mycsect+23C ← Set a single breakpoint at offset 23C into the csect "mycsect" of the module "myprogram".

AT .loopstart (R1,NE,7FFFFFFF):ALARM WHOA! IT HAPPENED! ← Set a breakpoint at a label in the program that will only stop execution if the contents of Register One are no longer equal to X'7FFFFFFF' Then, automatically create an alarm condition to alert the programmer.

AT R5+10 (R3+3,AND,02,EQ,02) Set a breakpoint that will not be accepted until the 31st bit (i.e., bit number 30) of R3 is on.

More Info: Enter Help COmmands AT. For conditions see HELP Breakpoints Conditional and Help Commands Syntax Breakpoints Conditions.

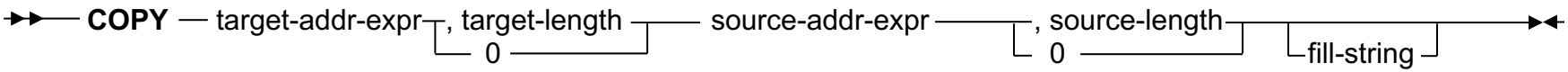
COMmentary

Open a “scratch pad” in the session log into which you may enter any text you like.

This command accepts no operands.

Remarks: COMmentary might be useful if you wish to include some thoughts during your debugging session that will appear in the session log. You have up to 50 lines of text to input. To save your text, press PFkey3 or PFKey5. To abandon the panel, press PFKey4.

More Info: Enter Help COmmands COMMENTAry.



Propagate the contents of memory to one location from another (with optional filling).

target-addr-expr Where the user wishes to copy the source memory contents to. Can also be a retry level register, a general register, an Access Register or a Floating point register.

target-length Length of the target location. Can be a constant, a hexadecimal number, a decimal number (when suffixed by "N"), the contents of a register, an arithmetic expression, the contents of a location in storage or the resolved value of an address expression.

source-addr-expr Where the user wishes to copy from. If source-addr-expr is omitted, then source-length must also be omitted.

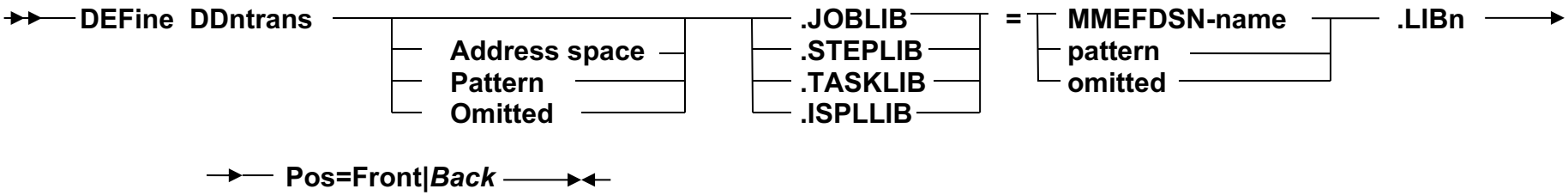
source-length Length of the source location. Can be a constant, a hexadecimal number, a decimal number (when suffixed by "N"), the contents of a register, an arithmetic expression, the contents of a location in storage or the resolved value of an address expression. If source-addr-expr and source-length is omitted, then target-length will be used as the default source-length.

fill-string An optional padding string of up to 256 characters. Character data is prefaced by a single quote. If omitted, then X'00' will be used as the default fill-string.

Remarks: If the target length is shorter than the source length, then the source is truncated. If the source-length is shorter than the target-length, then padding from a fill string is copied to the target location immediately following the data copied from the source. A source length of zero causes the target location to be filled entirely with padding from the fill string.
This command may not be used in Foreign Address Space Mode.

Examples: COPY R12! 300N mymod.mycsect.buffer 300N ← Copies 300 decimal bytes of data in a buffer to the memory pointed to (in the existing address mode, of 24 or 31-bit memory) by general register 12.
COPY R4?~ASID(pasn) 200N R3?~ALET(AR5) 200N ← Copies 200 (decimal) bytes from the location pointed to by general register 3 in the address/data space indicated by access register 5. The data is copied to the location pointed to by general register 4 in the retry level Primary Address Space.

More Info: Enter Help COmmands COPy.



Tell d/XDC where to find load module information within the output of the XDCMMEF utility for later use of the MAP command.

Address space	This parameter is optional. If patterned (such as with an asterisk) or omitted entirely, it applies to any address space in the dump.
.JOBLIB	This will relate the JOBLIB to a particular LIBn definition within XDCMMEF's output.
.STEPLIB	This will relate the STEPLIB to a particular LIBn definition within XDCMMEF's output.
.TASKLIB	This will relate the TASKLIB to a particular LIBn definition within XDCMMEF's output.
.ISPLLIB	This will relate the ISPLLIB to a particular LIBn definition within XDCMMEF's output.
MMEFDSN-name	The dataset name which contains the output of the XDCMMEF utililty.
pattern	Can be an asterisk ("*") to indicate any library ddname, or minus ("-") or "NONE" to indicate no MMEFDSN name. This will cause d/XDC to search ALL MMEFDSNs to find the matching LIB entry. Note that the minus sign has meaning ONLY within the DEFINE DDNTRANS command, and applies nowhere else in z/XDC.
LIBn	XDCMMEF assigns the names "LIBn" to each JOBLIB, STEPLIB, TASKLIB and ISPLLIB in XDCMMEF's output. "n" is an integer ranging from 1 to {I have no clue how many digits are supported.}
Pos=	If FRONT is specified, this leads the search order. If Back is specified, this is last in the search order. Back is the default.

Remarks: This command is optional. Normally, if d/XDC can find Job File Control Blocks (JFCBs) in the dump, then it will be able to find load module information in the output of the XDCMMEF file without intervention. However, if JFCBs are missing from the dump you may to use this command to set up a "linkage" between the dump and the output of the XDCMMEF utility.

You may issue as many DEFINE commands as are necessary. BUT, since DEFine is a command within the z/XDC product you CAN create a script to automate any z/XDC command, including DEFine.

Examples: DEFine DDNTRANS BOB.ISPLLIB=CUST.DUMP.LIB3 ← In the address space BOB, the ISPLLIB in the CUST.DUMP XDCMMEF file is assigned to "LIB3". Because the POS parameter is omitted, it will be placed at the back of the search queue.
DEFine DDNTRANS *=LIB2 POS=FRONT ← For any task library DDNAME ("*"), regardless of address space (because it is omitted), d/XDC will read every eligible XDCMMEF output file and the first instance of "LIB2" will be used. It will also be placed at the front of the search queue.

More Info: Enter Help COmmands DEFine. See Help COmmands DEFine DDNTRANS. See also Help Debugging Dumps Datasets

Tell d/XDC where to find dsect information within the output of the XDCMMEF utility for later use by the DMAP command.

<dsn> The name of the XDCMMEF output dataset to be defined to d/XDC. It must be catalogued, DSORG=PO (a PDS or PDSE) and RECFM=U (for a load library) or RECFM=VB (for an ADATA side-file).
The system described by the <dsn> must match the dump being worked upon, as follows:

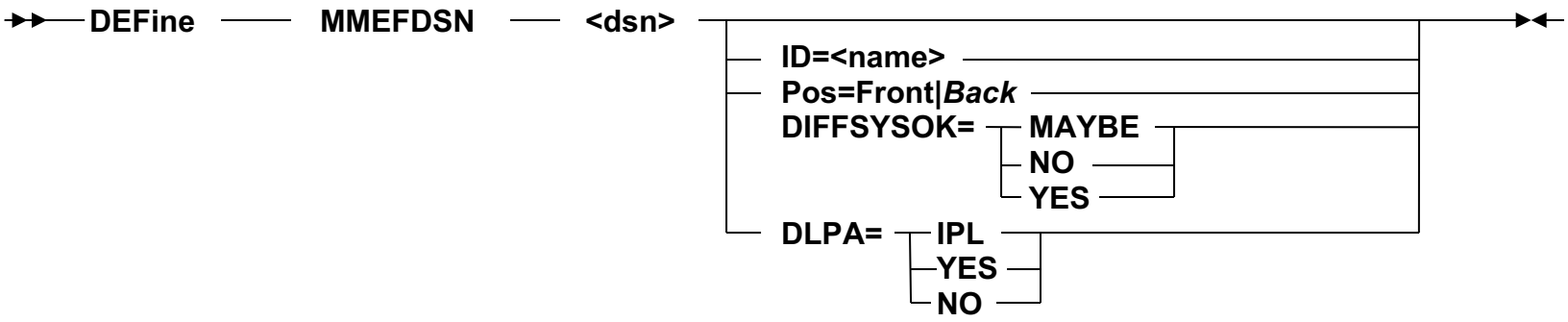
- CPUID (CPU ID and serial number;
- The Operating System name (normally, "z/OS");
- The OS version, release and mod level;
- If the dataset describes the system nucleus, the nucleus' start and ending addresses.

Pos= If FRONT is specified, this leads the search order. Back is the default position in the search order.

Remarks: When d/XDC is being used on a local system, d/XDC can determine the location of dsect libraries. However,in the case of a dump from an outside system (a customer site), d/XDC cannot make this determination. In this case the d/XDC user (the vendor) can ask the dump transmitter (the customer site) to download and run the XDCMMEF module map extraction Facility. Then one or more DEFINE DSECTLIB commands will direct d/XDC to look for dsect libraries in the XDCMMEF report sent to the d/XDC user. You may issue as many DEFINE commands as are necessary. BUT, since DEFine is a command within the z/XDC product you CAN create a script to automate any z/XDC command, including DEFine.

Examples: DEFine Dsectlib DDEFS.ZOS.V1R13.SYM ← This will add this dataset to the list of load module information datasets. It will appear at the back of the search order.

More Info: Enter Help COmmands DEFine. See Help COmmands DEFine Dsectlib. See also Help COmmands LISt DSEctlibs. See also Help Dumps MAppingdata.



Tell d/XDC where to find load module information within the output of the XDCMMEF utility.

<dsn> The name of the XDCMMEF dataset to be defined to d/XDC. It must be catalogued, DSORG=PS (sequential) and RECFM=VB (Variable Blocked).
The system described by the <dsn> must match the dump being worked upon, as follows:

- CPUID (CPU ID and serial number;
- The Operating System name (normally, "z/OS");
- The OS version, release and mod level;
- If the dataset describes the system nucleus, the nucleus' start and ending addresses.

ID= Allows you to assign an arbitrary name of up to 8 characters. You may later use this name in a subsequent DEFINE DDNTRANS command.

DIFFSYSOK Whether the MMEFDSN will be accepted if a system mismatch is detected ("Different System OK"). First, z/XDC displays message DBC793 lets you know a mismatch was detected, and prompts you to answer YES or NO. The MAYBE parameter works off your prompt in subsequent processing and may (or may not) use the MMEF dataset in d/XDC. NO tells d/XDC NOT to use the MMEF dataset if a mismatch is detected. YES tells dump/XDC to over-ride any error message and use the MMEF dataset anyway.

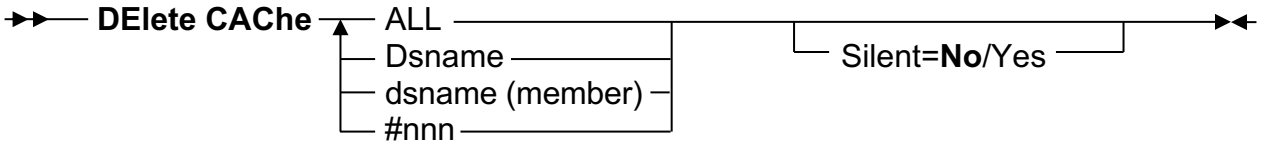
DLPA= Controls whether Dynamic Link Pack Area information in the MMEF dataset will be used. IPL tells dump/XDC to ignore DLPA information in the MMEF dataset if a mismatch is detected between the IPL time in the dump and the IPL time in the MMEF dataset (they SHOULD be equal). YES means that d/XDC will use the DLPA information in the MMEF dataset. NO means that DLPA information in the MMEF dataset will be ignored.

Pos= If FRONT is specified, this leads the search order. The default order is at the back of the queue.

Remarks: When d/XDC is being used on a local system, d/XDC can determine the location of load modules. However in the case of a dump from an outside system (a customer site), d/XDC cannot make this determination. In this case the d/XDC user (the vendor) can ask the dump transmitter (the customer site) to download and run the XDCMMEF Module Mapping Extraction Facility. The d/XDC user may then issue the DEFINE command to d/XDC. You may issue as many DEFINE commands as are necessary. BUT, since DEFine is a command within the z/XDC product you CAN create a script to automate any z/XDC command, including DEFine.

Examples: DEFine MMEFDSN DUMPS.CUST21.D052.LMINFO ID=FRANKCHU ← This will add this dataset to the list of load module information datasets. It may be referred to later by the name "FRANKCHU".
DEFine MMEFDSN DUMPS.CUST93.D052.LMINFO POS=FRONT ← This will add this dataset to the list of load module information datasets. It will reside at the beginning of the search order.

More Info: Enter Help COmmands DEFine. See Help COmmands DEFine MMEFDSN. See Help COmmands LISt MMEFDSN. See also Help Dumps Usageinfo Datasets.



Purge ADATA from cached memory to prevent memory shortage conditions.

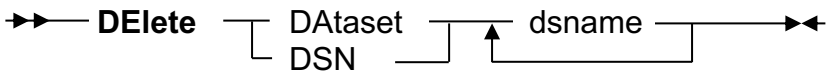
ALL	All cached ADATA files will be purged from memory.
dsname	Identify which cached file to remove.
dsname(member)	Equivalent to the above. A further way of qualifying what the user wishes to purge.
#nnn	A decimal number that identifies the cached member. The user can obtain this number from the output of the LIst CAChe command.
Silent=Yes/No	Suppresses (or does not suppress) progress and completion messages. This is intended mostly to suppress z/XDC's "internally issued DELETE EQUATE commands.

Remarks: When z/XDC reads ADATA, it caches an extraction of the data into memory in case it might be needed again. Because ADATA takes up a great deal of space, the DElete CAChe command is provided to allow the user to control a potential shortage of memory.

Note: Cole Software provides an "Adata Filter" that can minimize the storage required for the use of z/XDC's source statement support. See Help Maps AData Excessadata Xdcadata for more information.

Examples: DE CAC ALL ← All ADATA is removed from memory.
DE CAC #1 #3 ← Assuming that the LIst CAChe command had been entered earlier, this will remove the first and third elements of cached ADATA.

More Info: Enter Help COmmands DElete Cache. See also LIst CAChe.



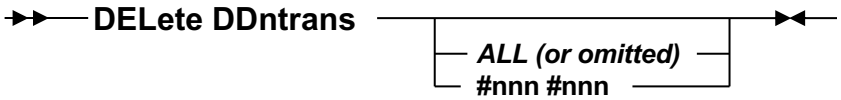
Delete a dataset from within a z/XDC session.

dsname The name of a fully-qualified cataloged dataset. The dataset name must contain a period. You may delete multiple datasets at once (so long as it all fits on the command line). A member of a PDS or PDSE cannot be deleted (you can do that from within TSO instead – IF the TSO environment is active).

Remarks: You can make a dataset go “bye-bye” if you want to. This command is subject to system security.

Examples: DE DSN ABC.SCRIPT.OUTPUT - Providing that the dataset exits, can be allocated with DISP=OLD and the user has the authority to delete it, the dataset will cease to exist.

More Info: Enter Help COmmands DELete DAtaset.



Delete a previously DEFINE'd DDNTRANS token – or the entirety of the list.

ALL (or omitted All DDNTRANs previously DEFINED are deleted.

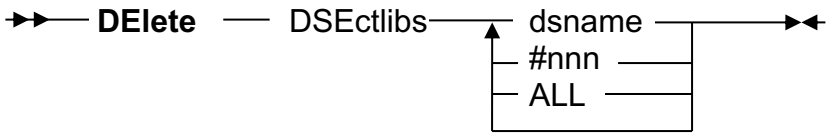
#nnn Only a particular DDNTRANS token is deleted.

#nnn #nnn Delete more than a single DDNTRANS token. You may be able to specify more than two tokens (I don't have a way to test this).

Remarks: This is only for use with dump/XDC. If the output of XDCMMEF was made available to the dump/XDC session and if dump/XDC was unable to find maps for load modules in the dump, AND IF the user then issued one or more DEFINE DDNTRANS tokens – AND IF the user later wishes to delete some of the tokens created by the DEFINE DDNTRANS command, then this will delete them.

Examples: DEL DD ← deletes all DDNTRANS tokens
DEL DD #2 #5 ← tokens #2 and #5 are deleted from the queue of DDNTRANS tokens.

More Info: Enter Help Commands DELete DDntrans. See also Helo Commands DEFine DDntrans.



Delete DSECTLIB definitions, making them unavailable to z/XDC.

dsname The name of a fully-qualified cataloged dataset. You may delete multiple DSECT libraries at once (so long as it all fits on the command line).

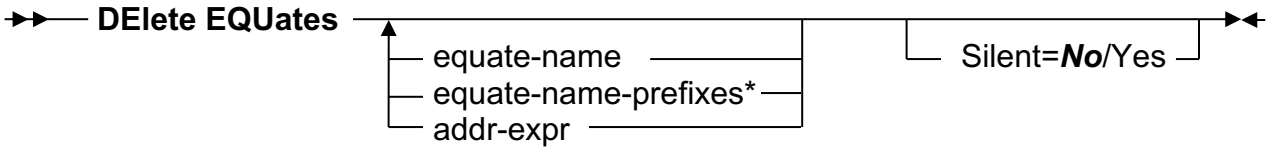
#nnn A decimal number reflecting the DSECTLIB's sequence as displayed in a prior LIST DSECTLIBS command.

ALL Deletes all DSECTLIB definitions. One of z/XDC's great advantages is showing you the field names and contents of your DSECTs. Why would you want to delete them?

Remarks: This command does not delete the actual DSECTLIB dataset. It only removes the DSECTLIB definition from z/XDC's purview.

Examples: DEL DSE FRANK.TEST.LOADPO Frank has no more need for this DSECTLIB.

More Info: Enter Help COmmands DELete DAtaset.

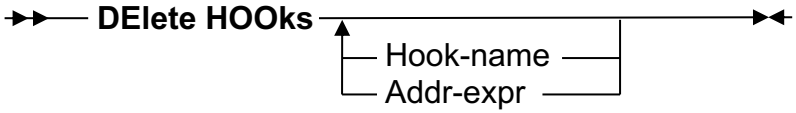


Remove the definition of a previously-defined equate.

omitted	All equates are removed.
equate-name	The name used when the equate was defined. Example: JACK, POINTER, etc.
equate-name-prefixes*	The first few characters of a group of equates followed by an asterisk (“*”) Example: RB#*.
addr-expr	The address-expression of the equate.
Silent=*No*/Yes	Suppresses (or does not suppress) progress and completion messages.

Examples: DE EQU TCB#* ← This command deletes all equates whose names begin with the characters “TCB#”.
DE EQU FRED ← The equate FRED is removed.
DE EQUATES RB#* CTCB ← The equates with names that begin with “RB#” are deleted along with the equate named “CTCB”.

More Info: Enter Help COmmands DElete Equates.



Remove a Hook definition.
Short-cut equivalent: “O”

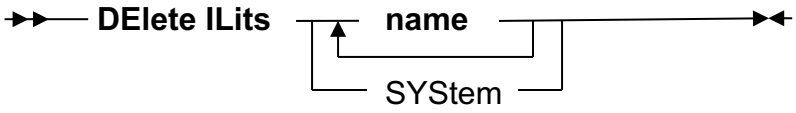
omitted	All Hook definitions are removed.
Hook-name	The specific name of a Hook. Issue LIST HOOKS to see all extant Hooks.
addr-expr	The Hook at the specified address is removed.

Remarks: The Hook command is a facility whereby the user can apprehend a program on an ad-hoc basis and begin a debugging session nearly anywhere. HOOK works by creating a z/XDC environment wherever the user wishes. Each Hook receives a unique name. The LISt HOoks and DElete HOOks commands are provided as well. The “O” command may be used to “OFF” a Hook as well as a breakpoint.

Note: With appropriate security clearance, it is possible to set a Hook in Common Storage. Also, Hooks can be “seen” only by the debugging session that creates them. So, if a user leaves a Hook in Common Storage, then that Hook will remain active and will cause a z/XDC session to be created the next time the code containing the Hook receives control. BE CAREFUL when setting Hooks in Common Storage.

Examples: DEL HOO ← All defined Hooks are removed
DE HOO mymod.mycsect.readloop ← Removes any Hook at this address in memory.

More Info: Enter Help COmmands DElete Hooks. See also Help COmmands Hook.



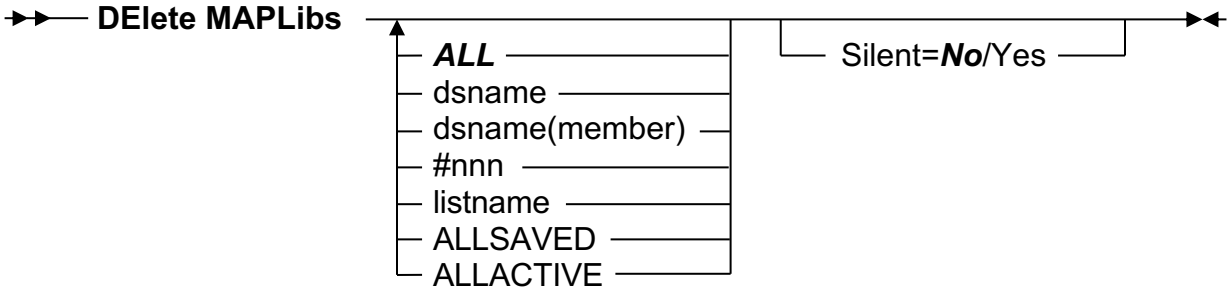
Delete an IPCS literal

name Any IPCS literal name. You may specify as many literal names as will fit on the command line (up to 255 characters).
Hook-name Deletes only dump/XDC-specific literals (their names begin with “DXDCS”).

Remarks: *This command is specific to dump/XDC only*. However, you can always delete literals with the IPCS DROPSYM command.

Examples: DEL IL MYL1 MYL2 ← The IPCS literals “MYL1 and MYL2 are removed.

More Info: Enter Help COmmands DELete ILits



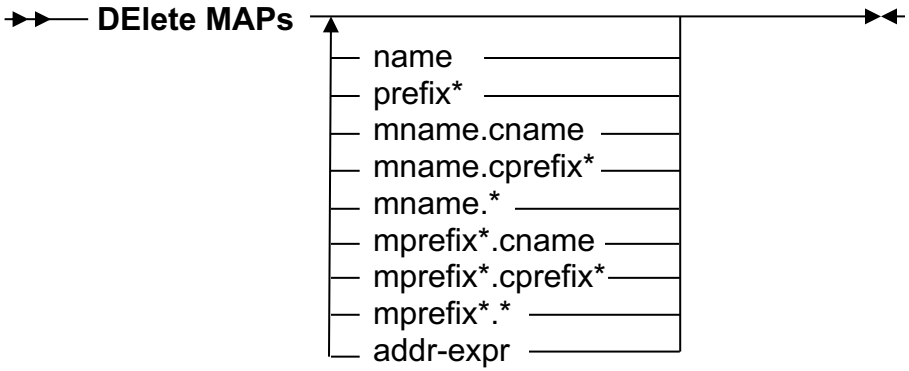
Remove elements (or all) of the list of the maps of ADATA information that z/XDC has access to.

ALL	All ADATA information will be purged. This is the default.
dsname	The name of a map to be deleted.
Dsname(member)	A further qualification of the maplib to be deleted.
#nnn	You may specified numbered elements (referenced from the output of a previously issued List MAPlibs command).
Listname	Purges the named MAPLIB list from the users's session profile. The user must issue a PROFILE SAVE command to make this permanent.
ALLSAVED	Purges ALL MAPLIB lists from the user's session profile.
ALLACTIVE	Purges all entries from the ACTIVE MAPLIB list. This is equivalent to the SET MAPLIBS RESET command.
Silent=*No*/Yes	Suppresses (or does not suppress) progress and completion messages.

Remarks: z/XDC has the ability to display source statements in Assembler and C languages. While current releases of z/XDC can automatically issue the MAP command for the user to obtain source statement information, sometimes one has to specify where z/XDC can find source libraries.

In the Assembler language environment, one may issue a "SET MAPLIBS <dsn>" command to tell z/XDC where to find ADATA in source statement libraries. In Cole Software's "world" if you can create a thing you can both list and delete it, hence the purpose of this command.

More Info: Enter Help COmmands DElete MAPLibs. Review the examples THERE.



Remove previously-defined dsect, csect, source level, and/or module maps from storage.

omitted If omitted, then ALL maps defined are deleted for all address spaces.

name The pure name of the csect or dsect map to be deleted.

prefix* A pure name suffixed with an asterisk. Deletes all maps with the prefix name.

mname.cname The name of a module and csect map. Wildcarding is allowed via the asterisk character as depicted above.

mname.cprefix* The name of a module and csect with an asterisk. Deletes all maps with the given cname as a prefix.

mname.* The name of a module suffixed with an asterisk. Deletes all modules with the prefixed module name.

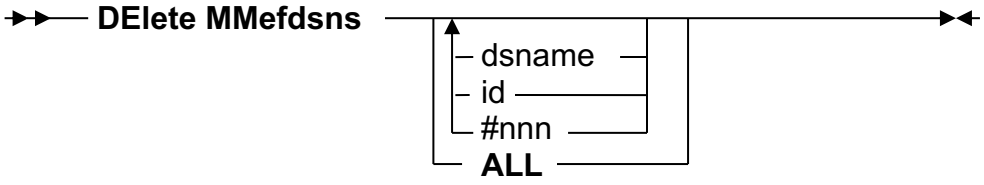
mprefix*.cname/mprefix*.cprefix*/mprefix* Similar to the above parameters except that the action is performed for csect maps that are within ALL modules whose names match the given mprefix.

addr-expr Deletes all maps that contain the address-expression.

Remarks: z/XDC makes heavy use of maps in order enhance object code displays for human readability. This command removes maps that one no longer cares to use.

Examples: DE MAPS EDIT.IKJEBIN1 ← This deletes the csect map for IKJEBIN1

More Info: Enter Help COmmands DElete MAps. See also Help Maps.



Remove previously-defined XDCMMEF dataset names.

dsname State the name of an XDCMMEF dataset name. Multiple dsnames may be deleted at once (up to the 255 character limit on our command line).

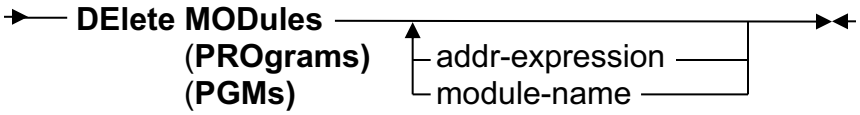
id If a user issues the DEFINE MMEFDSN command, he or she may assign an arbitrary 8-character name to that MMEF dataset. One may also DELETE an MMEFDSN by using this name.

#nnn When the LIST MMEFDSNS command is issued, dump/XDC creates a numbered list of MMEF datasets. You may specify a decimal number corresponding to the desired MMEF dataset.

ALL Deletes all MMEFDSN definitions to be deleted from dump/XDC's "awareness". I can't tell you if this is a default action because I don't have a test-bed for this.

Remarks: This command is specific to dump/XDC. When a dump is received from a "foreign" system (such as at a customer's site), the customer has the option to run our XDCMMEF tool, which collects the "boundaries" of modules, csects and system libraries. The output of XDCMMEF can be "folded into" a dump for use by dump/XDC. This command removes XDCMMEF dataset names from dump/XDC's awareness. It does not delete the files themselves.

Examples: DEL MMEFDSNS XXXX.TEST.OFCX1 ← This removes the MMEF dataset XXXX.TEST.OFCX1 from dump/XDC's awareness.
More Info: Enter Help COmmands DElete MMefdsns. See also DEFINE MMEfdsns and LISt MMefdsns.



Remove a previously loaded module from memory. Aliases for MODule are PROgrams or PGMs.

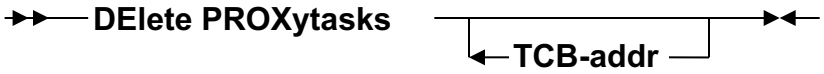
addr-expression An address expression that resolves to some location within the module to be deleted.
module-name The name of the load module(s) to be removed from memory.

Remarks: This command works ONLY for modules that were previously loaded into memory via the LOAD macro under the current Task Control Block. If multiple copies of a module exist in memory, the use count is decremented each time this command is run. If the use count for a module reaches 0, then the system removes that module from storage.

In practice, one doesn't need to specify the parameters "MODules", "PROgrams", or "PGMS". z/XDC will take the module-name and/or addr-expression without them.

Examples: DE PGMS MYPROG ← Deletes the module MYPROG from memory.
DE FARFEL+0 ← The module "FARFEL" is deleted from memory.

More Info: Enter Help COmmands DElete MOdules.



Delete Proxy Tasks that z/XDC may use in debugging Service Request Block-mode programs.

TCB-expr An address expression that resolves to the location of a Task Control Block that will anchor the Proxy tasks. If you issue "LIST TASKS" you may subsequently use the TCB#nn equates that are defined by List Tasks as equates for TCB-addr. You may specify multiple TCB-addr values in the same command string.

Remarks: What are "Proxy Tasks"? I'm glad you asked. Proxy Tasks are empty Task Control Blocks created by XDC so that IF one is debugging Service Request Block code, XDC has somewhere to "anchor" the SRB debugging session.

When debugging Service Request Block-mode programs, z/XDC runs as a Functional Recovery Routine. When you debug FRR code, z/XDC runs in two "phases":

i. The "local" phase is the address space where the SRB-mode code runs.
ii. The "remote" phase is the Home address space where z/XDC can attach a Task to run the debugging session.

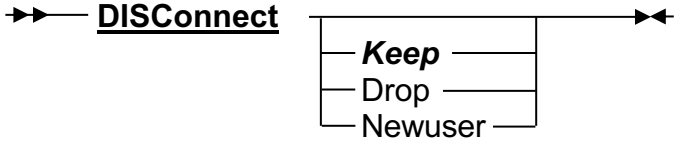
When a breakpoint or abend is sensed in the SRB-mode program, z/XDC suspends the SRB, deletes locks, and communicates with debugging address space to request debugging services. It looks for any suitable task in the remote phase and proxy tasks are tailor-made for the purpose. When you issue a "GO" or resume the SRB, locks are set, and the SRB resumes in its normal environment.

DElete Proxytasks allows you to discard unwanted Proxy Tasks. These tasks may be in any accessible address space.

There is no "LIST PROXYTASKS" command. Instead, issue List TAsks, and you'll see all tasks, including Proxy Tasks.

Examples: DE PROX TCB #6 ← A Proxy Tasks will be POST'd for removal. If you have issued List Tasks, and are using the TCB#nn equate names as parameters for this command, then you must capitalize their names precisely.
DE PROX TCB#6 TCB#7 007B11978 ← Two Proxy Tasks are deleted by their equate names. One is deleted using a pure virtual address.

More Info: Enter Help COmmands DELete PROXytasks. See also Help DEbugging Frr.



Sever the connection between a batch debugging session and a TSO or VTAM user without terminating the session.

Keep Default. (same as omitting any parm). The user's terminal is disconnected from the session but is returned to the Cross Domain Facility (XDC-CDF) job selection panel. The user may then select another job to debug or issue the "END" command to return to the z/XDC Startup Panel.

Drop The user's terminal is disconnected from both the debugging session and from XDC-CDF entirely.

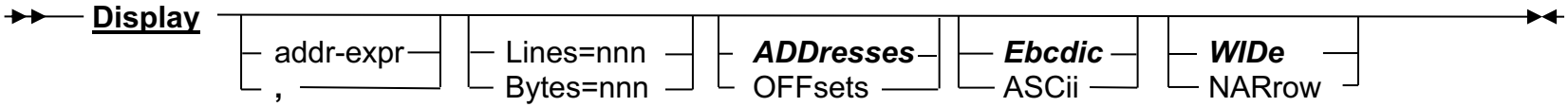
Newuser The user's terminal is disconnected from the debugging session. XDC-CDF then displays a logon panel from which a user may log back into XDC-CDF using a different TSO userid.

Remarks: If the user has connected to XDC-CDF from a TSO session, using the Newuser parameter does not affect the "old" TSO session.

Note: If you want to let the batch program run, AND, you want to disconnect from it, enter "**DISC;GO**". This will sever the connection, allowing you to go your separate ways. HOWEVER, A DISConnected batch job may persist even if "DISC;GO" has been issued. The best practice is to cancel the batch job if you really want things to "end and go away entirely".

Examples: DISC ← The user's terminal will be disconnected from the session, and will return to the XDC-CDF job selection panel.
DISC D ← The user's terminal will be disconnected from the session, and will leave the XDC-CDF environment completely.

More Info: Enter Help COmmands DISConnect. See also Help Debugging Batchjobs.



Produce a "dump" of virtual memory.
Short-cut equivalent: "D"

addr-expr	The location in memory where the dump is to begin. If addr-expr is omitted, then the next screen-full of memory is displayed. If addr-expr is omitted and other parameters are desired, then a comma must be used before stating the other parameters.
Lines=	The number of lines the user wishes to see. If omitted, z/XDC will fill the current window. LINES= will generate from 0-10,000 lines of display.
Bytes=	Will display either a number of bytes given in hexadecimal, or a decimal number, if suffixed with an "N" (example: BYTES=200N).
ADDresses	The virtual address of each line of the display will appear on the left side of the panel.
Offsets	The offset from the start of the data's object (load module, csect or dsect) will appear on the left side of the panel.
***Ebcdic*/Ascii**	Memory will be interpreted in EBCDIC format (default) or ASCIIi format.
***WIDe*/NARrow**	The display command will show 32 bytes of storage OR 16 bytes of storage per line.
omitted	If no parameters are given, z/XDC will display the NEXT LOGICAL screen's worth of memory – like a "scrolling" function.

Remarks:

This is one of the core commands in z/XDC. Display presents a "dump-like" display of raw storage.

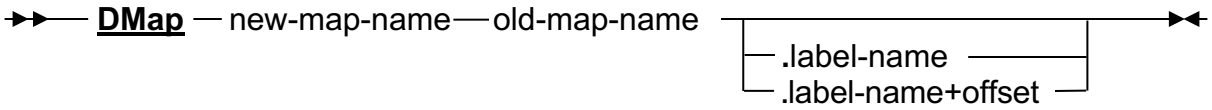
There is one field you'll see in Display that you won't see in a true dump: z/XC shows you the key structure of memory you're looking at.. "8F" would mean "Key-8, Fetch-protected". "0S" would mean "Key-0, Store-protected storage".

If you're interested in seeing a disassembly of object code instead of a mere display of storage, use the FORMAT command.

Examples:

D ,AS ← The next logical screen full of memory will be displayed in ASCII format.
D MYPROG.MYCSECT+24D 20 ← At offset 24D into MYPROG.MYCSECT 20 lines of memory will be displayed.
D PSW? Lines=20 ← A Display command will be executed where the PSW points in 31-bit mode and 20 lines of information will be shown.
D PSW? B=10 ← A Display command will be executed where the PSO points to in 31-bit mode and X'10' bytes will be shown. That's sixteen decimal, Cowboy.
D R12% ASC ← A Display command will be executed where R12 points to in 24-bit mode and the memory will be shown in ASCII format.

More Info: Enter Help COmmands DISPlay.



Create a “clone” (an exact copy) of an existing dsect map, assigning it a new name.
Helper dialog: DMAP ?

new-map-name The name you wish to assign to the copy of the existing dsect map.
old-map-name The name of the existing dsect map you have previously loaded with a DMap load command.
.label-name The name of a field label within the new dsect where you would like to establish a “zero point”. The dot (“**.**”) must be used.
+offset The hexadecimal offset within the new dsect where you would like to establish a “zero point”. You can use a negative offset also (“-offset”).

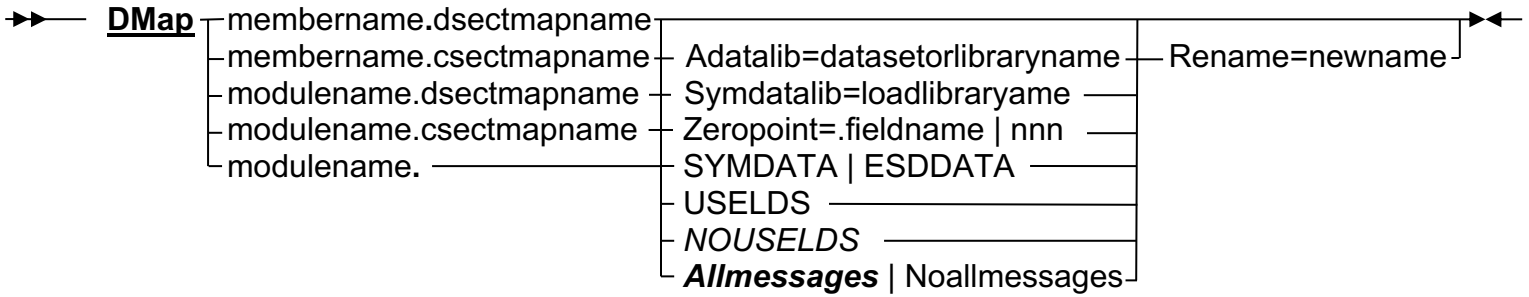
Remarks: This command gives you the option of establishing a “zero point” within the dsect map that will be referenced in a subsequent Using command.

Recent update: Enter “DMAP ?” and a “Helper Dialog” will appear. This will allow you to fill in all the command’s parameters!

IMPORTANT: When you have loaded a dsect into memory you must subsequently issue a Using command to “place” it at a specific virtual address. See the Using command.

Examples: DM NEWTCB TCBFIX.TCBRBP ← This creates a copy of the existing dsect map TCBFIX and establishes label.TCBRBP as its zero point.
DM TCBFRED,.TCBRBP ← This searches all maps for the label .TCBRBP. The first map found to contain this label is TCBFIX. A new dsect named TCBFRED is created but with the zero point set to .TCBRBP.

More Info: Enter Help COmmands DMap Clone. Also, Help COmmands Using.



Load a DSECT map into memory.
Helper dialog: DMAP ?

membername.dsectmapname	The name of a PDS dsect in ADATA or SYMDATA format. If membername name is omitted, then the load module scanned by any prior DMap command will be used again.
membername.csectmapname	The name of a CSECT that you wish to load as a dsect. This is useful in situations where one wishes to "preload" a csect in order to set a deferred breakpoint or hook at a label (.lablename) inside the csect, rather than using a hex offset.
modulename.dsectmapname	The name of a loadmodule and dsect to be loaded – OR a **.csectmapname** which will be loaded as a dsect.
modulename.	When you specify "modulename-dot", it forces z/XDC to LOAD a new dsect map – NOT clone an existing dsect map.
Adatalib= | Symdatalib=	Libraries to be searched for either ADATA or SYMDATA versions of a csect.
Zeropoint=	You may establish a "beginning" field using .fieldname or an offset using a hex value (or decimal with an "N" suffix).
SYMDATA | ESDDATA	These two interchangeable parameters both force z/XDC to IGNORE ADATA information. I don't know why I'd want that.
USELDS | *NOUSELDS*	This are specific to dump/XDC. You can force DMAP to "Use Local Datasets" in a dump/XDC session, OR ignore XDCMMEF and DSECTLIB tokens in a dump/XDC session.
***Almessages* | Noallmessages**	You can choose to view or discard messages that are generated during z/XDC's "hunt" through the search order.
Rename=newname	Allows you to load the dsect with a different name of your choosing. For example, if you loaded a dsect map for a particular TCB, you could name it JSTEPTCB to denote the Jobstep Task dsect map. Now it stands out.

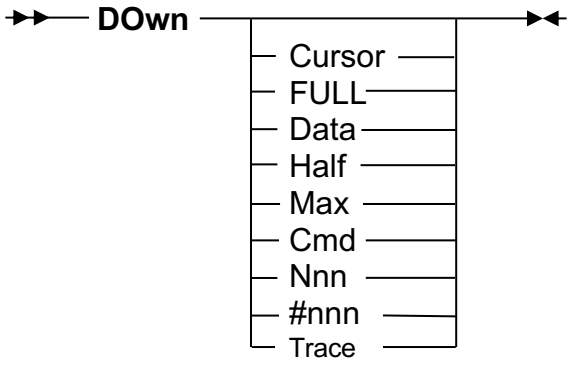
Remarks: The DMap command has two forms. This form loads a copy of the SYM (or ADATA) records for a dsect into memory. The "clone" form of the DMap command (appears before this command in this book) will make a copy of an existing dsect definition with another name.

IMPORTANT: When you have loaded a dsect into memory *you must subsequently issue a Using command* to "place" it at a specific virtual address. See the Using command. In order to make ADATA available to the DMap command one may need to issue the Set MAPLibs command.

Helper Dialog: **Enter "DMAP ?" and a "Helper Dialog" will appear. This will allow you to fill in all the command's parameters without having to memorize them. This is a real convenience.**

Examples: DM XDCMAPS.TCBFIX ← This command loads a copy of the TCB dsect from the XDCMAPS module supplied with the z/XDC product.

More Info: Enter Help COmmands DMap Load. Also, Help COmmands Using and/or Help COmmands SEt MAplibs



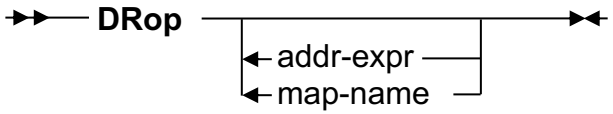
Move the display window down a specified distance through the session log.

omitted	Scroll down by the window's default scroll amount (as controlled by the SEt WIndow command).
Cursor	The window is scrolled to the line on the display that contains the cursor. This parameter may only be entered via a PFkey setting.
Full	Scroll the distance of a full window. "Page" is a synonym for this parameter.
Data	Scroll so that the previous screen's bottom becomes the top of the screen.
Half	Scroll one-half of the vertical distance of the screen.
Max	Scroll to the end of the session log.
Cmd	Scroll down to the next command.
nnn	Scroll down a specified number of lines.
#nnn	Scroll to a specifically numbered command in the log.
Trace	May only be used in the Working Window. If you have previously issued an UP to navigate to an earlier time (up) in the Session Log, then DOWN Trace will display the NEXT time z/XDC got control and painted a screen. Dave Cole points out a useful idea: Issue an UP MAX, then DOWN TRACE;RETRIEVE. This will allow you to see your debugging session run like a movie! That might be worth trying out some time.

Remarks: Every parameter in this command will accept simple arithmetic that alters the "value" of the DOwn command's output.

Example: DOwn Half 10+5 ← Same as entering "DOwn Half 15".
DO CMD-5 ← goes down one command in the session log (presuming that you're located "above" the current command) minus five lines.
DO 12-3+4 ← Same as "Down 11".

More Info: Enter Help COmmands DOwn. z/XDC's UP command works the same way – in the opposite direction.



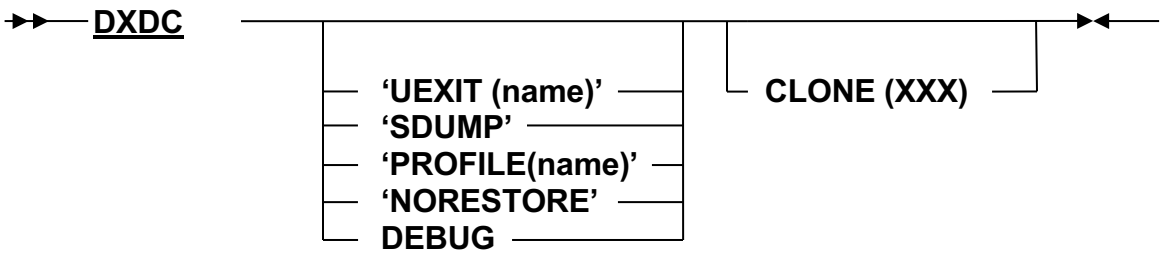
Release the base address for a dsect. This “undoes” the effect of a previously issued Using command.

addr-expr A location within the dsect (or dsects) to be dropped.
map-name The name(s) of the dsect(s) to be dropped.
omitted If no addr-expr or map-name is specified, then ALL dsects are dropped.

Remarks: If you wish to retain the dsect definition and place it elsewhere in memory, you can issue another Using command. In other words, you don’t have to issue the DRop command before reassigning the dsect elsewhere.

Examples: DR TCBFIX ← The dsect map named TCBFIX loses its origin address. It will appear in a List Maps command display as INACTIVE.

More Info: Enter Help COmmands DRop.



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

UEXIT(name) This is passed to dump/XDC as an argument. 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 This is passed to dump/XDC as an argument. If d/XDC, %DXDC, or z/XDC itself abends, the system will create an SVC dump.

PROFILE(name) This is passed to dump/XDC as an argument. Nominate a z/XDC profile other than the "factory default" as your favored profile for this d/XDC session. You can also issue a Profile Read command from within your dump/XDC session.

NORESTORE This is passed to dump/XDC as an argument. d/XDC stores information about a dump for use in a subsequent dump-reading session, in effect "journaling" much of the commands you issue during the session. This argument will NOT reload this stored information. Since it's REALLY convenient that dump/XDC plays back these saved commands, this argument isn't recommended.

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.

DEBUG This is a parameter, not an argument (so it doesn't get surrounding single quotes). In normal operation, dump/XDC suppresses IPCS's informational messages so that they don't interfere with dump/XDC's display process. If you get strange results in your dump/XDC session, you could start the session with "DXDC DEBUG". You will now see all of IPCS's informational messages, which might help in problem determination.

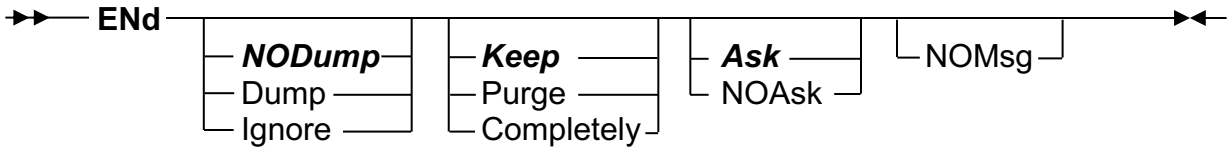
Remarks: This is how you "launch" d/XDC running IPCS under TSO READY, or ISPF. Arguments such as UEXIT, SDUMP or PROFILE must be enclosed with sinqle quotes. You may specify any combination of parameters, again enclosed in single quotes.

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. Also, Help Dumps Startingdumpxdc Dxdcproc and Help Dumps Startingdumpxdc Dxdcproc Arguments. See also IBM Publication SA22-7594-04 (MVS Interactive Problem Control System (IPCS) Commands.



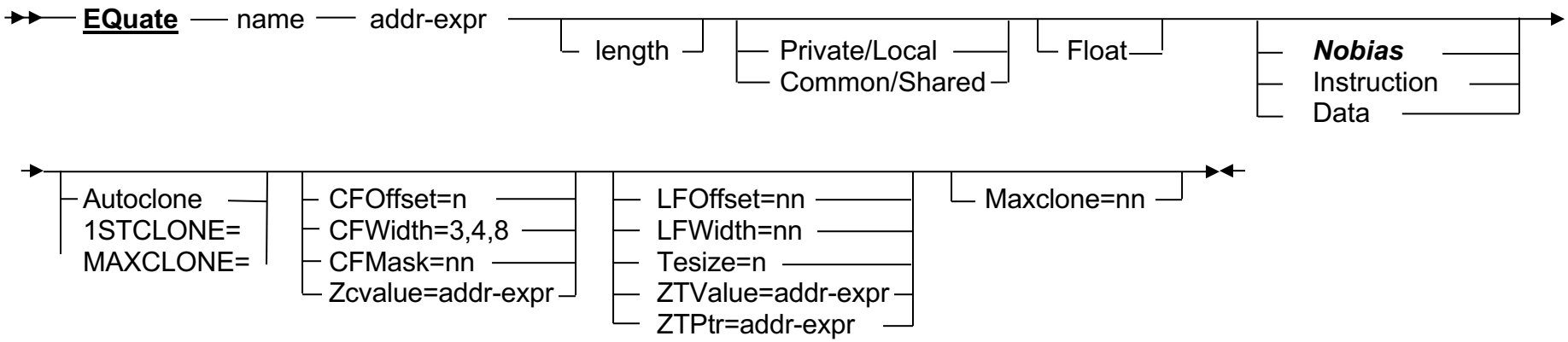
Terminate the z/XDC session.

NODump	A default action. z/XDC will suppress the printing of a dump.
Dump	z/XDC will print a dump of the session.
Ignore	Prevents z/XDC from altering the dump flag in Recovery Termination Manager. A dump may or may not be printed depending upon how the dump flag has been affected by other processes.
Keep	A default action. Retains breakpoints, maps and equates. Useful if the user expects z/XDC to regain control under another task or ESTAE/I.
Purge	Removes all breakpoints, maps and equates before terminating ALL of the user's session(s). A synonym of "Completely".
Completely	Removes all breakpoints, maps and equates before terminating ALL of the users' session(s).
Ask	A default action. z/XDC will ask the user if he or she really wants to end the session.
NOAsk	Bypasses the termination prompt. z/XDC proceeds with termination of the session.
NOMsg	z/XDC purges all messages that may be pending and have not yet been displayed.

Remarks: ENd allows a fair amount of "tailoring" to meet specific situations. My personal favorite is "END COMPLETELY" ("z/XDC, go away. Don't prompt me. I know what I'm doing").

Examples: END ← Prior to ending the debugging session, z/XDC will prompt the user to confirm with "Y" ("I want to end this session.") or "N" ("NO! I do not want to end this session.").
END NOASK ← z/XDC ends the session without prompting the user.
END COMP ← z/XDC terminates ALL debugging sessions for this user.

More Info: Enter Help COmmands ENd.



Create an Equate in memory at a specified address.

name A name of up to 63 characters in length. Names may include alphahanumeric or national characters (@,#,$._). If autocloning is desired then name becomes a "root" and subsequent copies of the equate will receive a suffix of "#nn" where "nn" is incremented for every instance of the equate.

addr-expr The location where the equate is to be placed.

length The amount of memory the equate is to "span", in hexadecimal numbers. In other words, equates may represents areas of storage as well as points.

Private A synonym of Local. Forces the equate to be treated as if it occurs only in the Private Area, even if it's actually assigned to a Common Area.

Common A synonym of Global and Shared. The equate will be owned by all address spaces even if it resolves to a private storage location.

Float The addr-expr is saved in unresolved form and is resolved whenever the equate is referenced. Thus, the equate "floats" as the addr-expr changes.

Nobias Default behavior. z/XDC will treat the memory pointed to by the equate as pure data.

Instruction z/XDC will consider the equate as pointing to an instruction, and will attempt to disassemble memory at the equate whenever it is referenced.

Data z/XDC will NOT consider the equate as pointing to an instruction.

Autoclone Directs z/XDC to create a list of named equates that refer to pointers in a chain, or elements in a queue according to Autoclone's own parameter list. If all other parameters are omitted, then z/XDC assumes these defaults: CFOFFSET=0, CFWIDTH=4, CFMASK=7FFFFFFF, ZCVALUE=0 and MAXCLONE=10.

1STCLONE= Normally z/XDC will number the first cloned equate as "1". You may force any initial value from 0 to 999999..

MAXCLONE= Sets a limit on the number of autocloned equates. When 1STCLONE=1, the range can be from 1 to 999999.

CFOFFSET=n The offset of the entry's chain field, starting from the beginning of the dsect. Can be expressed as a hexadecimal number OR a decimal number if followed directly by an "N". That is, "10" means 16, and "10N" means 10. The default offset is 0.

CFWIDTH=n Defines the width of the chain field. Permitted values are 3, 4, or 8. The default width depends on whether or not the CFMASK option is used. If CFMASK is not given, then the default width is 4.

CFMASK=n Defines a selection mask that is AND'd against the chain field's value to eliminate bits that are not part of the chain pointer. The width of the mast must match the width of the chain field (as defined by the CFWIDTH parameter).

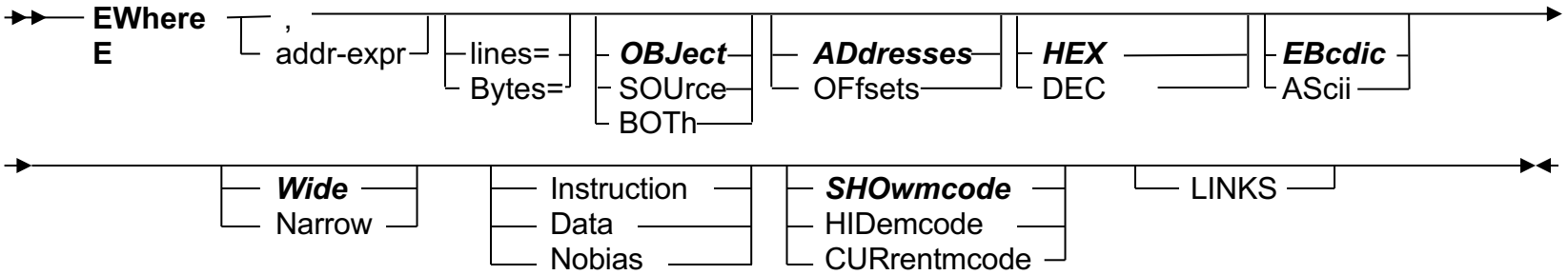
ZCVALUE= the "zero chain field" value. z/XDC always considers that a chain has ended when it finds a zeroed chain field. However, you can also give an address expression that will be compared to the contents of each entry's chain field. A match identifies the chain's last entry.

You can also use the keywords "NEGATIVE" or "NOT POSITIVE" as operands of the ZCVALUE parameter. Both of these keywords cause any zero or negative value to signify the end of the chain. When testing for a negative value, the CFMASK= parameter is ignored. z/XDC checks the field's hi-order bit. If it is ON, then the chain is ended.

LFOFFSET=n If the table entries have a length field, then this operand provides the offset of that field from the start of each table entry. As with CFOFFSET the number can be expressed as a hexadecimal number OR a decimal number if followed directly by an "N". That is, "10" means 16, and "10N" means 10. The default offset is 0.

LFWIDTH= If the table entries have a length field, then this operand provides the width of that field. LFWIDTH may range in value from 1 to 8. There is no default value for this parameter. If LFOFFSET= is stated, LFWIDTH must also be stated.

TESIZE= If the table has fixed-length entries, or if the entries are variable, but have a fixed section of constant size, then this parameter provides the length of the fixed section of the entry. Tesize is either pure hexadecimal, or a number followed directly by the character "N". The default value is 0.



Cause z/XDC to dis-assemble memory pointed to by the Error Program Status Word. This is the error level PSW's abend address.

Shortcut command equivalent: "E"

addr-expr This parameter is optional. If stated, it must resolve to a location that is at or preceding the execution address for the EPSW. If these rules aren't followed, z/XDC ignores addr-expr and gives you the output of EWhere without using addr-expr. Note, however that if you want to use any of the other parameters in this command, AND you don't want to use the addr-expr parameter then separate the EWhere command from any other parameters with space-comma-space ("_,_"). That looks like an emoticon, but WHICH one?

IF you use the space-comma space option, then:

Lines=nnnn The number of lines the user wishes to see. The upper limit is 10,000. If omitted, z/XDC will fill the current window.

Bytes=nnnn The number of bytes the user wishes to see, expressed as a hexadecimal number ("10" = 16). If you suffix "nnnn" with a "N", z/XDC will treat nnnn as a decimal number.

Source/*Object*/BOth If a source level map is available, source code can be shown as part of the formatted display. OBject is z/XDC's normal output. BOth should be obvious. These settings have no true default, as they can be set in the Profile facility (in the SET FORMAT command).

Addresses The virtual address of each line of the display will appear on the left side of the panel.

Offsets The offset from the start of the data's object (load module, csect or dsect) will appear on the left side of the panel.

***Data*/Decimal** Displacement fields of machine instructions will be displayed as hexadecimal or decimal numbers as desired. Operative ONLY when source statement support is not being used.

***Ebcdic*/AScii** Memory will be interpreted in EBCDIC or ASCII format, as desired.

***Wide*/Narrow** Shows 32 bytes of memory (default) per line, or 16 bytes per line.

Instruction/Data/Nobias Causes the Ewhere command to show the target address-expression as instructions or data. When Nobias is used, and absent other factors, storage contents will be interpreted as machine instructions.

***SHOWMCODE*/HIDEMCODE** If z/XDC's Source Statement support (ADATA or DWARF) is being used, z/XDC will SHOW or HIDE macro expansions. If you use these parameters you MUST specify an address-expression.

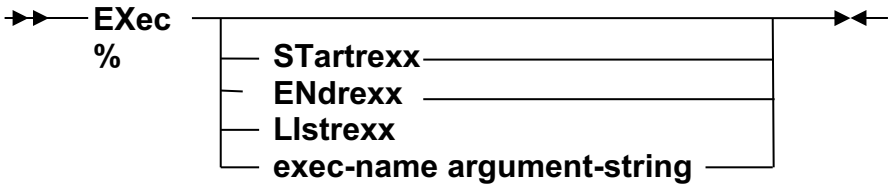
Links When storage is being formatted via a disassembly, and if a line of display shows a snippet of data that is 3, 4, or 8 bytes long, then LINKS will attempt to resolve that snippet as a storage pointer. Then it will display that resolution as a comment on the display line. This is useful, but VERY CPU-EXPENSIVE.

Remarks: z/XDC keeps track of two distinct environments: The Retry Level (where your program will resume if you perform a Trace, or Go command) and the Error Level (the execution environment at the time of the Abend, breakpoint or the execution of a trace). Generally, the Where and Ewhere commands will produce similar displays EXCEPT when the Retry Level and Error Level environments are *different*. Then the Where command will show the retry-level resume address, while the Ewhere command will show the error level abend address.

"E" The Ewhere command may now be used as a shortcut command on any (permitted) line of a z/XDC display.

Examples: EW ← z/XDC will disassemble where the PSW points to.

More Info: Enter Help COmmands EWhere. For a good discussion on the Error level vs. the Retry Level in z/XDC's documentation, see Help EXEcutionlevels..



Invoke a REXX exec from within your z/XDC session.

STartrexx Optional parameter. z/XDC calls IRXINIT to locate z/XDC's Rex interface function package (load module xxxEFMVS (where "xxx" is the current "clone name)), and loads it into storage. It also locates the user's REXX exec library (//xxxREXX ddname) and opens it. It also builds a REXX Environment Block.

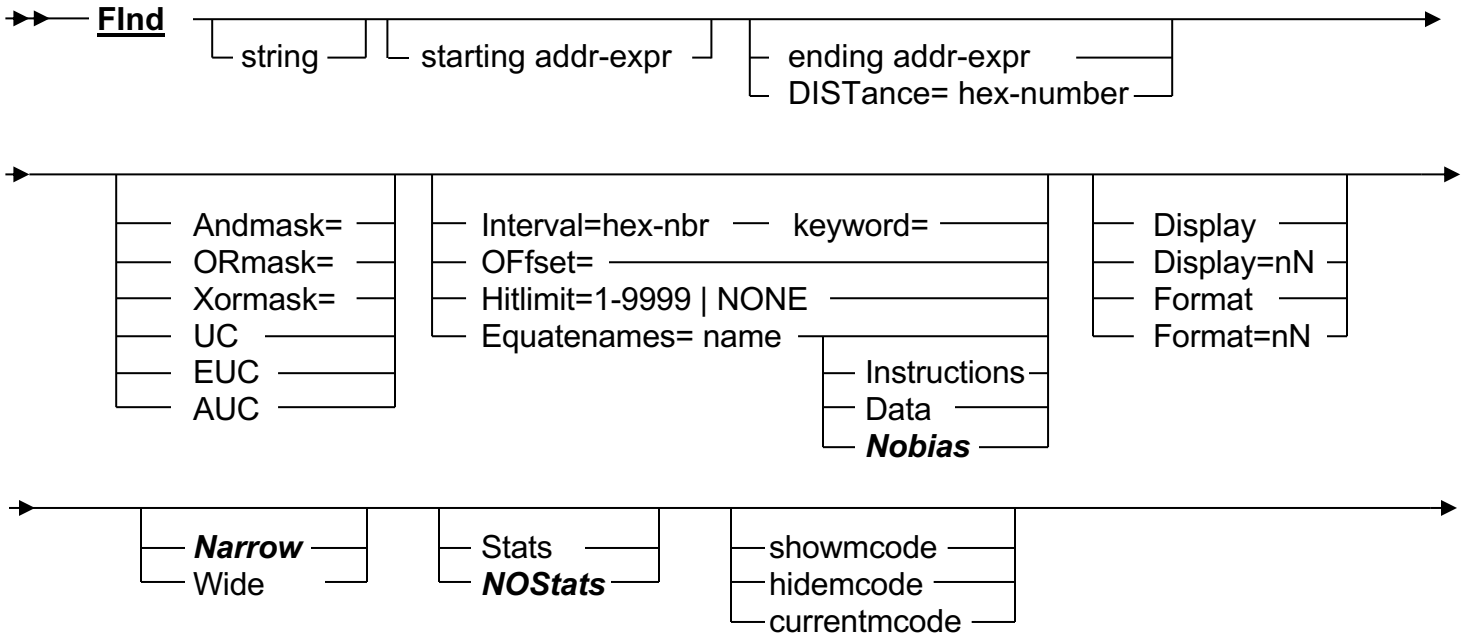
ENdrexx Shut down the REXX interface.

LIstrexx If used, this issues the LIST REXX command for you. So, why not just issue LIST REXX?

exec-name argument-string The name of the REXX exec to be run, along with its arguments, if any.

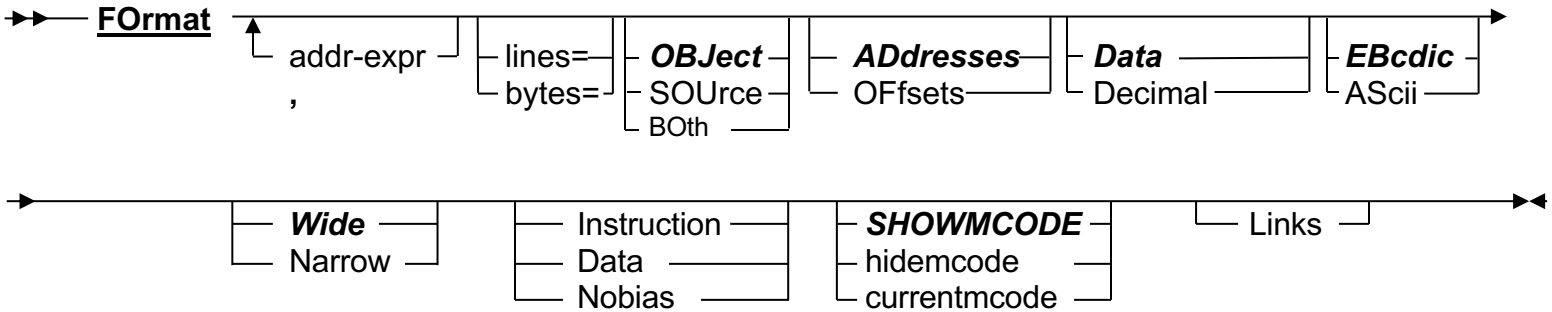
Remarks: z/XDC supports the execution of REXX execs, which allows you to write your own "z/XDC commands" in the REXX programming language. This command also accepts a percentage sign instead of the EXec command, which would mean less typing.

More Info: Enter Help COmmands Command-name Exec. See also Help Rexx. For a list of built-in functions that z/XDC provides, see Help Rexx XDCServices.



Search for a specific string of data sequentially from a given point in memory.
Helper Dialog: "FIND ?"

string	That which you seek(!). The string can be Hexadecimal, EBCDIC (enclosed in single quotes), ASCII or a mixture of all. If string is omitted, z/XDC uses the previously defined string. Thus, in searching for multiple occurrences of the same "thing", one can merely re-issue the FIND command without parameters.
Starting addr-expr	The location in memory where the user wishes to begin the search. The search will proceed "upwards" in memory from this point.
Ending Addr-expr or DISTance=	Where the user wishes the search to end. DISTance= is a hex number ranging from 1 to two-times-ten to the 64th power minus one. If DISTance is specified, z/XDC will begin with Starting addr-expr and will stop when DISTance is reached.
Andmask=/Ormask=/Xormask=	Boolean arithmetic/Masking operations to be performed on the target data before comparison. The user supplies a mask of any length.
UC	Cause what is found to be upper-cased depending upon the current format set in the SET FORMAT command.
EUC	Cause what is found to be upper-cased in EBCDIC format.
AUC	Cause what is found to be upper-cased in ASCII format.
Interval=	Normally, FIND operates on every sequential byte within the range of storage to be searched. Interval cause the find to be executed at "every nth byte" instead. Can be a pure hexadecimal number or a decimal number suffixed by "n" (ex: "10N").
Keyword=	Equivalent to Interval. Keyword can be "BYTE", "HWORD", "FWORD", "DWORD", "QWORD", "HALFWORD" , "FULLWORD", "DOUBLEWORD", "QUADWORD", "PAGE".
Offset=	Either a hexadecimal number or a decimal number suffixed by an "N". If an Interval is specified that is greater than 1, then the offset is used to test for a match at the specified offset within the interval.
Hitlimit=	Normally, FIND stops at the first "hit" in sequential storage. Hitlimit causes FIND to run until a predetermined number of "hits" are found. This may be a decimal number up to 1000.
=NONE	The FIND command will show ALL "hits" within the range of storage to be searched.



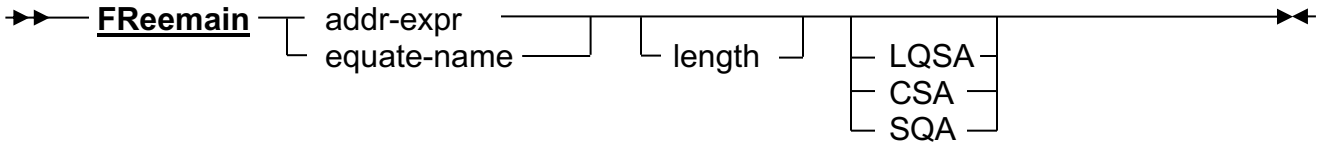
Disassemble memory at a specified location.
Short-cut command equivalent: "F"

addr-expr	The location in memory where the disassembly is to begin. If addr-expr is omitted, then the next screen-full of memory is disassembled. If addr-expr is omitted and other parameters are desired, then a comma must be used before stating the other parameters. The comma is a "place-holder".
Lines=	The number of lines the user wishes to see. The upper limit is 10,000. If omitted, z/XDC will fill the current window.
Bytes=	Specifies the number of bytes to be displayed in hexadecimal, or decimal if suffixed with "N" (example: 100N=100 decimal bytes).
Source/*Object*/BOth	If a source level map is available, source code can be shown as part of the formatted display. OBject is z/XDC's normal output. BOth should be obvious. These settings have no true default, as they can be set in the Profile facility (Under the SET FORMAT command).
ADdresses	The virtual address of each line of the display will appear on the left side of the panel.
OFfsets	The offset from the start of the data's object (load module, csect or dsect) will appear on the left side of the panel.
Hexadecimal/Decimal	Displacement fields of machine instructions will be displayed as hexadecimal or decimal numbers as desired. Operative ONLY when source statement support is not being used.
Ebcdic/AScii	Memory will be interpreted in EBCDIC or ASCII format, as desired.
Wide/Narrow	Shows 32 bytes of memory (default) per line, or 16 bytes per line.
Instruction/*Data*/Nobias	z/XDC makes a number of decisions in the Format process, many of which are dependent on a great many other variables. Consequently, the Format command may, under certain circumstances, mis-interpret data as instructions if the format command is attempted within data rather than object code. The Instruction, Data and Nobias parameters affect z/XDC's determination in the Format command. Please see Help Commands Format for a thorough understanding of how to use these parameters.
***SHOWMCODE*/HIDEMCODE/CURRENTMCODE**	z/XDC will SHOW or HIDE macro expansions - IF – z/XDC is displaying source statements at the time.
Links	When storage is being formatted via a disassembly, and if a line of display shows a snippet of data that is 3, 4, or 8 bytes long, then LINKS will attempt to resolve that snippet as a storage pointer. Then it will display that resolution as a comment on the display line. This can be useful, but is VERY CPU-EXPENSIVE.

Remarks: **This is one of the most important z/XDC commands**, one that will be used in every session. Use the Display command if what you really want is a "dump" of raw storage.

Examples:
F MYPGM.MYCSECT.MYLABEL 999 ← z/XDC will disassemble memory at the specified location for 1000 lines. PF7 and PF8 can then be used to scroll forward and backwards in the disassembled code.
F ← The next logical screen full of data will be disassembled.
F PSW? Li=10 ← z/XDC will produce a FORMAT'd panel starting where the Program Status Word points to, in 31-bit addressing mode. Only ten lines will be displayed.

More Info: Enter Help COmmands FOrmat.



Release an area of storage that was previously obtained by a GETMAIN macro or a z/XDC GETmain command.

addr-expr The address of the area to be freed. This can be an Equate that resolves at the start of or inside the storage to be freed.

Since any GETMAIN command issued within z/XDC automatically receives an equate name of “AREA” (for the most recently executed GETMAIN) and AREA#nn (where nn is the sequential number assigned by any GETMAIN command) it’s often handy to issue “FREEMAIN AREA” (for the most recent extent created with GETMAIN), or “FREEMAIN AREA#nn” (for any other storage extend created with any other GETMAIN).

FREEMAIN will also delete any other equates defined within this storage, Finally, FREEMAIN will fail unless addr-expr is doubleword aligned.

length This is a hexadecimal value giving the length of storage to be freed. The FREEMAIN SVC rounds this value up to an 8-byte multiple. This operand is required unless the address operand is a pure equate name in which case the length operand can be omitted and the length of the equate is used in its place. Bad news: If you FREE an area less than the length of the extent you created with GETMAIN, then *you will have created your own memory leak*, as “orphaned” storage will remain. Good news: At the end of the debugging session z/XDC will clean this up for you.

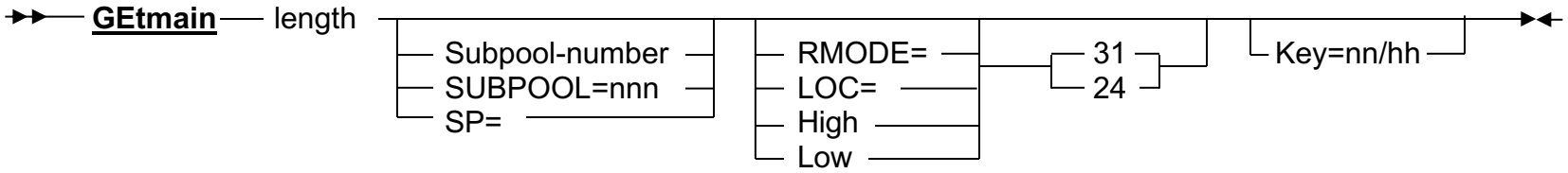
Equate-name If the user invokes the Freemain command with the z/XDC-defined equate name (via the GETMAIN command), then the addr-expr and length parameters need not be given (z/XDC always assigns an equate name of AREA#n to the memory accessed via an z/XDC GETmain command).

LQSA, CSA, SQA These parms may be specified when z/XDC is running APF-authorized. If the storage to be freed resides within one of these areas, then you MUST specify this parameter to avoid freeing storage from these areas accidentally.

Examples:

FR AREA#3	← The storage represented by the equate AREA#3 is freed, along with any equate names that occur within the memory’s extent.
FREEMAIN FRED 50	← z/XDC will free X’50’ bytes of storage beginning at the storage location beginning at the address of the equate “FRED”.
FR F00000,100,SQA	← If z/XDC is running authorized, and if location X'00F00000' falls within the System Queue Area (SQA), and if that storage currently is allocated, then this command frees it. (This command fails when any of the above listed conditions are not met.)

More Info: Enter Help COmmands FReemain. See also Help Commands GEtmain.



Obtain a portion of memory from a specified subpool.

length The desired amount of memory, given as a hexadecimal number. The STORAGE OBTAIN service will automatically round this up to an 8-byte multiple.

Subpool-number/ SUBPOOL=nnn/SP= The desired subpool identifier, given as a decimal number.

High The desired memory is to be located "above" the 16MB line, in 31-bit storage. An alias is RMODE=31 or LOC=31.

Low The desired memory is to be located "below" the 16MB line, in 24-bit storage. An alias is RMODE=24 or LOC=24.

omitted If neither High or Low are specified, z/XDC will obtain the memory from either above or below the line depending on the addressing mode of the program being debugged. The user may issue LIST PSW FORMAT to inspect the address-mode bit, if desired.

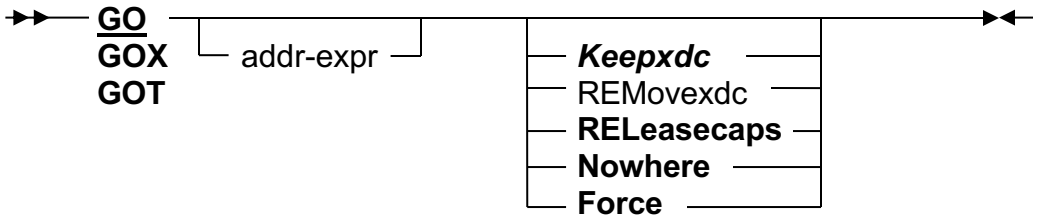
Key=nn/hh Requests that the storage be assigned to a particular key. The parameter may be specified in decimal (Key=11) or hexadecimal (Key=0B, or key=B). HOWEVER, certain rules apply to using this parameter. For one thing, it is effective only for multi-key subpools. Also, when z/XDC is APF-authorized, the System will only accept key values permitted by your program's key mask. Typically, you would be limited to Key=8 or =9. Read the Help Subsystem for important information on this parameter.

Remarks When GEtmain is issued, z/XDC creates two equates in memory: "AREA" and AREA #nn (where nn is the number of times you've issued the GEtmain command in this debugging session). On any subsequent GEtmain command, AREA will be updated to the most recently allocated storage and AREA#2 (or larger) will be created. This way, you can issue GEtmain in a script and refer to the most recently "got-mained" storage as "AREA".

This command may not be used in Foreign Address Space Mode. Well, you CAN use the command, but the memory will be allocated to YOUR address space, not the Foreign one.

Examples: GE 100,78 ← z/XDC will obtain X'100' bytes of storage from subpool 78. It will also create two Equates:
AREA ← This Equate is created or updated every time the GEtmain command is executed. It always points to the most recently obtained storage.
AREA#n ← This Equate uniquely identifies each area of storage created by a GEtmain command.

More Info: Enter Help COmmands GEtmain. See also Help COmmands FReemain.



Release the program being debugged for processing by the operating system.

addr-expr The desired resume address. If omitted, then z/XDC uses the address pointed to by the retry-level Program Status Word (PSW).

KEEPxdc This is the default. All current instances of z/XDC will remain and the current recovery environment will remain unchanged.

REMovexdc If z/XDC got control in this debugging session via a HOOK, then this parameter will remove the current instance (only) of z/XDC from your program's recovery environment. If other instances of z/XDC are pending in the recovery stack they will remain in effect. This parm is intended to remove instances that have been established dynamically via the Hook command (or the old Hook script). It is NOT a comprehensive way to remove z/XDC under all circumstances. Read and understand the Help Subsystem's instructions before use of this parameter.

RELeasecaps This ends a debugging session without also ending the program being debugged. The program resumes but all z/XDC-related processes are terminated, including all maps, all breakpints all equates and all dsects.

Nowhere This causes z/XDC to resume the program and return to z/XDC right away, which allows Recovery Termination Manager to reschedule z/XDC. This parameter is intended to work with z/XDC's recently added SET AUTH command, which allows a user to make any debugging session APF-authorized (subject to system security rules). See Help COMmands SEt AUTH for more information.

Force When using TRAP2 instructions for trapping and tracing with z/XDC, the Trap Save Area can become corrupted. In this case message DBC799 will be displayed. It is then incumbent upon the user to restore the proper values to the PSW, General Registers and AR15. One may then issue the GO FORCE command to resume the debugging session. Note: It is beyond the scope of this book to discuss how the Trap Save Area corruption can occur, but an excellent overview can be found by entering "Help Messages DBC799".

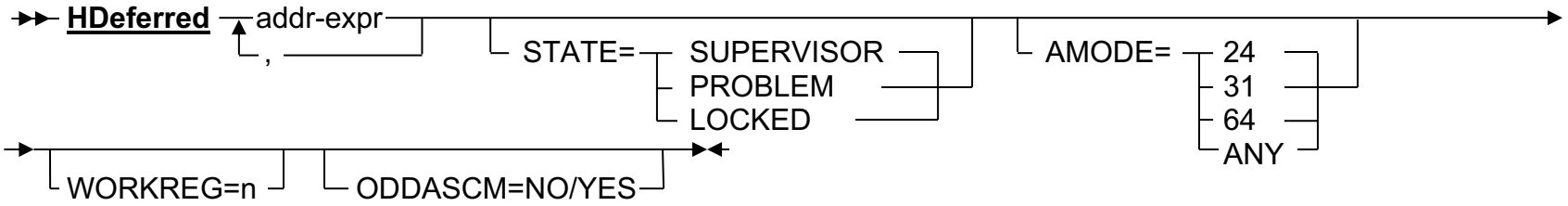
Remarks: This is a CORE command in z/XDC. The user's program will resume execution, and will stop at the next breakpoint or abend. GO, GOX and GOT are equivalent, with minor differences. With GOX, z/XDC will attempt to restore whatever information was being displayed prior to the GOX command. GOT causes the user program to be resumed without clearing the display screen.

We recommend that you read the Help Subsystem on this command, especially with attention paid to the various parameters above. There are a lot of moving parts here.

This command may not be used in Foreign Address Space Mode.

Examples:
GO R10! ← The user's program is resumed NOT where the PSW points to, but where the contents of R10 point to (in 64-bit mode).
GO REM ← IF z/XDC got control of the debugging session by means of a Hook, then the program is resumed and z/XDC is removed from the picture.
GO N ← Used in conjunction with the SET AUTH command, this can help a previously unauthorized debugging session in another address space to become APF-authorized.

More Info: Enter Help COmmands GO.



Create a deferred Hook in the desired module.

addr-expr An address expression that names a module, a module.csect and (if desired) the offset where the deferred Hook is to be placed. The "base" of the addr-expr MUST be the primary name of a load module. This parameter is required. All other parameters are optional, and may be omitted.

STATE= Omit this operand to have the Hook set in the current execution state. Use this operand only if the state of the Hook Point is going to be different from the execution state of the current code at the moment you set the deferred Hook. If you are placing a hook into code that you expect to run locked or disabled, then you would select "STATE=LOCKED".

AMODE= Omit this operand to have the Hook set in the current addressing mode. Use this operand only if you wish the hook to be placed in an addressing mode that is different from the current addressing mode. ANY directs z/XDC to use the current addressing mode.

WORKREG=n A register that can be used by the generated Hook processing code that is specific to this HDEFERRED request. You need this if a) you wish to hook address that are above the bar, or b) you wish to hook code that may be executing in secondary or in home ASC-MODE (in which case you will also need to specify ODDASCM=YES). The register must range from 0-F and the Workreg will be set to all zeroes.

ADDASCM=NO/YES Indicates that the hook needs to support code that is executing in secondary or home ASC-MODE. Failure to use this parm when executing in secondary or home ASC-MODE will result in a S0D3 abend. So, if you specify ODDASCM then you MUST also specify WORKREG

Remarks: The HOOK command creates a z/XDC session "on the fly". The HDeferred command creates a Hook for a module that is not yet in memory, but that will be loaded in the future. Modules are loaded into storage as a result of the LINK(X), LOAD, ATTACH(X) and XCTL(X) macros, and when a target module is read into storage as a result of one of these macros, then z/XDC will set the hook. Note that if the system decides to satisfy the LINK(X), LOAD, ATTACH(X)or XCTL(X) macro with a copy of the module that *already resides in storage*, then the deferred Hook definition is ignored and no new Hook is created. Therefore, HD works on only NEW COPIES of the target module.

This is an extremely cool command! z/XDC's HOOK facility (of which HDeffered is a part) completely solves the conflict between z/XDC's ESTAE and ones defined in your own programs. Consider HOOK as the "uber-breakpoint", which WILL give you control wherever you want no matter what other recovery structures are in place.

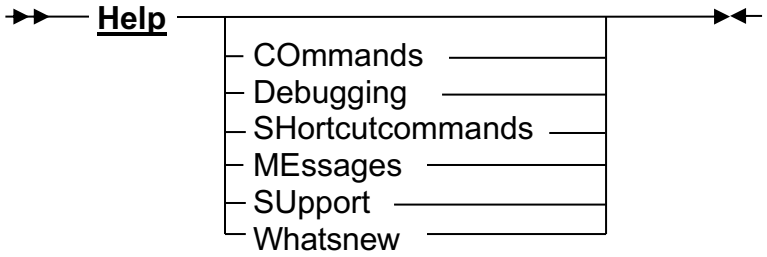
[*How it was: Back in the old days, if you set a breakpoint "over there" in storage and entered GO, by the time execution reached the breakpoint any number of other ESTAEs could have been issued. Then, z/XDC lost control of the debugging session, and your application went down with a S0C1. No bueno. HOOK solves this problem.]*

Hook is available in four ways: As an interactive command, or via the Shortcut command "K", as a deferred HOOK (see Help COmmands HDeffered) or as a macro call ("#XDCHOOK").

With Release z2.2, Hook's scope has been greatly expanded. It is no longer SVC-based, and can now be used in PC routines, SRB routines, FRR-protected code, Locked code, Disabled code as well as in its traditional environments.

Examples: HD mypgm.mycsect ← When mypgm.mcsect is loaded, a Hook will be placed so that an z/XDC session will initialize when the code executes.

More Info: Enter Help COmmands HDeferred. Also, refer to Help Debugging Hooks. You REALLY should familiarize yourself with this powerful command. There are important considerations well beyond the scope of this book.



Access z/XDC's comprehensive Help Subsystem.

COmmands	Access the main Commands panel. You may select further down through the system from here using the "S" line command.
Debugging	A useful overview for "getting started" with z/XDC
Shortcutcommands	Access a table of z/XDC's single-character "shortcut commands" (single characters that may be input in column one of a z/XDC display). Shortcuts are very powerful ways to get z/XDC commands accomplished quickly.
Messages	Access z/XDC's message facility. If preferred, the user may put an "H" next to a displayed z/XDC message and press Enter. This is a "hyperlink" to the explanation.
SUpport	When all else fails, contact Cole Software for assistance. We are always glad to hear from you.
Whatsnew:	Whenever a new z/XDC release becomes available, this section is updated. It's like reading a Release Guide, but on your terminal.

Remarks: Help is a pyramid structure with hyperlinks that will navigate to the information you wish to read. Use "S" in column one of any Help display to and press Enter to follow the hyperlink.

With so many frames of Help information to you, there just isn't any practical way to describe all that is available in this page. Thus, only a few of the many topics within the Help Subsystem have been listed.

When you issue HELP without any operands, you will see a main panel with two kinds of information. Non-highlighted sub-topics contain tutorial discussions, and highlighted subtopics are references.

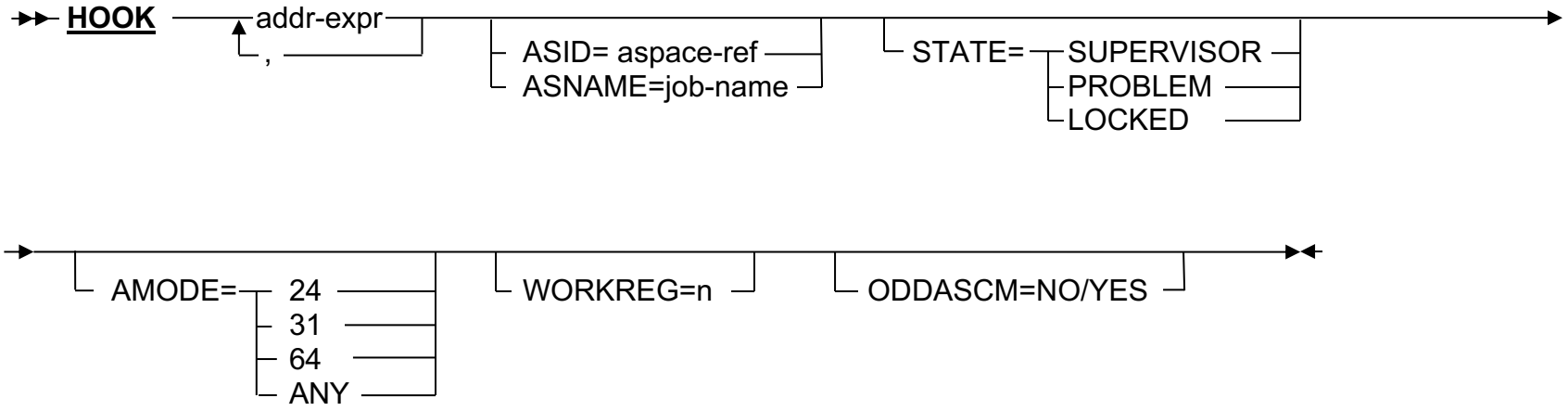
Each Help Panel's unique name appears at the top of the panel. They follow an index structure, so instead of entering "Help" and then selecting "Commands" you could enter "Help COmmands" and go directly to that area of Help. The minimum abbreviation of each Help panel's name is CAPitalized.

Whenever z/XDC presents a message you can research it by putting an "H" next to the message and pressing Enter. You'll go straight to the relevant page.

Help's contents discuss a good deal of z/OS architecture in addition to product-specific information.

To leave Help, enter END or type another z/XDC command.

For a top-down "index" into z/XDC's Help system, enter the LIST HELP command. LIST HELP opens a vertical menu of sub-topics. Enter an "L" next to any category to expand it. When no further expansion is possible, put an "H" next to the results and press Enter to go to that specific page within Help.



Create a debugging session regardless of prior existing recovery environments
Shortcut command equivalent: "K"

addr-expr An address expression that names a module, a module.csect and (if desired) the offset where the deferred Hook is to be placed. The "base" of the addr-expr MUST be the primary name of a load module.

ASID= Specify a specific address space that currently exists. ASID can be a 4-digit hexadecimal number (presumably from the Address Space Vector Table), a 1-8 character job name, or a keyword such as HOME, PASID, etc. For more information see HELP COMMANDS SYNTAX ASIDS.

ASNAME= Specifies the address space by name. Here's an important distinction: If you use ASNAME=jobname and the address space does not yet exist, AND if it will exist in common storage, then the hook is created anyway. Conversely, if you use ASID=jobname and the address space doesn't yet exist, the hook command will fail.

STATE= Omit this operand to have the Hook set in the current execution state. Use this operand only if the state of the Hook Point is going to be different from the execution state of the current code at the moment you set the deferred Hook.

AMODE= Omit this operand to have the Hook set in the current addressing mode. Use this operand only if you wish the hook to be placed in an addressing mode that is different from the current addressing mode. ANY directs z/XDC to use the current addressing mode.

WORKREG=n A register that can be used by the generated Hook processing code that is specific to this HDEFERRED request. You need this if a) you wish to hook address that are above the bar, or b) you wish to hook code that may be executing in secondary or in home ASC-MODE (in which case you will also need to specify ODDASCM=YES). The register must range from 0-F and the Workreg will be set to all zeroes.

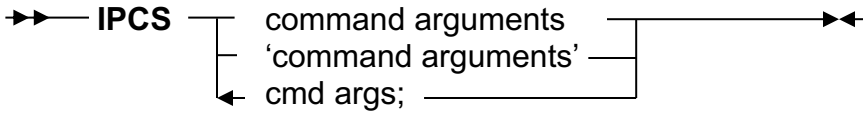
ADDASCM=NO/YES Indicates that the hook needs to support code that is executing in secondary or home ASC-MODE. Failure to use this parm when executing in secondary or home ASC-MODE will result in a S0D3 abend. So, if you specify ODDASCM then you MUST also specify WORKREG

Remarks: The HOOK command creates a z/XDC session "on the fly".

This is an extremely cool command! z/XDC's HOOK facility (of which HDeffered is a part) completely solves the conflict between z/XDC's ESTAE and ones defined in your own programs. Consider HOOK as the "uber-breakpoint", which WILL give you control wherever you want no matter what other recovery structures are in place.

[How it was: Back in the old days, if you set a breakpoint "over there" in storage and entered GO, by the time execution reached the breakpoint any number of other ESTAEs could have been issued. Then, z/XDC lost control of the debugging session, and your application went down with a S0C1. Bad juju. HOOK solves this problem.]

Hook is available in four ways: As an interactive command, or via the Shortcut command "K", as a deferred HOOK (see Help COmmands HDeffered) or as a macro call ("#XDCHOOK").



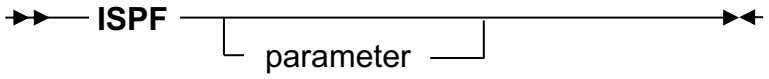
Issue one or more IPCS commands from within a dump/XDC session.

Command arguments An IPCS command, with its argument(s), if any.
'command arguments' If a command string contains syntactically sensitive characters, then it must be enclosed within tics (').
cmd args; You may submit multiple IPCS commands (with its arguments if any) separated from one another by semicolons (;).

Remarks: The ability to issue IPCS commands from within a dump/XDC session is one of dump/XDC's stronger features. It allows the close integration of both dump/XDC and IPCS, permitting one to have the best of both worlds – in a single place.
Further, when dump/XDC returns the output of an IPCS command to the user, storage fields shown in the output of the IPCS command are now sensitive to z/XDC's built-in point-and-shoot commands. This can be seen in the output of the IPCS SYSTRACE command, for example. Well done, Peter Morrison!

Examples: IPCS SYSTRACE (Hey, it's the only example I know of! –Bob)

More Info: Enter Help COmmands IPCS.



Invoke ISPF services, or re-invoke the ISPF interface.

parameter any valid ISPF parameter

Remarks: There's really not much to say here.

Examples: ISPF SPLIT

More Info: Enter Help COmmands ISPF.

LIBrarylists
LL

Manipulate c/XDC's access to IBM C/C++ Metal C source libraries.

Remarks: This is the first page in what will be a series of pages detailing the LIBrarylists command. There's a lot here, so hang in there.

NOTE: *Before you invest the time in reading this, the Librarylists command may NOT be necessary for you*. In recent years IBM has updated Metal C, so that c/XDC can "find" your source library automatically. ALSO, you can invoke z/XDC's MAP command with the SOURCELIB= and DWARFLIB= parameters. This makes the LIBRARYLISTS command "vestigial" for all but the most complex cases.

If you are running z/OS 2.4 or later, IBM's Metal C has the improvement included. If you are running z/OS 2.3 you must have IBM APAPR P199202 applied. In ALL cases, you must have z/XDC maintenance updated to at least the MDL-2104C level.

This Librarylists command was built in the "bad old days" before IBM improved Metal C (at our suggestion).

The LIBrarylists command allows you to perform a myriad of manipulations on your source datasets. You can:

1. ADD ← Add new lists of libraries to an existing list or reorder the list;
2. CHECKPOINT(or CKPT) ← save all libraries into a z/XDC Profile;
3. DELETE ← Delete lists or patterns of names from a Library;
4. DEFAULT ← identify the default list;
5. REDIRECT ← Cause c/XDC to look elsewhere for source library datasets;
6. RESET ← Clear one or all existing LIBrarylist definitions;
7. SHOW ← query the existing LIBrarylists defined;
8. TEST ← Test patterns and redirection settings.

The following pages will discuss each of these options in order.

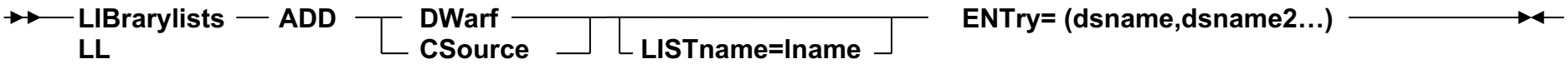
The LIBrarylists command is similar in concept to z/XDC's SET MAPLIBS command. Once created, a list of library names exists only for the debugging session and must be recreated – UNLESS – you issue LIBRARYLIST CHECKPOINT and then PROFILE SAVE. PROFILE SAVE <blank> will set this list as a default in your own, personal default profile. PROFILE SAVE <name> will set up a default list for that named Profile.

LIBrarylists supports sequential datasets, members of a PDS or PDSE, PDS or PDSE datasets, ZFS files, ZFS directories, other list names and DDNAMES. Entries that contain a "." are presumed to be MVS dataset names. Entries that contain a "/" are presumed to be ZFS filenames. ZFS filenames are case-sensitive. DDNAME names must be preceded with "////". All other entries are presumed to be list names.

In doing its work LIBrarylists builds and manages three kinds of lists that may be significant to you but are beyond the scope of this book. Please see Help * REDIRECTLIST, Help * CANDIDATESLIST and Help * RESULTSLIST.

Note: LIBrarylists is a complex command, so it would be an *excellent* idea to create a script or series of scripts to handle LIBrarylists.

More Info: Enter Help COmmands Librarylists. See also Help DEbugging C. See also Help DEBugging C LIBrarylists GLOSsary. ← The terms discussed therein are REALLY important to your understanding of the Librarylists command!



Create a new list of C source datasets, add new entries or reorder entries within a specific list.

DWarf Select a DWARF-type library to create a list from. This is a "Family" designation

CSource Select a C source-type library to create a list from. This is a "Family" designation.

LISTname A character string that identifies the Candidate List to be updated or created. LISTNAME can be abbreviated to LIST, LNAME or LN. The actual name of the list ("lname" as above) may not exceed 8 characters in length.

LISTname omitted The default list is used (if one exists).

ENTry= A parenthetical statement that contains the names of source datasets, separated by commas. When adding only a single dataset name, the parentheses are optional.
The "dsname" may be any of the following: the name of a classic z/OS sequential file (including a specific PDS member name), the name of a library (PDS or PDSE), the name of an HFS file, or foler, the DDNAME of a current allocation of one or more datasets, files, folders and/or libraries.

ENTries= A synonym for "ENTry".

Remarks: When adding new entries to a previously defined list, they will appear in the top of the list. If a dsname already exists in the list, it is reordered to the top of the list.

DWARF vs. CSOURCE: Most of the time you'll use the LIBrarylists command to tell c/XDC where DWARF resides. There are only two cases in which you'll need to specify CSource in your LIBrarylists command:

1. When the C source statements have been moved after compilation, and
2. When the C source statements were originally specified to the compile step in-line to the JCL as a //SYSIN file.

Examples: LL ADD DWARF LISTNAME=LIST1 ENTRY=(MY.DATA,//u//fred,MY.DWARFLIB(SAMPLE),////MYDDNAME) ← This command added 4 entries to *LIST1* for *DWARF* links. The entries are added to the list in the same order as they are listed. MY.DATA will be the 1st entry in the list, //u//fred the 2nd, and so on.

This command creates a list that looks like this:

```
LIST    LIST1
#1 MVS  MY.DATA
#2 HFS  //u//fred
#3 MVS  MY.DWARFLIB(SAMPLE)
#4 DDN  ////MYDDNAME
```

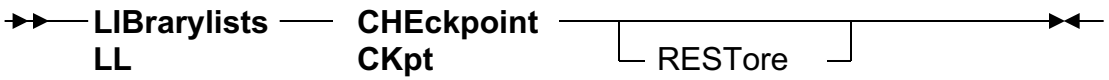
Note that LIST1 now has numbered "elements" that identify each entry (#1, #2, #3, #4, etc.). You can use these elements in other LIBrarylist commands.

LIBR ADD DWARF LISTNAME=LIST1 ENTRY=(#3 ,#1, #2, #4) ← This command will reorder the entries in LIST1. The values specified in the ENTRY= operand indicate the items in the list, and the sequence they are specified indicates the order they are to be changed to.

This command reorders the structure in this way:

```
LIST    LIST1
#3 MVS  MY.DWARFLIB(SAMPLE)
#1 MVS  MY.DATA
#2 HFS  //u//fred
#4 DDN  ////MYDDNAME
```

More Info: Enter Help COmmands LIBRarylists Add. See also Help DEBugging C. See also Help DEBugging C LIBrarylists GLOSsary.



Write current LIBrarylists settings to a temporary work area, or (with RESTore) retrieve a previously stored group of settings.

CHEckpoint/CKpt Copies the "Active Instance" of the Library Lists data into a buffer located in the Session Profile. It will then be referred to as the "Saved Instance". This Saved Instance will NOT be "hardened" to disk unless you issue a PROfile Save command. See the Remarks section below.

While the size of the Active Instance is unlimited, the size of the Saved Instance is limited to 8,000 bytes. If you attempt to checkpoint an active Instance that is too large, the command will abort with a message. See Help DEbugging C Librarylists Lists Saving for more information and a suggested work-around

RESTore Restore current LIBrarylists settings with the previously checkpointed ones. If there are no previously checkpointed settings, the command will fail.

Remarks: When building the proper set of LIBrarylist definitions, it's useful to "save" your work thus far. The CHEckpoint/CPKt function copies the current state of all defined lists and redirects to (and from) a Saved Instance within the session profile. This allows you to test out different LIBrarylists configurations without losing the settings that you know work.

When a CHEckpoint is issued, it will display a message that indicates the current percentage of space that is currently in use to help you gauge how full the area is becoming.

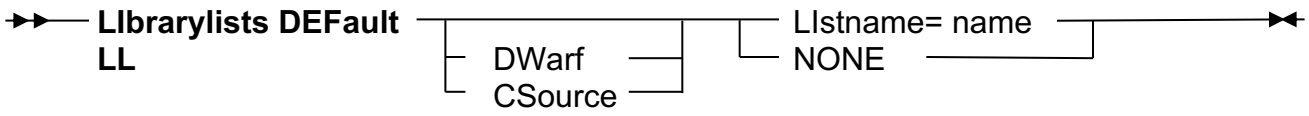
The checkpoint area is volatile and will NOT persist across debugging sessions. So, if you arrive at a set of LIBrarylists lists and redirects that you like, then you can perform a LIBrarylists CHEckpoint function and then save these settings into your Profile.

- Profile Save <blank> will save these settings into your default profile.
- Profile save <5-character-name> will save them into a named Profile.
- For debugging in batch, define an XDCPROF or ISPPROF DDNAME, and you can save your scripts into if for batch job debugging.

Your LIBrarylists lists and redirects will persist across debugging sessions when you save them into your Profile.

It's worth repeating that by far the best way to use the LIBrarylists command is through z/XDC Scripts. z/XDC scripts are sequential files filled with z/XDC commands separated by semi-colons (our command delimiter) or Return/New line. Once you've built your z/XDC script and saved it, you can invoke it in z/XDC with z/XDC's READ command, which opens the script and processes the commands in it one by one. This would maximize the ROI you put into the LIBrarylists command. See Help Scripts, Help Scripts Ryoscripts ("Rite-your-own"), and Help COmmands REAd.

More Info: Enter Help COmmands LIBrarylists Checkpoint. See also Help DEBugging C. See also Help DEBugging C LIBrarylists GLOSsary. See Help Scripts, Help Scripts Ryoscripts ("Rite-your-own"), and Help COmmands REAd.



Establish a default list for a library(ies) containing source statement datasets for later use by c/XDC.

DWarf Specifies that Librarylists will be working with a Dwarf-type library. This identifies a "Family" of DWARF datasets.

CSource Specifies that Librarylists will be working with a C Source library. This identifies a "Family" of C source files.

NONE This operand will clear the default entry.

LISTname= The (up to 8-character) name of a list of DWARF or CSOURCE lib datasets. Abbreviations for LISTNAME are LIST, LNAME or LN. Lists are created with the LIBrary Add sub-command's ENT= parameter, and are stored within the user's Profile. This identifies a Candidate List which you want to become the default.

Remarks: The Activate function sets a default list of DWARF or C Source libraries. Multiple lists may exist but only one can be the default at a time. You can clear the default list name by using the NONE parameter.

You can have many Families of DWARF or Source File libraries, each suited for a different purpose.

DWARF vs. CSOURCE: Most of the time you'll use the LIBrarylists command to tell c/XDC where DWARF resides. There are only two cases in which you'll need to specify CSOURCE in your LIBrarylists command:

1. When the C source statements have been moved after compilation, and
2. When the C source statements were originally specified to the compile step in-line to the JCL as a //SYSIN file.

Examples: LIBRARYLIST ACTIVATE DWARF LN=LIST1 ← A previously defined list of DWARF source datasets named LIST1 is established as the default.
LIB ACT CSOURCE NONE ← This command clears the default list for CSource.

More Info: Enter Help COmmands LIBrarylists DEFault. See also Help DEBugging C. See also Help DEBugging C LIBrarylists GLOSsary.

Remove unwanted lists or datasets from a LIBrarylists concatenation.

Or, to remain more in line with our Help system: Remove unwanted Result Objects from a Candidates List, an entire Candidates List or Pattern Match entries from a Redirect list.

DWarf Select a DWARF-type library ("Family") to delete lists or entries.

CSource Select a C source-type library ("Family") to delete lists or entries

LISTname A character string that identifies the list. LISTNAME can be abbreviated to LIST, LNAME or LN. LISTname may not exceed 8 characters in length.

ENTry= A parenthetical statement that contains the names of source datasets, separated by commas. When adding only a single dataset name, the parentheses are optional.

PATterns= Removes Match Patterns from a family's Redirect List. Each Pattern receives a sequence number (#1, #2, #3, etc.) and you can manipulate the patterns accordingly. But there's a lot more you can do to denote a pattern including "wild-carding" using a combination of characters, asterisks and question marks. Such patterns resolve to the names of individual members/datasets/libraries. Separate each pattern description with a comma – NOT a space.

Remarks: **DWARF vs. CSOURCE:** Most of the time you'll use the LIBrarylists command to tell c/XDC where DWARF resides. There are only two cases in which you'll need to specify CSource in your LIBrarylists command:

1. When the C source statements have been moved after compilation, and
2. When the C source statements were originally specified to the compile step in-line to the JCL as a //SYSIN file.

Wild-carding: Use "?" to match exactly one character. "???" would match three characters. Use "*" to match 0 or more characters up to the next period, open parenthesis, closing parenthesis or forward slash ("/"). Use "**" to match zero or more characters up the end of the "Original Name", the name of a dataset originally given that contains Dwarf or C source statements.

Example: The following examples presume the existence of this LIBrarylists structure:

```
 LIST THISLIST
 #1 LIST MYLIST3
 #2 LIST MYLIST1
 #3 LIST MYLIST2
 #4 HFS  /u/fred
 #5 MVS  MYFILE.XX
 #6 HFS  /u/dwarfdata
 #7 MVS  USER.Z22.DWARFLIB
```

LIBR DELETE DWARF LISTNAME=THISLIST ENTRY=(MYFILE.XX) ← This command will delete "MYFILE.XX" from THISLIST, resulting in:

```
 LIST THISLIST
 #1 LIST MYLIST3
 #2 LIST MYLIST1                    ← Note that c/XDC has re-sequenced the item numbers.
 #3 LIST MYLIST2
 #4 HFS  /u/fred
 #5 HFS  /u/dwarfdata
 #6 MVS  USER.Z22.DWARFLIB
```

More Info: Enter Help COmmands LIBrarylists DELete. See also Help DEBugging C. See also Help Debugging C Librarylist s Glosary.

Cause c/XDC to look "elsewhere" for source statement libraries, or reorder the priority of established lists.

DWarf Select a DWARF-type Family/library to create a list from.

CSource Select a C source-type Family/library to create a list from.

Patterns= Identifies one or more new or existing lists that you wish to establish the redirect to. That could be as simple as the sequence number in an existing list (#1, #2, etc.) or far more complex. Patterns denote datasets or libraries of datasets using combinations of characters and wild-card settings (discussed below) to resolve references to these datasets. Separate each pattern description with a comma – NOT a space. **NOTE:** Metal C users *must use this unique pattern*: PAT='-METALC-CSECT(<C-compilation-unit-name>)'

USE= Use may be any of the following:

USE=mvs.dataset.name
USE=mvs.dataset.name(member)
USE=mvs.&1.name(member&2)
USE=/u/rob/myfile
USE=/u/rob/my&3file
USE=listname
USE=DEFAULT ← connects the match patter to the current default candidates list.
USE=NONE ← Removes the pattern.

Remarks: Here is the "heart" of the LIBrarylists command. When your source listing has been moved, you use LIBrarylists REDirect to point to the new location of the source file. If you're using IBM's Metal C compiler, you HAVE to use LIBrarylists REDirect, so you'll probably get competent with it fairly soon. However, you might also want to create a z/XDC Script to do this donkey-work.

DWARF vs. CSOURCE: Most of the time you'll use the LIBrarylists command to tell c/XDC where DWARF resides. There are only two cases in which you'll need to specify CSource in your LIBrarylists command:

1. When the C source statements have been moved after compilation, and
2. When the C source statements were originally specified to the compile step in-line to the JCL as a //SYSIN file.

Wild-carding: Use "?" to match exactly one character. "???" would match three characters. Use "*" to match 0 or more characters up to the next period, open parenthesis, closing parenthesis or forward slash ("/"). Use "**" to match zero or more characters up the end of the "Original Name", the name of a dataset originally given that contains Dwarf or C source statements.

Example: LIBr REDirect DWarf PAT=* USE=my.dwarfdata ← This example redirects every dataset reference to a single file:

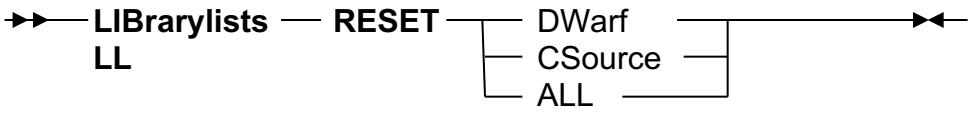
If you wanted to have something that redirects *.MYDWF1 and *.MYDWF2 to MY.DWARF1, MY.DWARF2, and MY.DWARF3 you have to use a List:

LIBR ADD DWwarf LIST=mylist ENT=(my.dwarf1,my.dwarf2,my.dwarf3) ← create the list and its entries
LIST MYLIST ← Show the list you've created
#1 MVS MY.DWARF1
#2 MVS MY.DWARF2
#3 MVS MY.DWARF3

LIBR REDirect DWarf PAT=(*.mydwf1,*.mydwf2) USE=mylist ← create the redirection, and
Pattern '*.MYDWF1' is now redirected to list 'MYLIST'
Pattern '*.MYDWF2' is now redirected to list 'MYLIST'

For Metal C: LL RED PAT='-METALC-CSECT(<your-compilation-unit)' USE=<dataset-containing-the-DWARF-for-this-compilation-unit>

More Info: Enter Help COmmands LIBrarylists REDirect. See also Help DEbugging C. See also Help DEbugging C Librarylists Glossary.



Clear all redirects and delete lists for a specific type of library, or for all of them.

DWarf Select a DWARF-type Family/library to purge.

CSource Select a C source-type Family/library to purge.

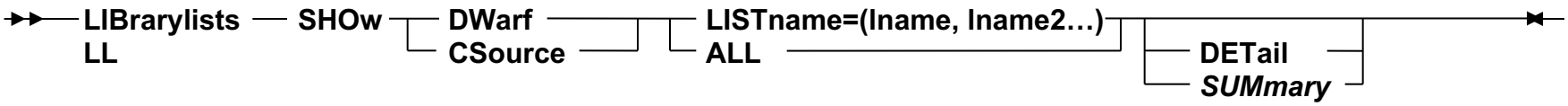
ALL ALL Families, their Redirect lists and all of their Candidate lists will be deleted from the Library Lists. In other words, blow EVERYTHING away.

Remarks: LL RES ALL is an excellent command to issue as a “sanity check” when working with the Librarylists command. It would be an EXCELLENT idea to include as the first statement in any z/XDC Script that you write to streamline the Librarylists command process.

Examples: LIBRARYLIST RESET ALL ← Clears all lists and redirects for all libraries. Just get rid of EVERYTHING.

LIBRARYLIST RESET DWARF ← Clears all lists and redirects for DWARFLIB. Other libraries are left untouched.

More Info: Enter Help COmmands LIBrarylists RESet. See also Help DEBugging C. See also Help DEBugging C LIBrarylists GLOSsary.



Display LIBrarylists concatenations, with an optional detail report.

DWarf Select a DWARF-type Family/library to display.

CSource Select a C source-type Family/library to display.

LISTname= A character string that identifies the list. LISTNAME can be abbreviated to LIST, LNAME or LN. LISTname may not exceed 8 characters in length. If you specify more than one list name then you must separate them by commas and enclose the group of list names in parentheses.

ALL Display all lists.

DETail Causes LIBrarylists to display individual items within a list. That, apparently, is called a "Candidates List". To become unconfused, see Help Debugging C LIBRarylists Glossary.

SUMmary (or omitted) Opt not to see the contents of the Candidates List.

Remarks: Think of this as a "List LIBrarylists command". Next to LL Redirect, this may be the most useful of the LIBrarylist command variants.

DWARF vs. CSOURCE: Most of the time you'll use the LIBrarylists command to tell c/XDC where DWARF resides. There are only two cases in which you'll need to specify CSource in your LIBrarylists command:

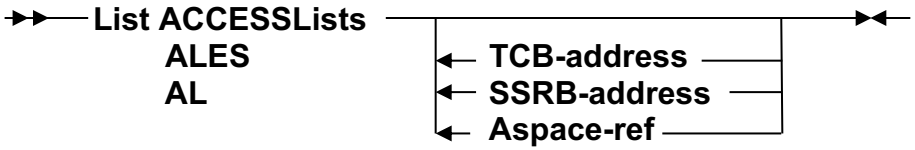
1. When the C source statements have been moved after compilation, and
2. When the C source statements were originally specified to the compile step in-line to the JCL as a //SYSIN file.

Examples: LIBR SHOW DWARF ALL ← Displays just the list names and any redirects for the DWARF library type.

LIBR SHOW DWARF ALL DETAILS← All lists, their entries and all redirects are displayed for the DWARF library type.

LIBR SHOW CSOURCE LI=(LIST1,LIST2) ← Detail entries for LIST1 and LIST2 will be displayed for the CSOURCE library type.

More Info: Enter Help COmmands LIBRarylists SHow. See also Help DEBugging C. See also Help DEBugging C LIBrarylists GLOSsary.



Produce a detailed display of any one or more access lists (PASN-ALs and or DU-ALs) associated with any task, SRB routines and/or address spaces, system-wide.

TCB-address The address of any Task Control Block in the system given as a hexadecimal address. Pro-tip: After you issue the LIST TASKS command for any address space you may use the automatic equates generated by that command in this command – i.e. TCB#1, TCB#2, etc. Thus, you could then issue LIST ACCESSLISTS TCB#2, and so forth.

SSRB-address The address of any Suspended Service Request Block (SSRB, which resides below the 2GB bar) or Suspended Service Request Block Extension (SSRX, which resides above the 2GB Bar).

Aspace-ref The PASN-AL Access List for the referenced address space is displayed. You may specify the address of the address space's Address Space Control Block (ASCB), it's Address Space Identifier (ASID), the address space's job-name, a keyword such as HOME, SASID, etc. See HELP COMMANDS SYNTAX ASIDS for more information

Remarks: The use of this command requires that z/XDC be running in Authorized mode. Any number of references may be given (TCB, SRB, Aspace-ref). When the command is issued without parameters two or three access lists are displayed. This includes the DU-AL for the task or SRB being displayed, the DU-AL for the home TCB and the home address space's PASN-AL. In addition, this command produces several automatic equates, such as ALE#n (the Access List entries that make up the list), ASTE#n (the ASN Second Table Entries) and #dspacename (identifiers for each of the dataspaces that are accessible to z/XDC).

Examples: LIST ACCESSLISTS TCB#3 ← Shows the Access Lists for any given Task Control Block
LIST ACCESSLISTS 7DE7A8 ← Shows the Access Lists for a TCB when its hexadecimal number is specified.
LIST AL JES2 ← shows the Access List for the JES2 job.

More Info: Enter Help COmmands LISt ACCESSList

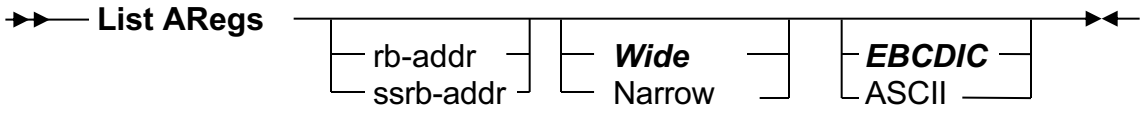
List AMode

Display the addressing mode found in the retry-level Program Status Word.

Remarks: z/XDC will return the value “24-”, “31-“ or “64-bit addressing” as appropriate. Addressing mode information is also available via the List PSw Format command.

Examples: L AM

More Info: Enter Help COmmands LISt AMode.



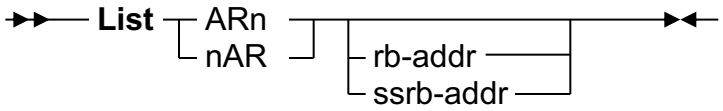
Display the contents of the Access Registers.

rb-addr	An address expression that resolves to a specific Request Block queued from any TCB in any accessible address space in the system.
ssrb-addr	An address expression that resolves to a SSRB or SSRX for any SRB routine currently suspended in any accessible address space in the system.
Wide	z/XDC will show 8 words of data in the resulting display. This is the default.
Narrow	z/XDC will show 4 words of data in the resulting display.
EBCDIC	z/XDC will interpret the contents of the Access Registers as EBCDIC information, shown in the far right of the display. This is the default.
ASCII	z/XDC will interpret the contents of the Access Registers as ASCII information, shown in the far right of the display.

Remarks: This command displays the full set of e retry-level Access Registers (List EARegs returns the error-level Access Registers). All registers are displayed in hex-text (dump-like) format.

Examples: L AR ← Shows the contents of the Access Registers.

More Info: Enter Help COmmands LISt ACCResregisters Registerset. See also Help COmmands LISt ACCESSRegisters, and Help COmmands LISt ACCESSRegisters Individual.



Display the contents of a particular Access Register for the retry-level environment.

n n is the number of the desired Access Register and ranges from 0 to 15. z/XDC will accept “nAR” as well as “ARn”.

rb-addr An address expression that resolves to a specific Request Block queued from any TCB in any accessible address space in the system.

ssrb-addr An address expression that resolves to a SSRB or SSRX for any SRB routine currently suspended in any accessible address space in the system.

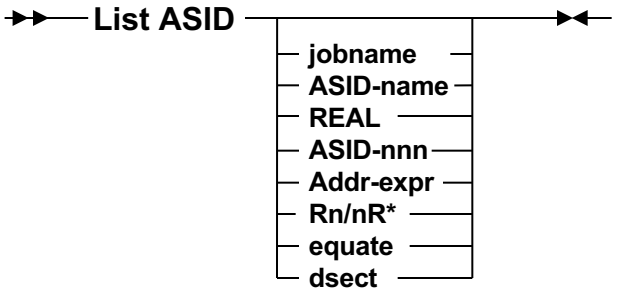
z/XDC displays the result in several ways:

- As a raw hexadecimal number,
- As a 4-byte signed fixed-point decimal number
- As a 4-byte floating-point decimal number.

Remarks: This command may not be used in Foreign Address Space Mode. To view an Access Register for the error-level environment, use List EARn or LIST nEAR.

Examples: L AR3 ← Shows the contents of Access Register number 3.
Issue LIST TASKS, then select the TCB you’re interested in. Issue LIST RBS <TCB#n> then issue LIST ARn for RB#n in that TCB#.
The above example takes advantage of the TCB# equates that are automatically generated aft the LIST TASKS command and the RGB# equates that are automatically generated after the LIST RBS command.

More Info: Enter Help COmmands LISt ACCessregisters Individual.



Display the Address Space ID of an address space.

Jobname	A jobname, such as “JES2”.
ASID-name	An identifier such as “HOME”, “PASID”, “ESASID” “IASID”, etc.
REAL	The keyword “REAL”, indicating real storage.
ASID-nnn	The number of an address space - from the Address Space Vector Control Table.
addr-expr	An address expression that points to an ASID number.
Rn/nR*	A register (expressed as Rn or nR) with an indirect operator (“!”, “?” or “%”) that resolves to an ASID number.
equate	An equate name that has been assigned to represent storage in an address space.
dsect	A dsect that has been assigned to represent storage in an address space.
Omitted	When no parm is given, the current target address space is displayed.

Remarks: This command displays the ASID number, the jobname, the name of the owning subsystem and the JES2/JES3 job-type and number.

Examples: LI AS JES2 ← display the Address Space ID for the JES2 jobname.

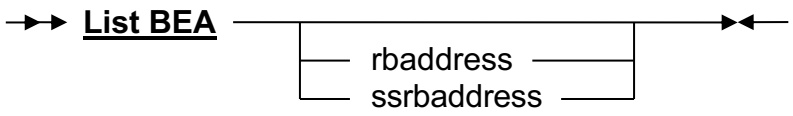
More Info: Enter Help COmmands LISt ASID. See also Help COmmands Syntax ASids.

List AUtomap

Determine whether (or not) z/XDC will automatically attempt the automatic mapping of Assembler, or C programs.

Remarks: This command accepts no parameters. You may also issue “SET AUTOMAP ON/OFF from the command line in a z/XDC session. This setting may also be accessed in the PROFILE menuing system under PROFILE/Display/Change AUTOMAP PRINT and Other Settiings.

More Info: Enter Help COmmands LISt AUtomap. See also Help COmmands SEt AUTOmap.



Display the Breaking Event Address.

rbaddress The address of any Request Block, queued from any Task Control Block, located in any accessible address space in the system.

ssrbaddress The address of any suspended Service Request Block (SSRB, residing below the 2GB bar) or suspended Service Request Block Extension (SSRX, residing above the 2GB bar) in any accessible address space in the system.

omitted The current retry-level Breaking Event Address for the program being debugged is displayed (current Task Control Block, current Request Block).

Remarks: LIST BEA displays the location of the most recent branching instruction. You may use the LIST RBS and LIST SSRBS command and then reference the automatic equates that z/XDC generates as parameters for this command (RB#n or SSRB#n).

This command is HUGELY COOL for debugging wild branches. When BEA support is enabled on the processor, the Breaking Event Address field can be found in the SDWBEA field in the SDWARC4 section of the SDWA (the System Diagnostic Work Area). But you don't have to know or care about this. z/XDC just displays it on command.

To see if the Breaking Event Address is available on your processor, issue z/XDC's LIST FEATURES command.

If the BEA feature is enabled for the processor, then the BEA will also be displayed when the LIST BRANCHES command is invoked.

Examples: LIST BEA ← Shows the address that branched to "here".

More Info: Enter Help COmmands LISt BEA. See also Help Commands LISt BRAanches. See also Help COmmands LISt Features. See also Help COmmands LISt EBea (yes, there's a List BEA for the error level program as well).

List BFRn

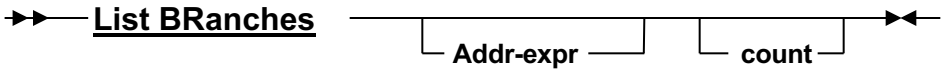
Display the entire 8 bytes of an individual floating point register.

n The number of a floating point register (1-16).

Remarks: This command will display the floating point register in raw hex-EBCDIC or hex-ASCII as well as binary floating point, both short and long.

Examples: LIST BFR10 ← Shows the contents of the tenth floating point register

More Info: Enter Help COmmands LIS BFR#.



View the contents of z/XDC's Branch History Table.

To be more in agreement with the Help System: The list branches command displays a list of recent user-program machine instructions that a) have been executed by the user program, b) are branch-type instructions and c) that z/XDC KNOWS about.

addr-expr A "filter" that, when given, includes branch history table entries only with this branch-from or branch-to address (in the target address space).

count A decimal number in the range from 1 to 4096 that sets the upper limit of Branch History Table entries to be displayed. If outside this range, the number will be treated as an address expression. If TWO counts are given, the second will be parsed as an address expression, not a count.

Remarks: **In order to work, your processor must have BEA support enabled.** You may issue the LIST FEATURES command to find out if BEA is enabled.

Every time that z/XDC gets control, it checks to see whether or not the user program's current retry level instruction is a branch-type instruction. If so, then an entry is added to its Branch History Table. NOT every branch that the user program executes will be recorded in the Branch History Table. Only those branches at which z/XDC receives control will be recorded.

Note: If the Breaking Event Address field is available for the current processor, then a LIST BEA command will automatically be included in the output of LIST BRANCHES.

This command may not be used in Foreign Address Space Mode.

Examples: *LI BR mymod.mycsect.myloop* ← *investigates all branches from, or to this point* in the program.
LI BR ← show all branch statements at which z/XDC received control.

More Info: Enter Help COmmands LISt BRAnches.

Display all of the breakpoints that are currently defined for the z/XDC session.

This command accepts no parameters.

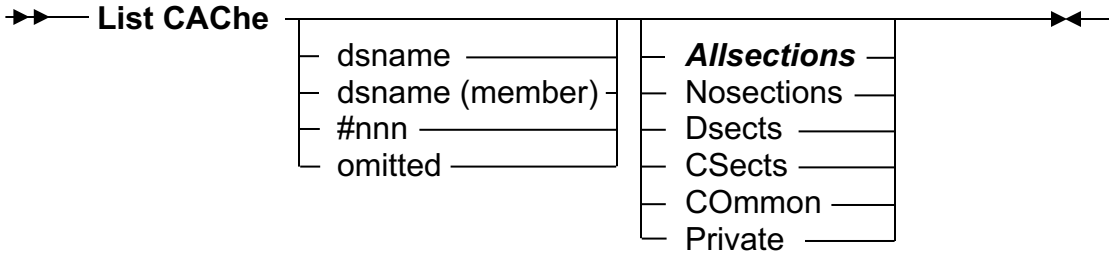
Remarks: The List Breakpoints command displays the following information about all currently defined breakpoints:

- The breakpoint's unique name;
- Location, both as a raw virtual address and as relative to the nearest relevant load module, map or equate (if any);
- The associated conditional expression, if any;
- Associated automatic z/XDC commands that are associated with the breakpoint, if any;
- Whether or not the breakpoint is enabled or disabled (via the "X" line command);
- The number of times the breakpoint has been "hit". A zero ("0") indicates that the breakpoint has never been taken.
- For deferred breakpoints, the number of times the breakpoint has been cloned, and the number of cloning errors (if any).
- Whether or not execution location displays will roll or scroll during tracing (SET TRACE SCROLL keeps the current instruction at the top of the display).

This command may not be used in Foreign Address Space Mode, because z/XDC is not in control of any program when in FASM. You get "look-don't-touch" access.

Examples: L B

More Info: Enter Help COmmands LISt BReakpoints.



Display the list of ADATA source information that z/XDC has cached for later use.

dsname/dsname(member) Information about a particular dataset name. If the cache file is associated with a sequential dataset, then the dsname form must be used. If the cache file is associated with a member of a partitioned dataset then the dsname(member) form must be used.

#nnn The unique decimal number that was assigned to a cache dataset in a previous List CAChe command.

Omitted Information for all cached files is displayed.

Allsections This is the default. Displays csect, dsect, common blocks and private areas described by the cached ADATA information

Nosections Suppresses the display of all section names, and shows only the names of the MAPLIBS that own the cache files.

Dsects/Csects/Common/Private Limits the scope of the display to cached information in these areas**.**

Remarks: IBM's implementation of ADATA results in HUGE amounts of storage consumption. Thus it is possible that the presence of cached data may cause a storage shortage. In such cases it would be a good idea to re-run the job with REGION=0M.

Cole Software also offers an Adata filter program that can reduce the amount of disk storage consumed. See Help Maps AData Excessadata for more information on how to reduce the disk "footprint" of the programs for which you want source statement support.

As to the command itself: Any combinations of operands may be given in any order. Operands from the first column allow you to limit the display to only those cache files owned by particular MAPLIB datasets. Operands from the second column allows you to limit the display to only particular ESD types.

Examples: L CAC CO Shows the list of cached ADATA information in the Common Area.
L CAC Shows the entirety of the cached ADATA information.

More Info: Enter Help COmmands LISt CAChe. See also Help COmmands DElete Cache.

List CANdet

Displays whether (or not) Cancelled or Detached programs may be debugged.

This command accepts no parameters.

Remarks: Normally, there is no point in debugging programs that experience termination abends (cancelled jobs – x22 abends or subtasks that are detached – x3E abends). For this reason, z/XDC's default is "SET CANDET IGNORE". You may, however over-ride this choice with "SET CANDET DEBUG".
You may also change the CANDET value in a z/XDC Profile. You may issue PROFILE, then navigate to the third category from the bottom of the list: Display/Change AUTOMAP PRINT and Other Settings.

More Info: Enter Help COmmands LISt CANdet. See also Help COmmands SEt CAndet.

List CDf

Displays the status of the Cross Domain Facility environment for the current debugging session.

This command accepts no parameters.

Remarks: Cole Software's Cross Domain Facility allows the user to debug programs running in other address spaces. Thus, "CDF" is a critical resource within the product. This command shows whether or not CDF is installed, whether it is available to the current debugging session, how CDF will notify the system when a program awaits debugging connection and the amount of time CDF will "wait" for a user to connect to that session.

More Info: Enter Help COmmands LIS CDf. See also Help COmmands SEt NOWtor, Help COmmands SEt WTor and Help COmmands SEt SIGNonwait.

List COLORs
COLOurs

Display how 3270 colors are assigned.

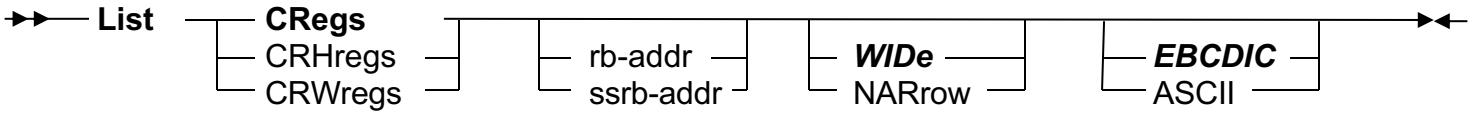
This command accepts no parameters.

Remarks: There are four 3270 fields displayed, and you can manipulate the color attributes for each of them with z/XDC's SET COLORS command.

These are:

- Low Intensity Input fields ← Storage displays
- High intensity input fields ← Command lines
- Low intensity protected fields ← Non-over-writable information
- High Intensity protected field ← Titles, headers, warning messages, etc.

More Info: Enter Help COmmands LISt COLORs. See also Help COmmands SEt COLORs.



Display the set of Control Registers.

CRegs	Display the Control Registers. In 64-bit addressing mode, this will show only the low-order half of the 64-bit Control Registers.
CRHregs	In 64-bit addressing mode, this will show only the high-order half of the 64-bit Control Registers.
CRWregs	In 64-bit addressing mode, this will show the entirety of the 64-bit Control Registers.
rb-addr	The address of a Request Block that is currently queued from any TCB located in any accessible address space anywhere in the system.
ssrb-addr	The address of a suspended Service Request Block or an SSRX (an extension of the SSRB that resides in 64-bit storage).
WIDe	Default width. z/XDC will display eight words of data per line in its display.
NARrow	z/XDC will display four words of data per line in its display.
EBCDIC	Default. The display will be shown in EBCDIC format.
ASCII	The display will be shown in ASCII format.

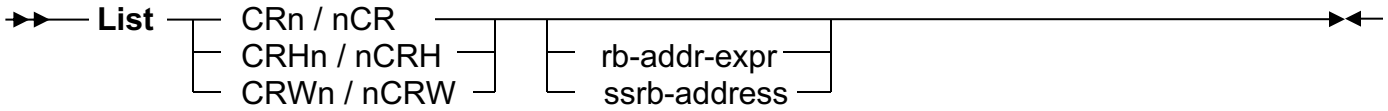
Remarks: Control registers are used for controlling a large number of fundamental characteristics of the computer's hardware. You can learn more about them in IBM's Principles of Operation manuals (SA22-7832 in z/OS).

There is a set of control registers for every execution thread, whether the code runs in Task-mode, or SRB-mode – but not for each individual running program.

In some cases, the contents of all or part of a control register are not knowable. In this case the "unknowable" fields will be displayed as dashes. See z/XDC's Help Commands LISt CONtrolregisters for Dave's treatment of this subject.

While it is possible for z/XDC to alter General Registers, Floating Point Registers, and Access Registers, it is not possible to alter Control Registers. Dave can't think of a good reason why you should need or want to do that. Upon hearing this, most of our users nod sagely in agreement.

More Info: Enter Help COmmands LISt CONtrolregisters Registerset. See also Help EXEcutionlevels.



Display the contents of an individual Control Register.

n n is the number of the desired Control Register and ranges from 0 to 15

CR If z/XDC detects a 64-bit environment, this parameter will display only the low-order half of the Control Register.

CRH If z/XDC detects a 64-bit environment, this parameter will display only the high-order half of the Control Register.

CRW If z/XDC detects a 64-bit environment, this parameter will display the entirety of the Control Register.

rb-addr-expr if an address-expression is given, it must resolve to the address of a Request Block that is currently queued from any Task Control Block located in any address space on the system. If you have previously issued the LIST RBS command for any particular Task Control Block, then the RB#n equates generated by LIST RBS are excellent parameters for LIST CRn.

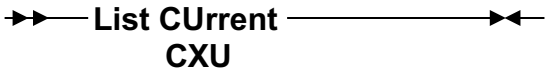
ssrb-address The address of a Suspended Service Request Block SSRX (an extension of the SSRB that resides above the bar in 64-bit storage) for any SRB routing currently suspended in any accessible address space anywhere in the system. If you have previously issued "LIST SSRBs" command, then you may use the SSRB#n and SSRX#n equates for this parameter.

Remarks: The slash above ("/") indicates that z/XDC will accept either form of the command. When issued, z/XDC produces a comprehensive display of the contents of the desired Control Register, formatted out field by field for your convenience.

In some cases, the contents of all or part of a control register are not knowable. In this case the "unknowable" fields will be displayed as dashes. See z/XDC's Help Subsystem for Dave's treatment of this subject.

Examples: L CR4 ← Display the contents of Control Register number four.
L CR2 RB#2 ← Display the contents of Control Register number two for Request Block #2 under the current task (I used LIST TASKS, then LIST RBS to get the RB#2 equate).

More Info: Enter Help COmmands LISt CONtrolregisters Individual. See also Help Command LISt CONTrolregisters.



Display information about the current execution thread.

This command accepts no parameters.

Remarks: With the advent of Cole Software's dump/XDC product, the concept of a "Current Execution Unit" was created. This is also referred to as the "CXU" in our parlance. LIST CURRENT displays the Task-mode or SRB-mode thread that "abended", thereby passing control to our products.

In dump/XDC one can also alter the CXU to wherever one desires, if the user determines that dump/XDC has mistaken the CXU's placement in storage. There's even a shortcut command to do it ("CU").

More Info: Enter Help COmmands LISt CURRENT. See also Help COmmands SEt CUrrent

List CXDC

Displays the availability of the c/XDC Feature.

This command accepts no parameters.

Remarks: LIST CXDC shows whether the c/XDC Feature is licensed, whether XDC's Server job is running and whether the Server is running c/XDC's support task. If c/XDC is NOT working, this command reports that, as well. One may also set the c/XDC Feature on or off (but I don't know why you'd WANT to do that).

More Info: Enter Help COmmands LISt CXdc. See also Help COmmands SEt CXdc.

List DDntrans

Display a list of currently defined ddname translation tokens.

This command accepts no parameters.

Remarks: This command is specific to dump/XDC. Ddname translation tokens are used to translate task library ddnames found in a dump to LIBn ddnames found in Module Mapping Extraction files.

More Info: Enter Help COmmands LISt DDntrans. See also Help COmmands DEFine DDntrans. See also HELP DUMPS. Information on Cole Software's Module Mapping Extraction Files (the XDCMMEF Utility) may be found here: https://www.colesoft.com/support/module-extract-utility.

List DFR#

Display the entire 8 bytes of an individual floating point register in decimal floating point format.

Any one of the 16 floating point registers 0 – 15.

Remarks: This command shows the target floating point register in raw hex-EBCDIC or raw-ASCII formats and in Decimal Floating point formats, both short and long.
Note: z/XDC's view of the world is EBCDIC-centric. Therefore, if you NEED to see a floating point's contents in ASCII, you must issue SET FORMAT ASCII. Later you can always issue SET FORMAT EBCDIC to get it back to the other format.

Examples: Li DFR3 ← Shows the contents of floating point register two

More Info: Enter Help COmmands LIST DFr#. See also Help COmmands LIST FLOATingpointregisters. The LIST FR# will show the contents of a floating point register in decimal format, but also binary format and hexadecimal format.

List DSEctlibs

Display a list of all known DSECT library “tokens” that have been created for use by dump/XDC.

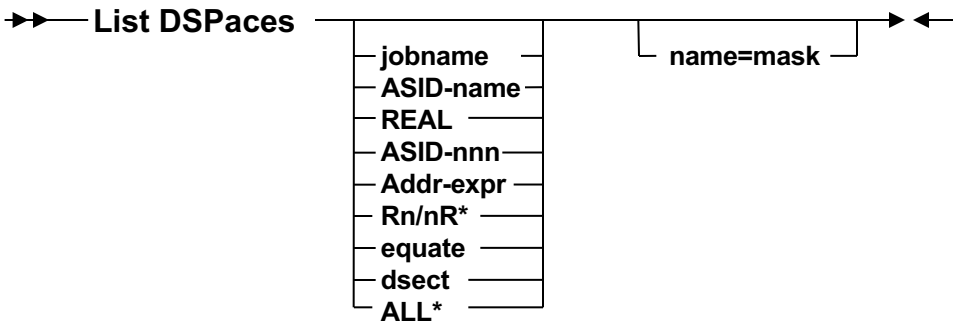
This command accepts no parameters.

Remarks: When working in dump/XDC upon a dump that was generated OUTSIDE the local environment (a customer or colleague on another system), we often recommend that the downstream customer/entity run the free-ware utility XDCMMEF which resides on our colesoft.com website at: https://www.colesoft.com/support/module-extract-utility. XDCMMEF extracts useful information from the sender’s system for incorporation into the dump by dump/XDC.

Subsequent to that, it may be necessary to define DSECTLIB tokens that attach these libraries to the dump for processing by dump/XDC. LIST DSECTLIBS allows you to see a list of what you have previously defined to dump/XDC.

Examples: LI DSE ← A list of DSECTLIB tokens will be displayed, if any exist.

More Info: Enter Help COmmands LISt DSEctlibs. See also Help COmmands DEFine, and Help COmmands DEFine DSectlib. For general information on dump/XDC, please see Help DUmps.



Display the address spaces owned by specified address space(s).

Jobname	A jobname, such as “JES2”
ASID-name	An identifier such as “HOME”, “PASID”, “ESASID” “IASID”, etc.
REAL	The keyword “REAL”, indicating real storage.
ASID-nnn	The number of an address space - from the Address Space Vector Control Table.
addr-expr	An address expression that points to an ASID number.
Rn/nR*	A register (expressed as Rn or nR) with an indirect operator (“!”, “?” or “%”) that resolves to an ASID number.
equate	An equate name that has been assigned to represent storage in an address space.
dsect	A dsect that has been assigned to represent storage in an address space.
ALL*	The asterisk is NOT significant! It is merely to draw one’s attention to the fact that this parameter is ONLY for use with dump/XDC.
Omitted	When no parm is given, the current target address space’s list of address spaces (if any) are displayed.
Name=mask	Limit the display to selected dataspace names. You may use ? and * wildcard characters.

Remarks: The List DSPaces command displays a list of some or all dataspaces owned by a specified (and accessible) address space. The dataspaces are displayed in alphabetic order by dataspace name.

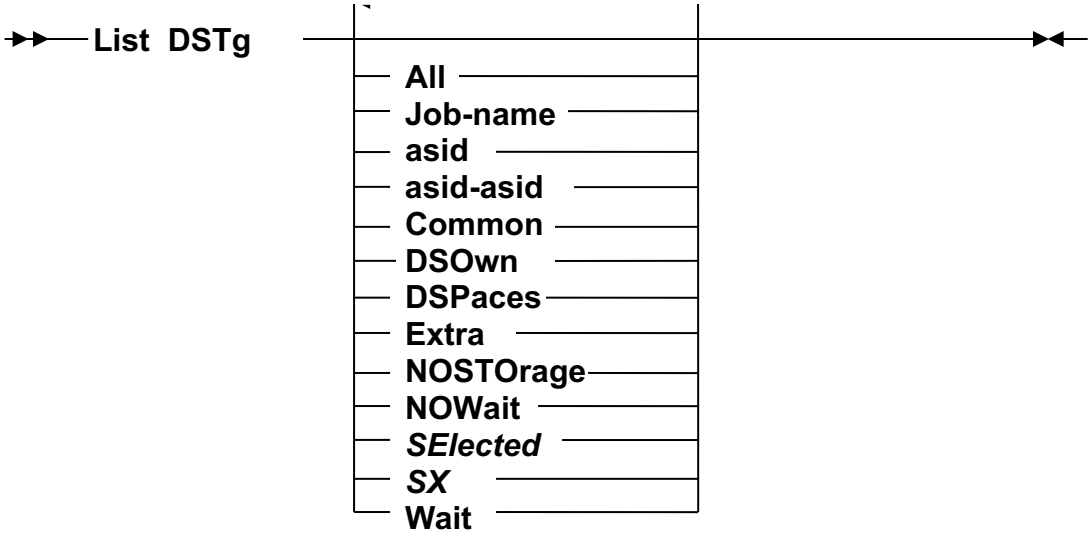
A wildcard mask (“*” or “?”) can be given to limit the display to only selected dataspaces. For information on masks, see Help Commands Syntax Masks, but (to save you time), a question mark matches any single character in a dataspace name and an asterisk matches zero, one or more characters in a dataspace name.

For each data space, the command displays the dataspace’s name, its type, whether it is public or private, the number of authority table entries defined for it (up to the first 32 entries) and uniquely numbered equates of the form “ASTE#n” (as in “ASN Second Table Entries”) that define the dataspaces and anchor their page tables. If a prior set of ASTE#n equates exist, then they are replaced the next time you issue the LISt DSPaces command

Note: Under dump/XDC, this command can be used to obtain a list of the dataspaces (and their owning ASIDs) actually present on the dump. The parameter “ALL” may only be used in the context of dump/XDC.

Examples: LI DSPACES ← Display all address spaces “owned” by the current address space.
LI DSPACES JES2 ← Display address spaces owned by the system’s JES2 job.

More Info: Enter Help COmmands LISt SSPaces. See also Help COmmands Syntax Masks.



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

Omitted	If no parameters are specified, the command acts as if "COMMON" and "SX" were specified.
All	Shows storage for all address spaces 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 an ASID number instead of a job-name.
asid	A single asid number; a 4-byte hexadecimal number from the Address Space Vector Control Table. Typically these will be hexadecimal numbers but they may also be decimal numbers (expressed with a suffix of "N"). Example: For JES2 you could use the hex number "1E" – OR – "30N".
asid-asid	Displays storage allocations for a range of address space identifiers.
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 if 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	If dump/XDC's information-gathering routine has not completed when this parameter is entered, then the command will wait for it to complete
Remarks:	This command will display ONLY the storage allocations that are present in the dump. If you want to display information about a jobname that has the same name as one of the 'LIST DSTG' parameters (for example, a jobname of 'SX'), then use that job's ASID instead (for example, if jobname 'SX' had an ASID of 0145 (hex) then use 0145) 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 COMMON 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 DSTg.

List Dump

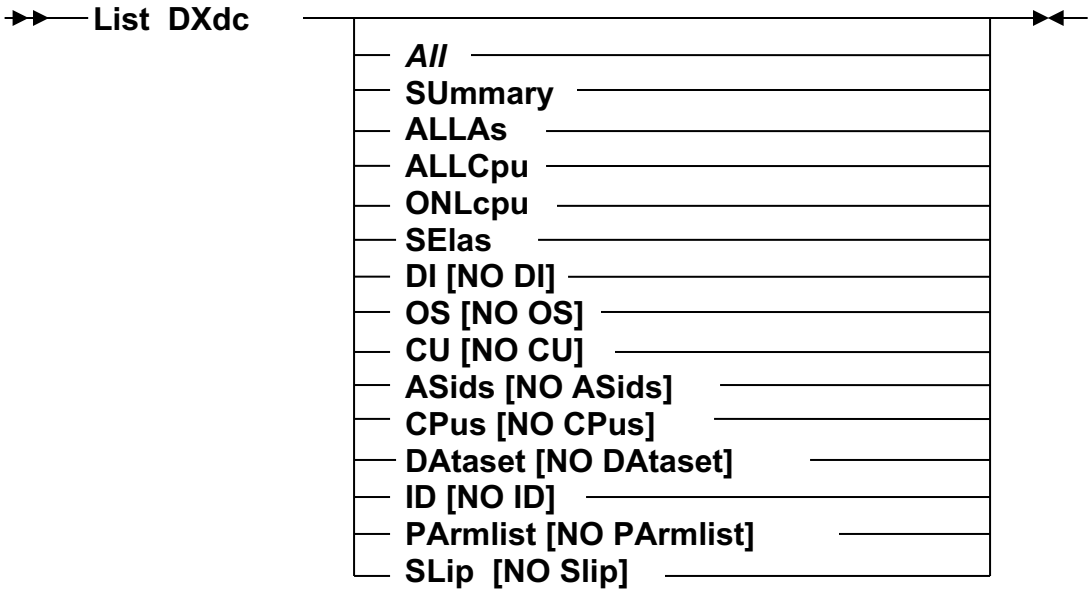
Display the current setting of the SET DUMP command.

This command accepts no parameters.

Remarks: The List Dump command merely displays the “state” of dump processing within z/XDC (not outside of z/XDC).

To expand, the Set Dump command may be used to enable or disable the possibility of producing a system dump in the event that an external service called by z/XDC fails with an ABEND. The normal “state” of Set Dump is “DISABLE”.

More Info: Enter Help COmmands LISt Dump. See also Help COmmands SEt Dump.



Display (or limit) information about the current dump environment in a dump/XDC session.

All	Default. Displays the entire output 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.
ALLAs	Displays all address spaces available in the dump.
ALLCpu	Displays all defined CPUs.
ONLcpu	Displays all online CPUs.
SElas	Displays "Selected" address spaces
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. Pro-tip: You can use the "CU" short-cut command on one of the lines of output from this command to directly change the CXU manually.
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.

⟶⟶— List EAR# —⟶⟵

Display the contents of a particular error-level Access Register.

n is the number of the desired error-level Access Register and ranges from 0 to 15. z/XDC will accept “nEAR” as well as “EARn”.

z/XDC displays several items as a result of this command:

- The Access Register’s value both as a raw hexadecimal number and in EBCDIC formats;
- The address space that “owns” it;
- The name of the address space;
- The type of the address space (such as “TSO”);
- The Address Space ID, given as a hex number (from the Address Space Vector Control Table).

Examples: L EAR1 Display the contents of error-level Access Register number one.

More Info: Enter Help COmmands LISt ACCESSRegisters Individual. For more information on the “Error-level” structures within z/XDC, enter Help EXEcutionlevels.

List EARegs

Display the set of Error-level Access Registers.

This command accepts no parameters

Remarks: The List EAREgs command displays the full set of sixteen error-level Access Registers. By “error-level” we mean the Access Registers that were being used by the problem program (or a subroutine or system service routine called by the program) at the moment that an abend occurred.

More Info: Enter Help COmmands LISt ACCESSRegisters. A reference may also be found in Help COMmands LISt ACCessregisters Registerset. For more information on the “Error-level” structures vs. the retry-levels within z/XDC, enter Help EXEcutionlevels.

List EBEA

Display the Error Level Breaking Event Address.

This command accepts no parameters.

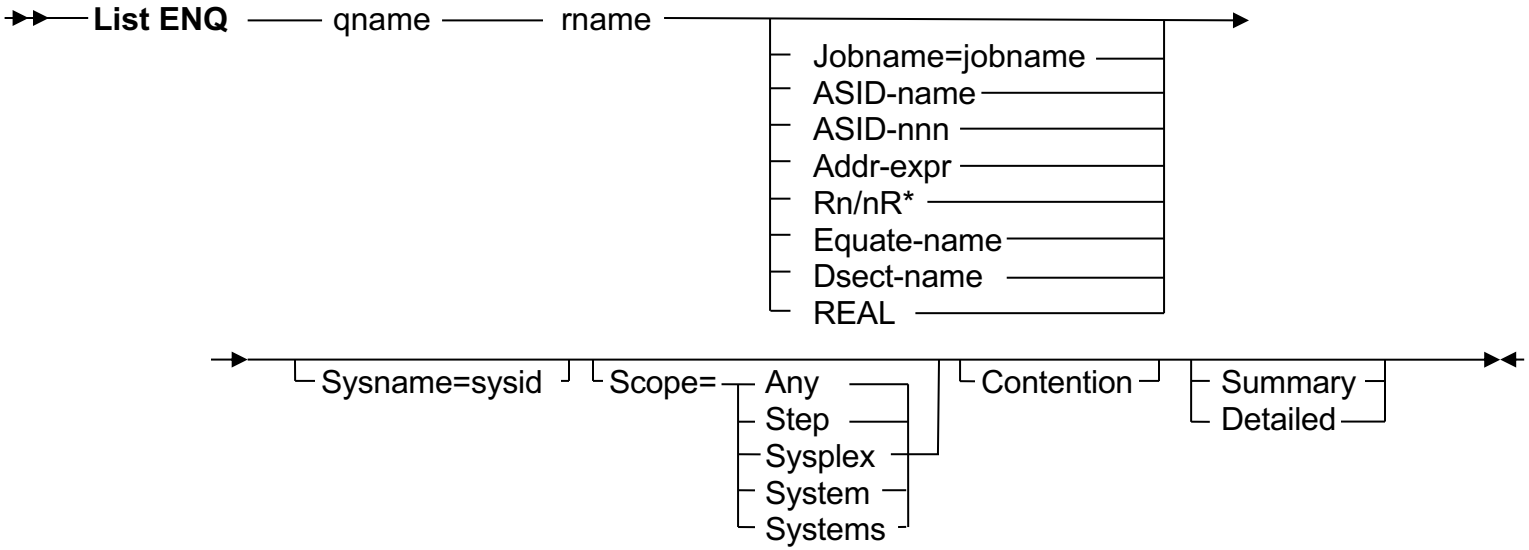
Remarks: When entered, it shows the Breaking Event Address for the Error Level environment - BUT – only when the error level and retry level environments are DIFFERENT from one another. For a good reference on the Error Level vs. the Retry Level environments, see Help EXEcutionlevels.

z/XDC maintains an automatically-generated equate for you called @EBEA.

If the BEA feature is enabled for the processor, then the BEA will also be displayed when the LIST BRANCHES command is invoked.

Examples: LIST EBEA ← Shows the address that branched to “this place” in the error level environment.

More Info: Enter Help COmmands LISt EBEA. See also Help Commands LISt BRAnches. See also Help COmmands LISt Features. See also Help COmmands LISt BEA.



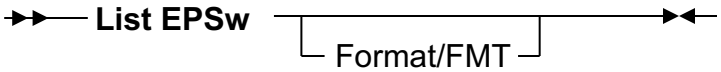
Display a list of system and user enqueues, including contention amongst enqueues in either Summary, or Detailed mode.

qname or rname	*When any parms are stated then qname or rname are required.* You may use wildcard characters ("*" and"?"), single quotes, etc. See Help Commands Syntax Masks (and Help Commands Syntax Patternmatch) for more information, but to save you time, a "?" can substitute any single character in a name. An"*" can substitute zero-to-many characters.
jobname	A jobname, such as "JES2"
ASID-name	An identifier such as "HOME", "PASID", "ESASID" "IASID", etc.
REAL	The keyword "REAL", indicating real storage.
ASID-nnn	The number of an address space - from the Address Space Vector Control Table.
addr-expr	An address expression that points to an ASID number.
Rn/nR*	A register (expressed as Rn or nR) with an indirect operator ("!", "?" or "%") that resolves to an ASID number.
Equate-name	An equate name that has been assigned to represent storage in an address space.
Dsect-name	A dsect that has been assigned to represent storage in an address space.
sysname=sysid	The name of one or more systems connected in the current sysplex. If omitted, all systems are reported on.
scope=	Any: The report is not filtered as to "scope". This is the command's default. Step: The scope of the report is confined to the address space within which the issuing address space is running. System: The scope of the report is confined to the system within which the issuing address space is running. Sysplex/Systems: The enqueue's scope extends across the entire sysplex. A "synonym" for "Sysplex" is "Systems".
Contention	Shows only enqeueues which are in a state of contention.
Summary/detailed	Control the level of detail produced in the command's output. There is a very useful description between the two modes in Help COmmands LISt ENq.

Remarks: This command displays the sysid, the asid, the jobname, the type (shared or exclusive), the qname, rname and rank.

Examples: LI ENQ SYSDSN * ← Displays all opened datasets.

More Info: Enter Help COmmands LISt ENq. See also Help Commands Syntax Masks for an overview of using the "*" and"?" wildcard characters.



Display the Error-level Program Status Word in the older 8-byte format.

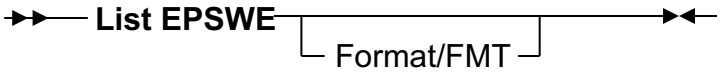
Format/FMT Causes z/XDC to translate various fields in the PSW for the user. These fields are:

- The PER bit;
- Dynamic address translation (ON/OFF), I/O (ON/OFF), External interrupts (ON/OFF);
- Protection Key;
- Problem state/Supervisor state;
- Address Space Control Mode (Primary, Access Register, Secondary, Home);
- Fixed point overflow (ON/OFF), Decimal Overflow (ON/OFF), Exponent underflow (ON/OFF), Significance (ON/OFF);
- Addressing mode (24/31-bit);
- The instruction Address.

Remarks: The difference between the "error-level" and "retry-level" environments is one of the fundamental concepts in z/XDC. The error-level environment consists of the PSW, register settings and the contents of certain control blocks that existed at the time of the abend. The retry-level environment is the same group of settings, but are the ones that will be used at retry time (A GO, a TRACE or TRAP command – any resumption of the program). Most of the time these environments are, and remain the same. But when these two environments are different, Dave flags it and announces it in the beginning of a z/XDC Session ("The Error level and retry level environments are DIFFERENT."). In that case you may have a recursive abend and a debugging scenario that is a bit more complex than you expect. The LIST EPSW command allows you to see the error level PSW. Conversely, the LIST PSW command shows you the retry-level PSW. For Dave's excellent discussion on this topic, see Help EXEcutionlevels.

Examples: L EPS F ← Display the Program Status Word with the above fields "broken out".

More Info: Enter Help COmmands LISt EPsw.



Display the Error-level Program Status Word in the more recent 16-byte format.

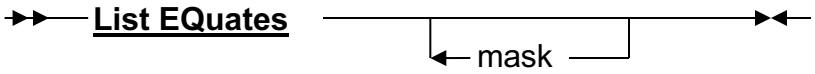
Format/FMT Causes z/XDC to translate various fields in the PSW for the user. These fields are:

- The PER bit;
- Dynamic address translation (ON/OFF), I/O (ON/OFF), External interrupts (ON/OFF);
- Protection Key;
- Problem state/Supervisor state;
- Address Space Control Mode (Primary, Access Register, Secondary, Home);
- Fixed point overflow (ON/OFF), Decimal Overflow (ON/OFF), Exponent underflow (ON/OFF), Significance (ON/OFF);
- Addressing mode (24/31-bit);
- The instruction Address.

Remarks: The difference between the “error-level” and “retry-level” environments is one of the fundamental concepts in z/XDC. The error-level environment consists of the PSW, register settings and the contents of certain control blocks that existed at the time of the abend. The retry-level environment is the same group of settings, but are the ones that will be used at retry time (A GO, a TRACE or TRAP command – any resumption of the program). Most of the time these environments are, and remain the same. But when these two environments are different, Dave flags it and announces it in the beginning of a z/XDC Session (“The Error level and retry level environments are DIFFERENT.”). In that case you may have a recursive abend and a debugging scenario that is a bit more complex than you expect. The LIST EPSW command allows you to see the error level PSW. Conversely, the LIST PSW command shows you the retry-level PSW. For Dave’s excellent discussion on this topic, see Help EXEcutionlevels.

Examples: L EPSWE F ← Display the Program Status Word with the above fields “broken out”.

More Info: Enter Help COmmands LISt EPSWE.



Display the list of Equates defined for the session.

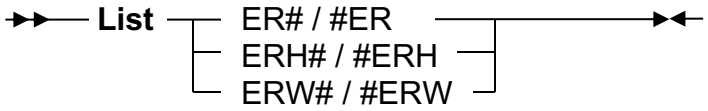
mask Specify combinations of names, and wildcard characters to "filter" the output of this command. For example, If you had previously issued "List RBs" you could issue "List Equates RB*" and all equates with the name "RB#n" would be displayed. Use a question mark ("?") to substitute any single character in an equate name. Use an asterisk ("*") to replace one or more characters in an equate name.

Remarks: When entered, z/XDC will produce a display of all equates defined for the session. This includes a list of "built-in" equates that the product offers as defaults (see the section on z/XDC's Built-in Equates for a list), as well as user-defined equates (see the Equate command), and others that are generated automatically as results of the List TAsks, List RBs, List EStae and GEtmain commands. All of the built-in equates are prefaced with an at sign ("@") and IF they're not static are constantly updated to show the current values (they "float"). Thus @R1 can be used as a symbolic reference no matter what the register happens to contain at any moment.

The List EQuates command details these fields:

- Equate name;
- "(D)" for data or "(I)" for instruction boundary;
- Virtual address;
- "width" of the equate in bytes;
- Which address space created the equate;
- Whether the equate is "Built-in", "Private", "Global" or "Common".

More Info: Enter Help COmmands LISt EQuates. Also, see Help COmmands EQuate, and Help COmmands DElete Equates.

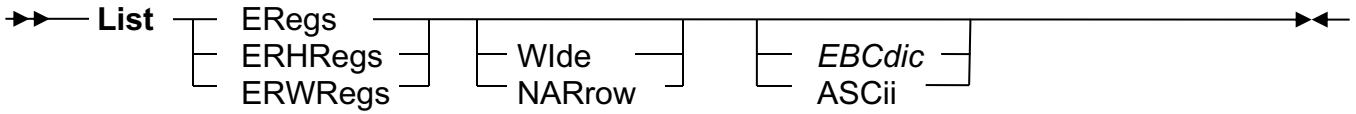


Display the contents of an individual Error-level Register.

The number of the desired Error Register and ranges from 0 to 15.
ER If z/XDC detects a 64-bit environment, this parameter will display only the low-order half of the Error Register.
ERH If z/XDC detects a 64-bit environment, this parameter will display only the high-order half of the Error Register.
ERW If z/XDC detects a 64-bit environment, this parameter will display the entirety of the Error Register.
Remarks: The slash above ("/") indicates that z/XDC will accept either form of the command. When issued, z/XDC produces a comprehensive display of the contents of the desired Error-level Register.

Examples: L ER4 ← Display the contents of Error Register number four.

More Info: Enter Help COmmands LISt Generalregisters Individual. For a discussion on z/XDC's Error-level vs. Retry-level environments, See Help EXEcutionlevels.

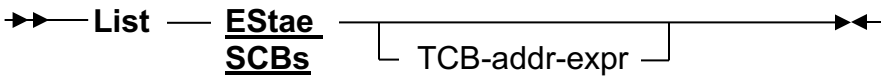


Display the contents of the Error-level General Register set.

ERegs	The Error-level register set. In 64-bit addressing mode, this will show only the low-order half of the 64-bit error-level General Register set.
ERHRegs	In 64-bit addressing mode, this will show only the high-order half of the 64-bit error-level Registers.
ERWRegs	In 64-bit addressing mode, this will show the entirety of the 64-bit error-level Registers.
WIDe	Display the output of the command using as little “vertical real estate” as possible. The output of this command takes up three lines of the display.
NARrow	Display the output of the command using more “vertical real estate”. The output of this command takes up five lines of the display, and is suitable for narrower screen geometries.
EBCdic	Default behavior. Shows the text representation the contents of registers in the EBCDIC character set.
ASCii	Shows the text representation of the contents of registers in the ASCII character set.
Remarks:	This command displays the full set of Error-level General Registers. The registers are displayed in hex-text (dump-like) format.

Examples: L ER ← The error-level general registers are shown.
L ER Narrow ← Display the error registers using a narrow format.
L ERWR ← if in 64-bit addressing mode, the entirety of the 64-bit error-level General registers will be shown.

More Info: Enter Help COmmands LISt Generalregisters Registerset. See also Help EXEcutionlevels.



Display the chain of ESTAE structures for the current, or any given Task Control Block in an address space.

TCB-addr-expr The address expression that points to a Task Control Block. If you issue "List TAsks" prior to List EStae, then you can use z/XDC's automatically-generated equates that point to each "TCB". They will be named "TCB#1" through "TCB#n", where "n" is the upper number of TCBs for the current address. In this situation, you could issue "List ES TCB#2", for example. Alternatively, you can give the address-expression that resolves to the address of a TCB.

omitted If the TCB-addr-expr parameter is omitted, then z/XDC will display the chain of STAE control blocks only for the current task.

Remarks: List EStae displays the STAE Control Blocks (SCBs) that are queued for any task in any accessible address space. An understanding of the SCB queue is very important for understanding the processing of abends in your programs as they occur and how z/XDC is integrated into your abend recovery structure. When this command is processed, z/XDC automatically creates a set of equates that are named SCB#n The list is displayed from oldest to newest. List SCB is an alias for this command.

If this command is to be used in Foreign Address Space Mode, then the List Tasks command must be entered for the address space first.

Examples: L ESTAE 8DC290 ← Show the chain of STAE control blocks for a TCB (TCB-addr-expr expressed as a virtual address).
L SCB TCB#5 ← After issuing the List Tasks command, this command lists the ESTAEs for the fifth Task Control Block in the address space.

More Info: Enter Help COmmands LISt EStae. For a detailed presentation of what this command displays, see Help COmmands LISt EStae Report.

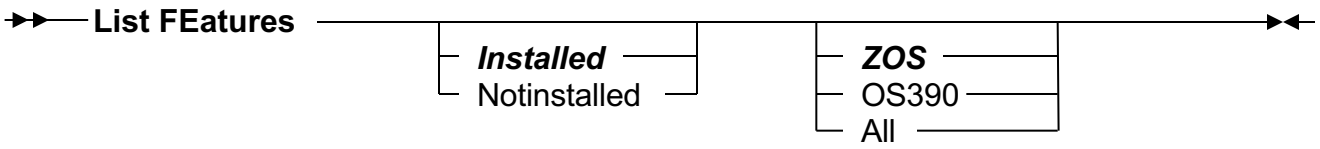
List EXits

Show all exits (if any) defined to z/XDC.

This command accepts no parameters.

Remarks: z/XDC Provides an exits module into which customers may load their own product exits. These are contained in the XDCEXITS module. Once the SET EXITS command has been issued, the List EXits may be used to view them.

More Info: Enter Help COmmands LISt EXits. See also Help COmmands SEt Exits. For more general information, see Help EXIts.



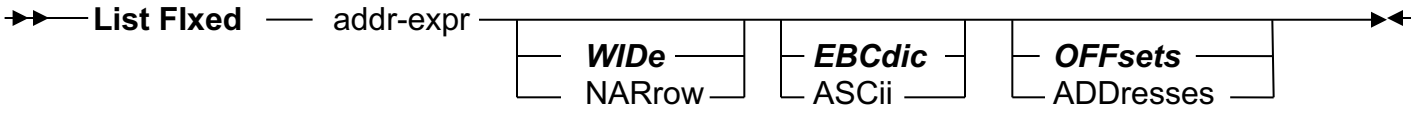
Display some of the hardware and system features that z/XDC uses in its processing.

Installed	Default. Shows a list of installed features.
Notinstalled	Shows features that are not installed.
ZOS	Default. View z/OS-specific features. If you just LOVE to type you can substitute the synonym “ZARCHITECTURE”.
OS390	View features prior to z/OS. If you just LOVE to type you can substitute the synonyms “ESA390” or “OARCHITECTURE”.
All	View all hardware and system features regardless of when they were introduced (includes OS390 and z/OS).

Remarks: The List FEatures command displays some hardware and software features used by the product. Some functions of z/XDC depend on the presence of hardware or software features, such as the Breaking Event Address.

Examples:
LI FE ← show z./OS-specific features available on this processor.
LI FE N ← show z./OS-specific features that are not available on this processor.
LI FE AL ← Show both OS390 and z/OS features that are available on this processor.

More Info: Enter Help COmmands LISt FEatures.



Display a given address as a fixed-point number.

addr-expr	The location that is to be interpreted as a fixed-point number.
WIDe	Default width. z/XDC will display eight words of data per line in its display.
NARrow	z/XDC will display four words of data per line in its display.
EBCdic/ASCii	Display the output in EBCDIC or ASCII format. EBCDIC is the default.
***OFFsets*/ADDresses**	Choose whether the far left column of the display shows an offset from the start of a load module or a csect, dsect or Equate – or – a pure virtual address. OFFset is the default.

Remarks: The List FIxed command displays four bytes of virtual storage in various fixed-point formats. The area being displayed need not be aligned.

The formats displayed are:

- Raw hex-text (dump-like format);
- 1-byte unsigned fixed-point decimal;
- 2-byte signed fixed-point decimal;
- 3-byte unsigned fixed-point decimal;
- 4-byte signed fixed-point decimal.

If an "L" short-cut command is entered next to this display, then z/XDC will "toggle" the display and shows the address as a floating-point number. That's kinda neat.

Examples:

L FI 24CDC ← The address "24CDC" is interpreted as a fixed-point number.

LI FI RW12! ← Thecontents of General Register 12 is treated as a 64-bit pointer to an address, which is then displayed as a fixed-point number.

More Info: Enter Help COmmands LISt FIxed. See also Help Commands LISt FLOAT.

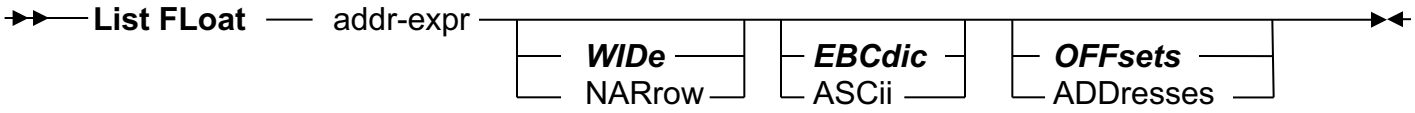
List FLC

Display z/XDC's current Function Leader Character (normally "~").

This command accepts no parameters.

Remarks: When using z/XDC Advanced address expression functions, the address expression needs a Function Leader Character to indicated that what follows the "FLC" is a function. Normally, this is a tilde character ("~") but you can set it to the cent sign ("¢") or a back accent ("`"). Why one would care to change the default is beyond me, but it's nice to have choices.

More Info: Enter Help COmmands LIst FLC. See also Help COmmands SEt FLC. See also Help FUNctions FLc. See also Help ADressing Functions.



Display a given address as a floating-point number.

addr-expr	The location that is to be interpreted as a floating-point number.
WIDe	Default width. z/XDC will display eight words of data per line in its display.
NARrow	z/XDC will display four words of data per line in its display.
EBCdic/ASCii	Display the output in EBCDIC or ASCII format. EBCDIC is the default.
***OFFsets*/ADDresses**	Choose whether the far left column of the display shows an offset from the start of a load module or a csect, dsect or Equate – or – a pure virtual address. OFFset is the default.

Remarks: The List FLoat command displays four bytes of virtual storage in various floating-point formats. The area being displayed need not be aligned.
The formats displayed are:

- The virtual address, the address space, any equates that map this address in memory;
- Raw hex-text (dump-like format),
- The address given as a Binary Floating point number (“BFP”);
- The address given as a Decimal Floating point number (“DFP”);
- The address given as a Hexadecimal Floating point number (“HFP””).

If an “L” short-cut command is entered next to this display, then z/XDC will “toggle” the display and shows the address as a fixed-point number.

Examples: L FL24CDC ← The address “24CDC” is interpreted as a fixed-point number.

More Info: Enter Help COmmands LISt FLOAT. See also Helo Commands LISt Fixed.

List FOrmat

Display the current settings for the Format command.

This command accepts no parameters.

Remarks: The Format command is essential to the use of z/XDC. The actions that Format will take can be configured with the SET Format command. One can choose to interpret raw storage as source code, or object code (or both). One can choose to see or hide macro expansions. One can choose to see z/XDC displays in terms of address or offsets, hexadeciamal vs. decimal, EBCDIC vs. ASCII and much more. One may configure the settings of the FORMAT command within any given Profile by under “Display/Change FORMAT TRACE and FIND Settings.

The default settings for FORMAT in the default z/XDC Profile are generally quite useful. That said, one may consider changing z/XDC’s default breakpoint Opcode from ZERO to TRAP2, thereby allowing z/XDC to run as a trap-handler. It’s all up to the user.

More Info: Enter Help COmmands LISt FOrmat. See also Help COmmands SEt FOrmat.

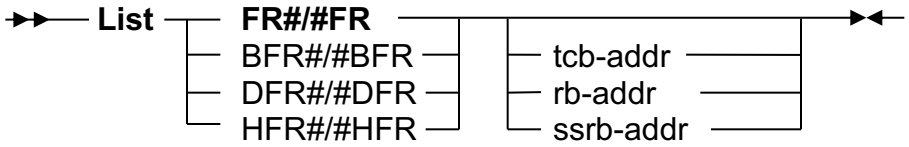
List FPC

Display the state of the Floating Point Control Register (if the Advanced Floating Point Facility is active).

This command accepts no parameters.

Remarks: The Floating Point Control Register is used to control some aspects of Binary Floating Point processing, Decimal Floating Point processing and vector processing. While every Task and SRB may have its own Floating Point Control Register, the facility must be activated in order to show any values as a result of this command.

More Info: Enter Help COmmands LISt FPC.



Display the contents of an individual Floating-point Register.

Where "#" is the number of the desired Floating-point Register. z/XDC will accept "#FR" as well as "FR#".

z/XDC displays the result in several ways:

- As a raw hex-EBCDIC or hex-ASCII number
- As a hexadecimal floating point number (HFP), both short and long.
- As a Binary floating point number (BFP), both short and long.
- As a decimal floating point number (DFP), both short and long.

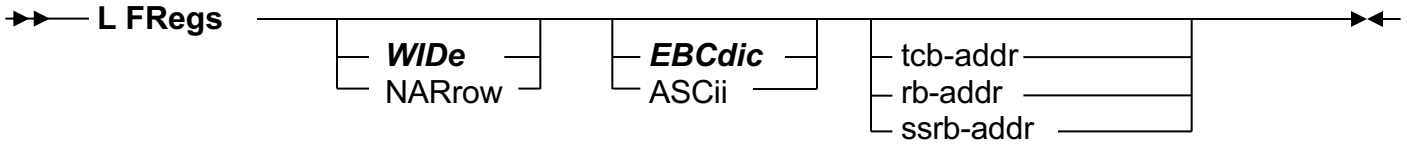
tcb-addr/rb-addr Displays the floating point register set owned by the specified task (Floating point registers are saved only at the task level, so specifying rb-addr is equivalent to specifying tcb-addr).

ssrb-addr Displays the floating point register set owned by the specified suspended Service Request Block. Note that if you issue the LIST SSRB command, then you may use the equates generated by that command as ssrb-addr parms for this command. The equates are SSRB#n and SSRX#n.

omitted The individual Floating Point Register for the current program is displayed in all formats.

Examples: L FR3 ← Shows the contents of Floating-point Register number 3 in Binary Floating Point, Decimal Floating Point and Hexadecimal Floating Point formats.

More Info: Enter Help COmmands LISt FLOATIngpointregisters. See also Help COmmands LISt FLOATIngpointregisters Individual.



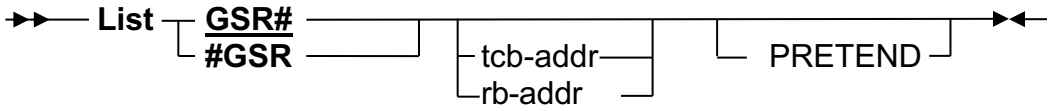
Display the contents of the Floating point Register set for a given task or suspended SRB.

omitted	The current program's Floating point register set is displayed.
WIDe	Default width. z/XDC will display eight words of data per line in its display.
NARrow	z/XDC will display four words of data per line in its display.
EBCdic	Default. The display will be shown in EBCDIC format.
ASCii	The display will be shown in ASCII format.
tcb-addr/rb-addr	Displays the floating point register set owned by the specified task (Floating point registers are saved only at the task level, so specifying rb-addr is equivalent to specifying tcb-addr).
ssrb-addr	Displays the floating point register set owned by the specified suspended Service Request Block. Note that if you issue the LIST SSRB command, then you may use the equates generated by that command as ssrb-addr parms for this command. The equates are SSRB#n and SSRX#n.
omitted	The Floating Point Registers for the current program is displayed in all formats. This is the default action.

Remarks: This command displays the full set of Floating-point Registers. The registers are displayed in hex-text (dump-like) format. If tcb-addr/rb-addr and ssrb-addr are unspecified, then z/XDC will display the Floating Point registers for the current program. This is the default action.

Examples: L FR ← Shows the contents of the Floating-point Registers.

More Info: Enter Help COmmands LISt FLOATIngpoint registers. See also Help COMmands LISt FLOATIngpointregisters Registerset. See also Help COmmands List SSRB.



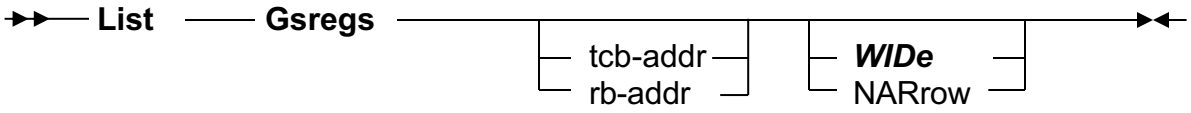
Display the contents of a particular retry-level general Register.

#	The number of the desired Guarded Storage Register and ranges from 0 to 15. z/XDC will accept “nGSR” as well as “GSRn”.
tcb-addr	The address of any Task. May be given as a hexadecimal address or symbolically (as in “TCB#n” once the LIst Tasks command has been entered).
rb-addr	The address expression of a given Request Block. The List RBs command yields equates that make this very convenient.
Omitted	The named Guarded Storage Register for the current program will be displayed.
PRETEND	Whether or not GSRegs are activated you may see what they would look like – even if all the entries are dashed out.

Remarks: Guarded Storage Registers are available only when the GSR facility is activated for a given program(s).

Examples: L GSR3 ←Shows the contents of Guarded Storage Register number 3.

More Info: Enter Help COmmands LISt GGSREgisters Individual. See also Help LIST GSR# . For general information see Help COmmands LISt GSREgisters.

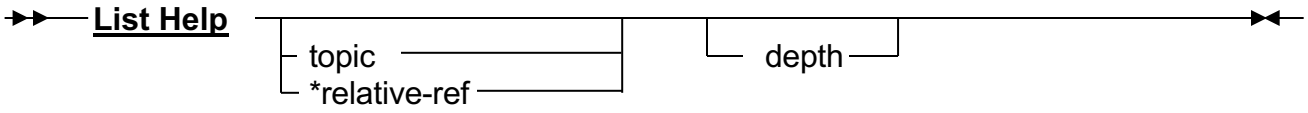


Display the set of Guarded Storage Registers.

Regs	Display the Guarded Storage Registers. In 64-bit addressing mode, this will show only the low-order half of the 64-bit Guarded Storage Registers.
RB-addr-expr	z/XDC will display the register set for the module running under the given Request Block. If you issue the appropriate LIST RBS command you may then use the RB#n equates that LIST RBS generates.
WIDe	Default. z/XDC will display eight words of data per line in its display.
NARrow	z/XDC will display four words of data per line in its display.
PRETEND	If you just want to see what these registers WOULD look like if the facility was activated, you may.

Remarks: Guarded Storage Registers are used in conjunction with some instructions (LGG and LGGFSG) to guard a range of storage so that references to that storage can be controlled. This facility is typically used to implement efficient “garbage collection”. Guarded Storage is documented in “z/OS: Principles of Operation (SA22-7832). If implemented on a processor, z/XDC”s LIST FEATURES command will show them. Guarded Storage is enabled only when a task needs it. Guarded Storage is not available for Service Request Block-mode programs.

More Info: Enter Help LISt GSREgisters. See also Help COmmand LISt Generalregisters Registerset.



Displays local and global indices into z/XDC's Help Subsystem.

topic The name of a Help Subsystem topic or sub-topic. Some examples are Breakpoints, Debugging, Commands, Support, etc.
***relative-ref** This is the "direction" in which the user wishes to go, prefaced by an asterisk ("*"). Vector can be*Up, *Down, *Forward, *Back, *Next, *Previous, and *Repeat.
Depth A number, beginning with 1. This controls the "depth" of the search into the Help Subsystem.

Remarks: This is a handy way to quickly navigate to your area of interest within z/XDC's Help Subsystem. To "arrive" at a desired panel, the user enters a path name to it by naming items in the index from the general to the specific. If you see a "+" next to a topic you may expand it with an "L" command. If you wish, you may put an "H" next to any line and press enter to navigate directly to the panel. You can also display the available number of sub-topics by using the depth parameter.

Recently our William Cole implemented effectively what is "LIST HELP" on the colesoft.com website. You may go to https://colesoft.com/help/?p=./HELP.html to see it.

Speaking personally, I think that the depth parm is a very good one. I have NO idea what *relative-ref" actually does besides put me somewhere I didn't expect to be. So, Dear Reader, happily ignore *relative-ref, use depth and be happy. - Bob

Examples: L H D O ← Leads the user to the list of topics to be found under Help Debugging Otherestaes.
List Help Debugging 4 ← This will expand the dispay to show you four levels of the available Help panels under "HELP DEBUGGING",
L H CO Z ← Leads the user to List Help COmmands Zap, showing the panels available from that "place".

More Info: Enter Help COmmands List Help.

List HFR#

Display the entire 8 bytes of an individual floating point register in Hexadecimal Floating Point format.

is one of the sixteen Floating Point Registers (0-15).

Remarks: This command displays the Floating Point Register in Hexadecimal floating point format. You can get the same display from LIST FR#, which shows you the register in three formats.

More Info: Enter Help COmmands LISt HFR#.

List HOoks

Display HOOKs created in memory, if any exist.

This command accepts no parameters.

Remarks: One of the greatest advances in z/XDC's debugging technology has been the advent of the HOok command. This powerful tool allows the user to create a z/XDC session nearly anywhere at any time within a program without regard to other, pre-existing ESTAEs. Nowadays HOOK works in "SVC-unfriendly" environments, including Service Request Blocks. So you can indeed hook almost anywhere/at any time.

The List Hooks command presents the unique name of the hook, the address space for which it has been created, the userid of the creator and the virtual address of the hook itself.

There are some limitations: List HOoks can only "see" the hooks created by the current user.

There are two kinds of Hooks, dynamic and static. Dynamic Hooks are created interactively with the "Hook <addr-expr>" command, the "HD <module+offset>" command and the "K" shortcut command. Static Hooks are created with a source modification such as the "#XDCHOOK" macro or the "cxdchook()" function (in c/XDC). List HOoks cannot "see" static hooks.

Examples: L HO ← All dynamically created Hooks are displayed. You can use shortcut commands in this panel. You may use the "O" shortcut command to delete aHook(s) if you wish, as well as "D" for Display, "F" for Format, "M" for Map, "N" for "Note", "O" for Off and "Q" for Set Qualifier.

More Info: Enter Help COmmands List HOoks. See also Help COmmands Hook.

List ILc

Displays the status of the ZAP command's "Instruction Length Check".

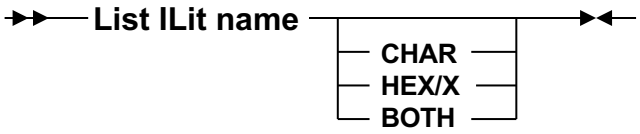
This command accepts no parameters.

Remarks: z/XDC's Instruction Length Check default setting prevents the user from using the Zap command to overwrite an instruction with a different length than the original length. So, normally one could not introduce a four-byte instruction where a two-byte instruction previously resided. This is generally a Very Good Thing.

HOWEVER, if you're a bit-wrangler with some expertise, you may decide you need to do just that. In that case, you may issue SET NOILC, thereby turning this restriction off in z/XDC.

Vaya con Dios.

More Info: Enter Help COmmands LISt ILc. See also Help COmmands SEt ILc.



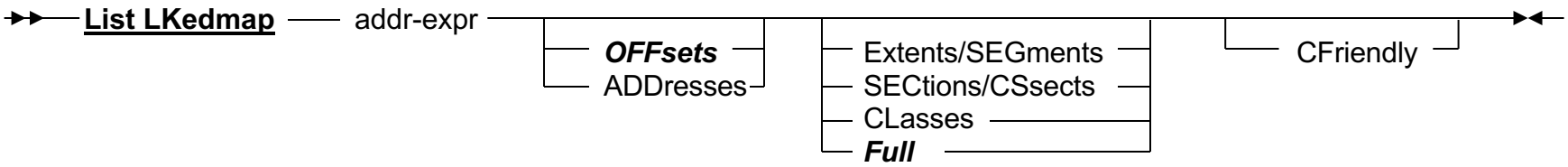
When using dump/XDC, display previously defined literals by name.

name	The name of a previously defined dump/XDC Literal.
CHAR	Returns the value of the Literal in text format.
HEX (or "X")	Returns the value of the Literal in hexadecimal format.
BOTH	Oh, take a wild guess!

Remarks: Under dump/XDC, it is possible to create one's own IPCS literals. The LIST ILIT command can display them.

Examples: LIst ILit fred char ← The literal "fred" is displayed, in character format.

More Info: Enter Help COmmands LISt ILit. See also Help Dumps Literals.



Displays the internal structure of load modules or program objects.

New guys: Think of this as a LIST BINDER command for your compilation units.

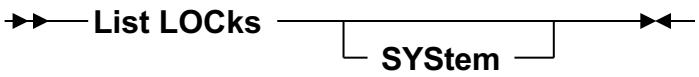
addr-expr	A virtual address that resolves into a location within a load module or program object.
Offsets	Default. Specifies that all displayed symbols will be shown as offsets relative to the start of the class or storage extent in which they occur.
Addresses	Specifies that the starting address of all displayed symbols will be prefaced with an 8-digit virtual address.
Extents/SEGments	Use either "Extent" or "SEGment". When given, the display will show the storage extents in which the load module or program object resides.
SECtions/Csects	Use either "SECtions", or "CSect". Show only the control sections within a load module or program object.
Classes	Shows only the classes within a program object (Classes do not exist within load modules). This parameter requires that the MAP command has been previously issued for a given module.
Full	Default. z/XDC will display all information that it can gather for the given module or program object.
CFriendly	The report will be "filtered" to remove all C language "boilerplate", leaving a report which contains only user-created code sections (csects) and external function names. "C-Friendly", "CF" or "C-F" are synonyms.
omitted	All information will be displayed for the load module or program object.

Remarks: This is a core command within z/XDC. It's a quick way of seeing the "anatomy" of a piece of code.

When this command is entered, z/XDC will display the Module name, the entry address, the length of the module/object and comments. You, as the user, have the opportunity to use a lot of cool shortcut commands in the output of this command. To find out what's allowed, put a "?" in column one and press Enter. Hey! You could MAP, for example.

Examples:
L LK PSW! ← z/XDC will display information about the "current" program.
L LK IGC0001I ← The Open SVC's load module and all of its entry points are shown.
LIS LKEDMAP PSW? C-F ← The entry points for this particular C code section are displayed (presuming you're looking at a C program), and extraneous information is not made visible.

More Info: Enter Help COmmands LISt LKedmap.

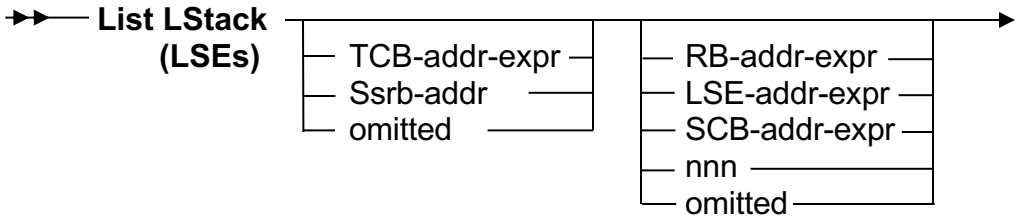


Display the names of all locks held by the program being debugged.

SYStem This parameter may only be used when in dump/XDC. It will show the locks held by the system.

Remarks: z/XDC allows the user to debug Service Request Block-mode code. This command interrogates the status of locks held by a program. The output of this command shows all locks found at abend error time in order of hierarchy, as well as any locks that will be reacquired upon program resumption.

More Info: Enter Help COmmands List LOCks. See also Help Debugging LOSTLOCKS, Help COmmands SEt LOCks and Help DEBugging SRBmode.



Display individual “state” entries from the Linkage Stack.

TCB-addr-expr An address expression that resolves to the start of a Task Control Block (“TCB”), such as TCB#n (An equate that is generated during the List Tasks command).

Ssrb-addr The address of an SSRB or SSRX (an extension to the SSRB that resides in 64-bit storage) located in any accessible address space.

Omitted A default TCB is chosen, depending on the value of the next parameter

RB-addr-expr An address expression that resolves to the start of a Request Block, such as RB#n (which comes from the List RBS command).

LSE-addr-expr The address of a specific Linkage Stack entry. If a previous List LStack command has been entered, then LSE#n may be used.

SCB-addr-expr The address of a STAE Control Block queued to the TCB whose linkage stack is to be examined. When RB-addr-expr is given and TCB-addr- expr is omitted, then the default TCB used is the TCB to which the given Request Block is queued.

nnn A decimal number specifying the state type LSE to be displayed. The oldest is counted as number 1, not including header or trailer entries.

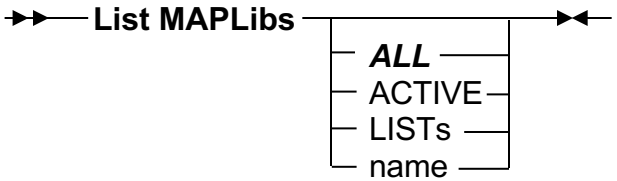
LSE/RB/SCB-addr-expr omitted z/XDC will show a summary of all state-type stack entries that exist in the specified or default linkage stack.

Omitted When all operands are omitted, the default used is the current TCB under which z/XDC is running.

Remarks This command displays individual Linkage Stack “State” entries for any task or SRB routing running in any accessible address space. When LSEs are displayed for the task in which z/XDC is running, the newest one listed will have the label (ARTIFACT) appended to it. This is something that z/XDC is using for its internal purposes.

Examples:
L LS 21C% ← Display the linkage stack for the current task.
L LS TCB#3 ← After issuing the LIST TASKS command, this command will display the linkage stack for the third Task Control Block in the address space..

More Info: Enter Help COmmands LISt LSTack. List LSEs is an alias for this command.



Display overview information on groups of MAPLIBs defined by the user.

ALL Default. Displays all active as well as saved MAPLIB lists. This parameter is mutually exclusive with all other parameters.
Active Shows the list of currently defined and available MAPLIBs.
LISTs Displays the names of saved (and therefore inactive) lists of MAPLIBs.
name Displays only the named list of MAPLIBs, such as “mymaplb”.

Remarks: MAPLIBs are repositories of ADATA information that z/XDC uses to provide source-statement debugging. z/XDC users can create uniquely named lists of MAPLIBs that are relevant to various projects that they undertake. This command allows the user to interrogate which list(s) of MAPLIBs are available or active.

At the time of this writing, z/XDC can process SYSADATA created by three assemblers: IBM’s High Level Assembler, Tachyon Software’s Cross Assembler and z/Assembler and Dignus’ Systems/ASM.

Examples: L MAPL ALL ← Show all MAPLIB lists
L MAPLIB LIST ← Show all saved (and inactive) MAPLIBS.

More Info: Enter Help COmmands LISt MAPLibs. See also Help COmmands SEt Maplibs. See also Help COmmands DElete MAPLibs.

List MAPS

Displays the names and locations of all currently defined Binder maps, CSECT maps and DSECT maps, shown in their search order.

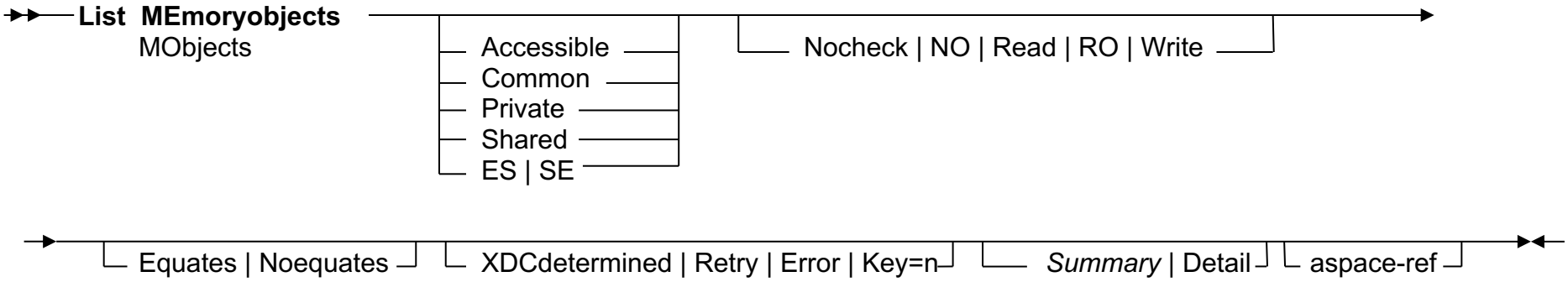
This command accepts no parameters.

Remarks: The MAP and DMAP commands bring in SYM and ADATA information into memory for use in z/XDC's formatted displays. This command allows the user to interrogate the list of maps currently in memory.

List MAPS also shows:

- The map's type (LKED, CSECT or DSECT);
- The map's data type (ESD, SYM, ADATA or DWARF – for c/XDC);
- Whether a map is global across address spaces or is local to a particular address space.
- Whether a map is case-sensitive (C) or not;
- Whether a map represents a Privately Loaded module (P) or not;
- Whether a particular csect map has been designated as the default qualifier (Q) by the Set Qualifier command;
- For dsects, whether the map has been assigned a fixed base address, a floating base address or remains with no base address defined;
- For floating dsects, what the "basing" address expression is.

More Info: Enter Help COmmands LISt MAPS. See also Help COmmands MAP, Help COmmands DMAP.



Display information about above-the-bar memory objects in any accessible address space.

Accessible Information about all Memory Objects that are known to the address space you're working from.
Common Memory Objects common to all address spaces.
Private Memory Objects that are defined to the current address space.
Shared Memory Objects that have been "shared into" the current address space.
ES or SE Abbreviation for "Eligibleforshare". Memory Objects that are eligible to be shared to the current address space.
Nocheck | No | Read | RO | Write Pick one. NO and NC are abbreviations for Nocheck (which means no access checks will be done). Read means Memory Objects that can at least be read. RO is an abbreviation for Read-Only (Memory Objects that can ONLY be read). Write means Memory Objects that may be written to and read from.
Equates | Noequates Pick one. Equate names may be created for each Memory Object. No equates means that equates will NOT be generated.
XDCdetermined | Retry | Error | Key= Pick one. The PSW key will be taken from the retry, or error level environments are the same (the retry level PSW PSW Key is used) or different (the error-level PSW Key is used). Retry forces the use of the PSW Key from the retry-level environment. Error forces the uses of the PSW Key from the error-level environment. Key= means you make one up yourself (using a hexadecimal number from 0 – F).
***Summary* | Detailed** Pick one. Summary means a single line will be displayed for each Memory Object. Detailed means that several lines will be displayed for each selected Memory Object.
aspace-ref This can be any of the following: Job name, a keyword identifying the address space such as HOME, PASID, ESAID, IASID, etc., an ASID number (hex number up to four digits in length), an address-expression pointing to an ASID number, a register containing an ASID number or the name of an equate or dsect that has been assigned to represent storage in an address space. If the parameter is omitted, z/XDC will attempt to show memory objects for the current address space.

Remarks: z/XDC can discern memory objects above the 2-gigabyte "bar", and will optionally create (or recreate) certain equates such as:

- MOB#1-nn ← Memory objects;
- GUARD#1-nn ← Guard areas, if any;
- MOBDATA#1-nn ← any unguarded area.

If you're using this command under dump/XDC and no Current eXecution Unit has been defined then dump/XDC will proceed as if Nocheck is in effect.
Share-Eligible Memory Objects cannot be accessed by a specific address space until they are made accessible by the **IARB64 REQUEST=SHAREMEMOBJ** Assembler macro.
Operands may be entered in any order, but BE CAREFUL. If a particular operand is specified more than once (and why do that?) then the first instance is treated as expected and the second instance is treated as an aspace-ref. This could potentially result in huge confusion.

Examples: List ME JES2 ← Memory Objects within the JES2 address space are displayed, if any exist.
LI ME PASID ← Memory Objects accessible from the retry-level, primary address space are displayed.

More Info: Enter Help COmmands LISt Memoryobjects.

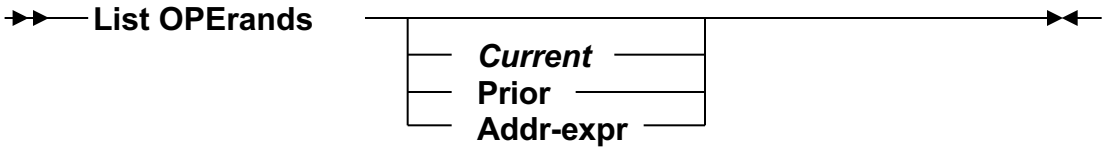
List Notes

Display previously created Notes, if any exist.

This command accepts no parameters.

Remarks: The Note Facility allows a user to make “book-marks” in storage or in programs. It’s a seldom-used corner of the product (as is COMMENTARY) The Note command is also a shortcut command (“N”), so Dave thinks it’s that useful. If it’s a shortcut command I’m honor bound to document it. Here you go.

More Info: Enter Help COmmands List Notes. See also Help COmmands Note, and Help SHortcutcommands.



Display the operands for the current or prior instruction.

Current	Or omitted. Displays the operands of the current instruction. This is the default action.
Prior	Displays the operands of the prior instruction.
Addr-expr	Displays the operands of a machine instruction at a specific address.

Remarks: This command was originally planned for z/XDC's C support. However due to the complexities of the design, it is now intended for Assembler language only, with future expansion planned. It's pretty cool for Assembler, though!

List OPErands really shines if you open a Watch Window and populate the command line it with it. It's even more useful if you populate the command line with "List OPERANDS;;LIST OPERANDS PRIOR". You can use z/XDC's point-and-shoot commands ("D_", "F_", "%"," ?" and "!") on the output of List OPErands to quickly view the contents of storage that will be affected by any instruction.

Note: Because it becomes progressively harder to predict the TRUE state of operands much in the future, the Addr-expr parameter is almost useless the farther you go into the "future" of program execution. And, while you can issue list operands for a machine instruction in the past, there isn't much point to that as it will already be seen in your z/XDC session log. I suggest that you confine your usage of List OPErands to "Current", and "Prior".

Examples: L OPE ← Displays the operands of the current instruction.
L OPE P ← Displays the operands of the prior instruction.

More Info: Enter Help COmmands LISt OPErands.

List PARseorder

Query the order of languages in which c/XDC will parse a string.

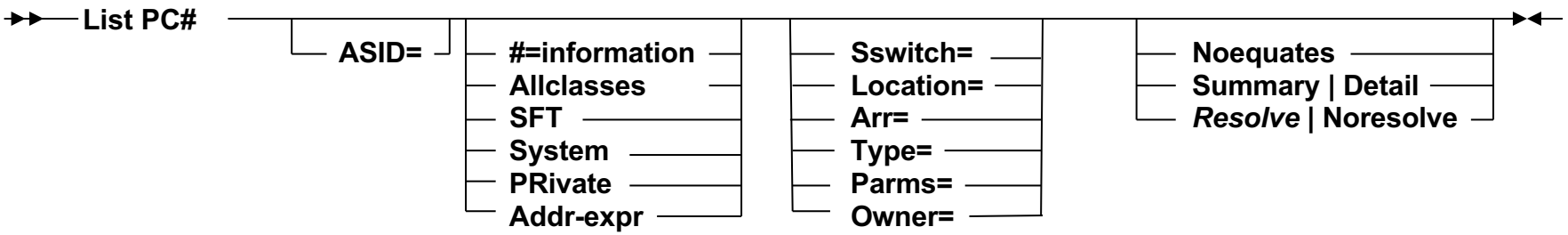
This command accepts no parameters.

Remarks: z/XDC can interpret strings in different ways depending on which High Level Language you wish it to “bias” towards. Indeed it MUST do so, because the syntax rules vary from one language to another. The variables supported at this time are “CEE” (for C/C++/Metal C) and “ASM” for (Assembler).

The default behavior is for z/XDC to use “ASM” first and “CEE” second.

The Parse order can be manipulated in your Profile. To review it, enter “Profile”, then select the sub-topic “Display/Change READ ZAP and Parse Order Settings”.

More Info: Enter Help COmmands LISt PARseorder. See also Help COmmands SEt PARseorder.



Display all available Program Call numbers (PC#s) for an address space.

ASID= This can be a jobname, but other parms are available (see Help Commands Syntax ASIDS). If this parameter is omitted, ASID resolves to the home address space.

3=information The name of a specific PC number – a one to eight-byte PC number or a range of numbers for an inclusive list of PC#s. It can also be D**, a one to six-byte hexadecimal number with a maximum value of 7FFFFF. All PC numbers from D00 to DFF (inclusive) will be displayed (i.e. all EXs for a given LX). Whatever that means.

Allclasses All PC numbers available to the nominated (or default) address space.

Sft Show all PC numbers supplied by the operating system. Synonyms are "MVS", "OS" and "ZOS".

SYstem Shows all PC numbers, system-wide (available to all address spaces). For some reason that doesn't include SFT entries, so you may need to run both or either.

Private Shows all PC numbers available to the nominated (or default) address space.

Addr-expr Shows PC numbers for a PC with the nominated routine address.

Sswitch= Shows space-switching PCs. Values can include Sswitch=NO, =YES, or a specified ASID number.

Arr= Shows PC numbers for Associated Recovery Routines. Values can include =NO, =Yes, =Common and =Private.

Type= May be "=basic (which do not use the linkage stack) and =Stacking (which DO use the linkage stack).

Parms= May be =no (select PCs that have both parm1 and parm2 as zero) or =yes (select PCs that have either or both parm1 and/or parm2).

Owner= Can be =asid, which means to select PCs owned by a single address space (again, see Help COmmands Syntax ASids).

Noequates Prevent z/XDC from automatically assigning equate names to the PC#s that are displayed.

Summary | Detail Pick one. Summary means one line (only) is displayed for each PC#. Detail shows multiple lines for each PC# displayed

***Resolve* | Noresolve** Pick one. Resolve is the default. It means that 2 of the detail lines contain the PC routine address and the Associated Recovery Routine address). Noresolve means that z/XDC will not identify the PC routine address and will not show the ARR addresses.

Remarks: With the near-demise of writing one's own Supervisor Calls (SVCs) and the dawn of writing Program Call routines instead, this command needed to exist in z/XDC. Guys, I think this one's monster-cool! Good work team.

Examples: LI PC S=n ← Show all PC numbers that do not do space-switching.
LI PC Ow=001E ← show all PC numbers owned by what I THINK is the Virtual Lookaside Facility (I did LIST ASID 001E and z/XDC told me. Must be true).

More Info: Enter Help COmmands LI PC. See also Help COmmands Syntax Asids.

List PET addr-expr

Display Pause Element Tokens for any task or SRB.

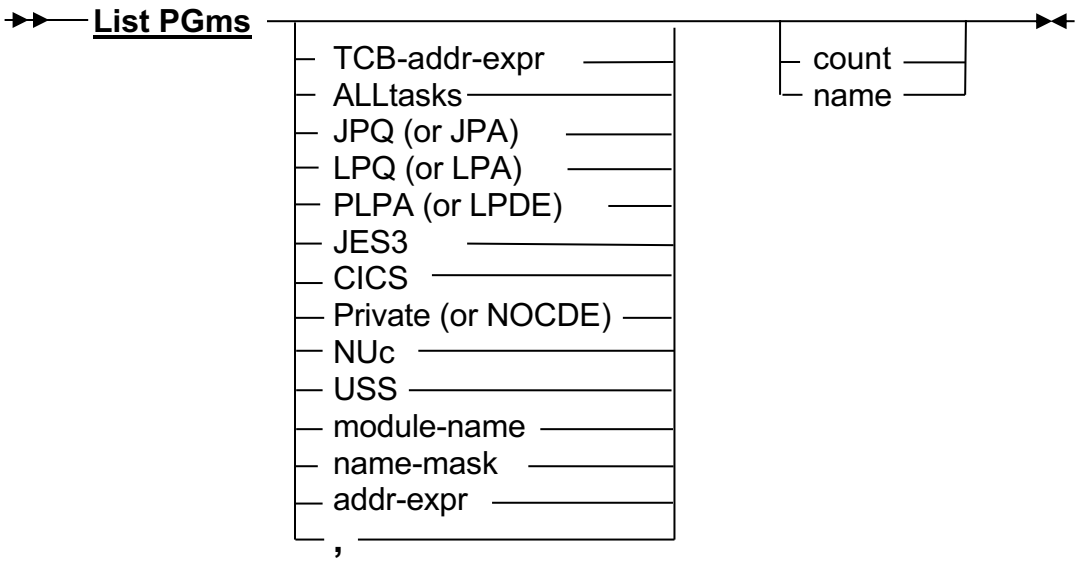
addr-expr the address-expression for any Pause Element Token

Remarks: Pause Element Tokens (“PETs”) are created (and later destroyed) as a pausing mechanism for tasks or SRBs. The z/OS module IEAVAPE allocates them. PETs may be used as input to Pause, Release, transfer and Deallocate_Pause_Element.

This command requires that z/XDC be running APF-authorized.

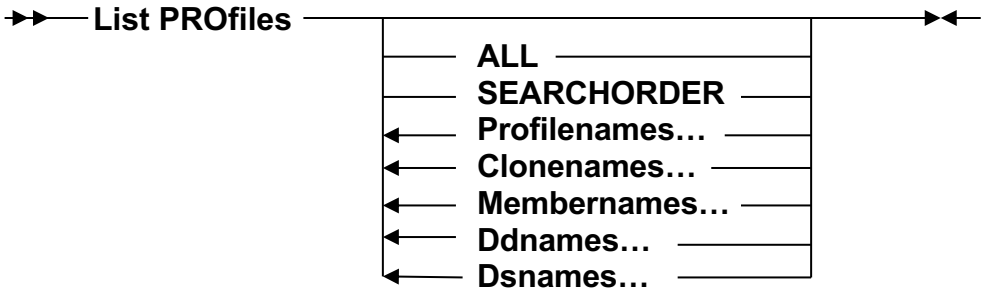
Examples: LI PET 0000F1D0 ← Information about the PET located at this address in the current address space is shown. Good boy, 61904!
LIST PET A0000000~ASID(MYJOB1) ← Information about the PET located at address A0000000 in the address space of MYJOB1 is shown.

More Info: Enter Help COmmands LISt PEt.



Display the list of load modules in memory for any given task, or locate a module via an address-expression.

1st parm omitted	Default. Display only load modules running under the current TCB. You can't omit this parameter when in Foreign Address Space Mode. **Note:** When the first parameter is omitted, and a second parameter stated, you must use a comma (",") to indicate the omission.
TCB-addr-expr	The queue of load modules associated with any given TCB in any accessible address space If you've previously issued LIST TASKS, then you may use any TCB#n equate as this parm.
ALLtasks	Displays all queues of load modules within all TCBs in this address space. You may NOT abbreviate this parameter to "ALL". It could end up confusing you. Dave says so…
JPQ	Displays the queue of load modules in the Target Address space's Job Pack Queues. An alias is "JPA".
LPQ	Displays the queue of load modules in both Systems' Dynamic Link Pack Area (DLPA) and the Classic Link Pack Area (LPA). This also includes the Modified Link Pack Area (MLPA) and the Fixed Link Pack Area (FLPA). An alias for this parm is "LPA".
PLPA	Displays the queue of load modules in the system's Pageable Link Pack Area. An alias is "LPDE".
JES3	Displays all load modules loaded by JES3's internal services. This command DOES work under Foreign Address Space Mode (wherein one issue the SET ASID command).
CICS	Displays the queue of load modules loaded via CICS services.
Private	A "Privately Loaded" load module is one that has been read into storage in such a way that its location is not described by system control blocks (such as via a directed load or by using the LOAD macro with GLOBAL=YES). Thus, such modules do not have either CDEs or LPDEs. This parameter displays a report of all mapped Privately Loaded load modules and program objects – IF- they have previously been MAP's or DMAP'd. An alias is "NOCDE".
NUc	Displays the system Nucleus (IEANUC01).
USS	Can also be stated as "Pathname", if you like typing. Displays the same list of programs that the ALLTASK operand shows – EXCEPT – that only programs that have a path name are shown.
module-name	Searches all address space and system queues are searched for this name.
Name-mask	Use "?" to substitute a single character in a queue's name. Use "*" to substitute one to many characters in a queue's name.
addr-expr	Returns the name of the module that addr-expr resolves into.
count	Suppresses the display of the first "count-1" entries on the list.
Name	Suppresses the display until "name" appears
2nd parm omitted	No messages are suppressed from the display.



Display any, all or some of the z/XDC Profiles available.

Omitted Displays only the current Profile.

All May not be abbreviated (now you know). Shows all profiles available (allocated to ddnames XDCPROF, ISPPROF, XDCTLIB, ISPTUSR, or ISPTLIB (ISPF's standard TLIB).

SEARCHORDER May not be abbreviated (or the command will look for profiles that match the abbreviated name).

Profilenames… The exact name of one or more profiles.

Clonenames… The names for a profile when the z/XDC clone name is other than "XDC". See Help XDCCLones for more information.

Membernames… The names of any ISPF profile library. In my case, it's "BOB.ISPF.ISPPROF".

Ddnames… Displays profiles allocated to any ddname (such as ISPPROF).

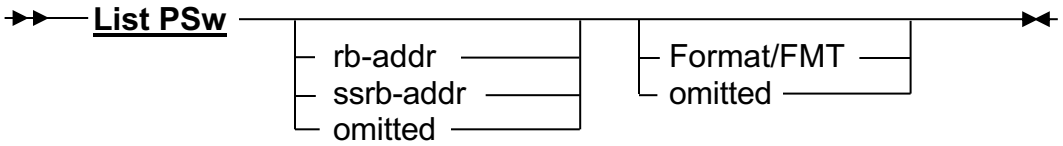
Dsnames… Displays profiles for a dsname (any accessible profile and table library, including those belonging to other TSO sessions).

Remarks: Profiles have been in the product for ages, and yet not many folks use them. Having profiles for different debugging scenarios (even different modules) can greatly help speed the debugging process. You can set groups of Watch, or Display windows on the panel, alter your PFKey settings, screen colors and even how you want z/XDC to look at "the world". You can affect the scope of your trace, set up for c/XDC debugging – and so much more.

To change these things in z/XDC involves learning and using the many SET commands (which is a pain). Profile gives you a menuing system so that YOU DON'T HAVE TO KNOW many SET commands. You can have as many different profiles as will fit into your ISPF profile library.

Examples: LI PRO BOB BOB1 BOB2 ← show each of three profiles.
LI PRO ISPPROF ← Show all profiles allocated to ISPPROF (z/XDC's preferred default).

More Info: Enter Help COmmands LISt PRofiles. See also Help PRofiles..



Display the retry-level Program Status Word for the current or any other Request Block (or Suspended SRB) in the old, 8-byte format.

rb-addr An address expression that resolves to the desired Request Block. If you have previously issued LIST RBS you may use the RB#n equate for this parameter.

ssrb-addr The address of any suspended SRB or SSRX(an extension of the SSRB that resides in 64-bit sotrage), located in any accessible address space in the system. If you have previously issued LIST SSRBS you may use SSRB#n or SSRX#n as a parameter to this command.

omitted z/XDC displays the retry-level PSW for the current program.

Format/FMT Causes z/XDC to translate various fields in the PSW for the user. These fields are:

- The PER bit;
- Dynamic address translation (ON/OFF), I/O (ON/OFF), External interrupts (ON/OFF);
- Protection Key;
- Execution State (Problem state/Supervisor state);
- Address Space Control Mode (Primary, Access Register, Secondary, Home), Condition Code;
- Fixed point overflow (ON/OFF), Decimal Overflow (ON/OFF), Exponent underflow (ON/OFF), Significance (ON/OFF);
- Addressing mode (24/31-bit) and;
- The instruction Address.

2nd parm omitted Formatting is not performed.

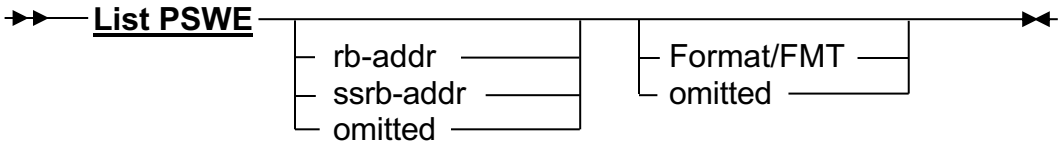
Remarks: This is one of the central commands in the z/XDC debugging tool. You'll use it just about every time you use the product.

The LIST PSWE command will show the full 16-byte format (when executing in the 64-bit environment). All the parameters remain the same.

Examples: L PS F ← Displays the older 8-byte PSW for the current program, and displays the state of the fields.

LI PSW RB#2 ← Displays the PSW for the code running under RB#2 for the current TCB. This command will only work after both the List TAsks and List RBs commands have been entered.

More Info: Enter Help COmmands LISt PSw.



Display the Error-level Program Status Word for the current or any other Request Block (or Suspended SRB) in the newer, 16-byte format.

rb-addr An address expression that resolves to the desired Request Block. If you have previously issued LIST RBS you may use the RB#n equate for this parameter.

ssrb-addr The address of any suspended SRB or SSRX(an extension of the SSRB that resides in 64-bit sotrage), located in any accessible address space in the system. If you have previously issued LIST SSRBS you may use SSRB#n or SSRX#n as a parameter to this command.

omitted z/XDC displays the retry-level PSW for the current program.

Format/FMT Causes z/XDC to translate various fields in the PSW for the user. These fields are:

- The PER bit;
- Dynamic address translation (ON/OFF), I/O (ON/OFF), External interrupts (ON/OFF);
- Protection Key;
- Execution State (Problem state/Supervisor state);
- Address Space Control Mode (Primary, Access Register, Secondary, Home), Condition Code;
- Fixed point overflow (ON/OFF), Decimal Overflow (ON/OFF), Exponent underflow (ON/OFF), Significance (ON/OFF);
- Addressing mode (24/31-bit) and;
- The instruction Address.

2nd parm omitted Formatting is not performed.

Remarks: This is one of the central commands in the z/XDC debugging tool. You'll use it just about every time you use the product.

The LIST PSWE command will show the full 16-byte format of the Error-level Program Status Word. (when executing in the 64-bit environment). All the parameters remain the same.

Examples:

L PSWE F ← Displays the 16-byte PSWE for the current program, and displays the state of the fields.

LI PSWE RB#2 ← Displays the PSWE for the code running under RB#2 for the current TCB. This command will only work after both the List TAsks and List RBs commands have been entered.

More Info: Enter Help COmmands LISt PSwE. To understand more about the Error-level vs. the Retry-level environments, see Help EXEcution levels.

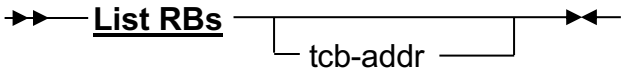
List Qualifier

Displays the default module-name and csect-name (if any has been defined) using the SET QUALIFIER command.

This command accepts no parameters.

Remarks: For your convenience, the Set Qualifier command will define a default mod.csect name. Thereafter you may refer to individual labels in the mod.csect as ".label-name". Otherwise in order to specify a proper label, you would have to issue a z/XDC command such as format this way: Format module.csect.label. That would be a pain, so investigate and USE the Set Qualifier command. Set Qualifier is so useful that there is a "Q" shortcut.

More Info: Enter Help COmmands LIS Qualifier. See also Help COmmands SEt Qualifier. See also Help SHortcutcommands.



Display the Request Blocks associated with the current, or any given Task Control Block.

tcb-addr The virtual address of a Task Control Block. If you have previously issued LIST TASKS, then you may use a TCB#n equate for this parameter.
omitted z/XDC displays the Request Block structure for the current task.

Remarks: For convenience, the List TAsks command should be issued prior to List RBs. Then, if you want to easily query the RB structure under any sub-task, you can put the shortcut command "L" next to any TCB# equate, and press Enter. Nice!

[List TAsks establishes equates in memory pointing to each Task Control Block (TCB#1-n, where n is the number of tasks in the address space) that can be used in the List RBs command.]

In fact, issuing List TAsks and List RBs one after the other on entry to a debugging session is a very good way to determine "where" the program is and why.

Elements displayed by List RBs include:

- The type of RB (PRB, SVRB, Etc.);
- The manner in which it was created (ATTACH, PGM-CHK, etc.);
- The name of the routine associated with it (load-module-name, SVC-n, abend-code, etc.);
- and the resume address relative to the nearest relevant module, map or equate (if any).

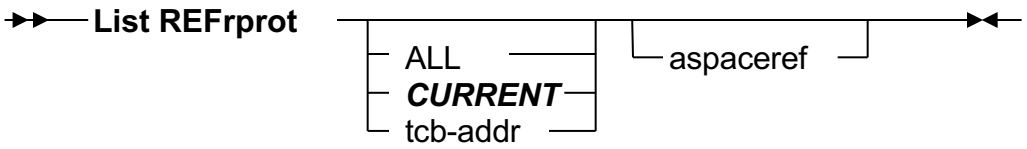
Pro tip: If you end up somewhere and you don't know how you got "here", try this command sequence:

```
LIST BEA;;LIST TASKS;;LIST RBS
```

By stacking these three commands you get a good idea of what's going on. The double semi-colons create blank lines between each command's output for readability. You're welcome.

Examples:
L RB TCB#2 ← Show all the Request Block information for the second Task Control Block in the address space.
L RBS 21C%+7C! ← Show the Request Blocks queued from the Jobstep TCB.

More Info: Enter Help COmmands List RBs. See also Help COmmands LISt TAsks.



Display the current settings for REFRPROT.

ALL Displays REFRPROT settings for all tasks in the current address space.

CURRENT Default. Displays REFRPROT settings for the current task.

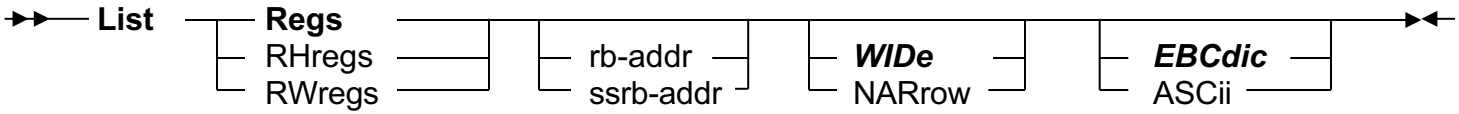
tcb-addr Displays REFRPROT settings for any task. The address given MUST be the starting address of a TCB in any accessible address space on the system. If you have previously issued LIST TASKS, you may use the TCB#n equates as parameters for this command.

aspaceref A 1-4 hexadecimal number corresponding to the address space ID, a 1-8 character jobname or TSO userid, an address space keyword (HOME, PASID, SASID, etc.) and one of these control registers: “CR3”, “CR4”, “ECR3” or “ECR4”. For more information see Help COmmands Syntax ASIDs.

Remarks: Beginning with z/OS V1R9, refreshable modules will ALWAYS be loaded into Key-0 storage regardless of whether they are loaded from an APF-authorized library and regardless of whether or not the RENT attribute is also on. This command allows you to know the REFRPROT settings within the system. These settings can be changed with z/XDC’s SEt REFrprot command.

Examples: L REF ALL ← Display REFRprot settings for all tasks in the current address space.
L REF TCB# JES2 ← Display REFRprot settings for the JES2 address space, Tach Control Block 3. This presumes a prior LIST TASKS command for the JES2 address space.

More Info: Enter Help COmmands List REFRprot. See also Help COmmands SEt REFRprot.



Display the set of Error-level General Registers.

Regs	Display the General Registers. In 64-bit addressing mode, this will show only the low-order half of the 64-bit General Registers.
RHregs	In 64-bit addressing mode, this will show only the high-order half of the 64-bit General Registers.
RWregs	In 64-bit addressing mode, this will show the entirety of the 64-bit General Registers.
RB-addr-expr	z/XDC will display the register set for the module running under the given Request Block. If you issue the appropriate LIST RBS command you may then use the RB#n equates that LIST RBS generates.
Ssrb-addr	z/XDC will display the register set for a suspended SRB or SSRX (an extension of the SSRB that resides in 64-bit storage).
WIDe	Default. z/XDC will display eight words of data per line in its display.
NARrow	z/XDC will display four words of data per line in its display.
EBCdic	Default. The display will be shown in EBCDIC format.
ASCii	The display will be shown in ASCII format.

Remarks: In order to use the RB-addr-expr parameter conveniently, issue LIST RBs for the desired task. That will allow the use of the equate RB#n in this command.

More Info: Enter Help LISt REGisters. See also Help COmmand LISt Generalregisters Registerset.

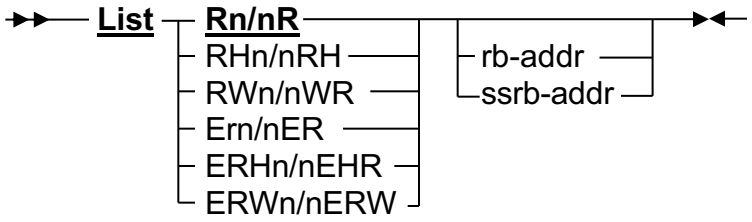
List REXx

Displays information about the currently active REXX interface for z/XDC, if one exists..

This command accepts no parameters.

Remarks: z/XDC allows the creation of a REXX interface. If this is in place you can issue REXX execs from within your debugging session.

More Info: Enter Help COmmands LIst RExx. See also Help Rexx.



Display the contents of a particular retry-level general Register.

n n is the number of the desired Register and ranges from 0 to 15. z/XDC will accept “nR” as well as “Rn”.

RHn/nHR Display the high order half of a particular 64-bit register.

RWn/nRW Display the totality of a 64-bit register

ERn/nER, ERH, ERW, etc.: Display the error level registers. See the “Remarks” section below for more information.

rb-addr The address expression of a given Request Block. The List RBs command yields “RB#n” equates that make this very convenient.

ssrb-addr The address expression of a given suspended SRB or SSRX (an extension of the SSRB that resides in 64-bit storage). If you have previously issued the “List SSRBs” command, then you may use the equates “SSRB#n and “SSRX#n” in this command for this parameter.

omitted The desired register for the program running under the current Request Block of the current Task Control Block is displayed.

Remarks: z/XDC keeps track of two sets of registers and PSWs. One is the Error Level PSW and registers, and the other is the Retry Level PSW and registers.

- The Error level constructs are those extant at the time z/XDC got control – i.e. at the time of the Abend, or breakpoint or tracing command.
- The Retry constructs are those that will be used when control is given back to the program to execute.

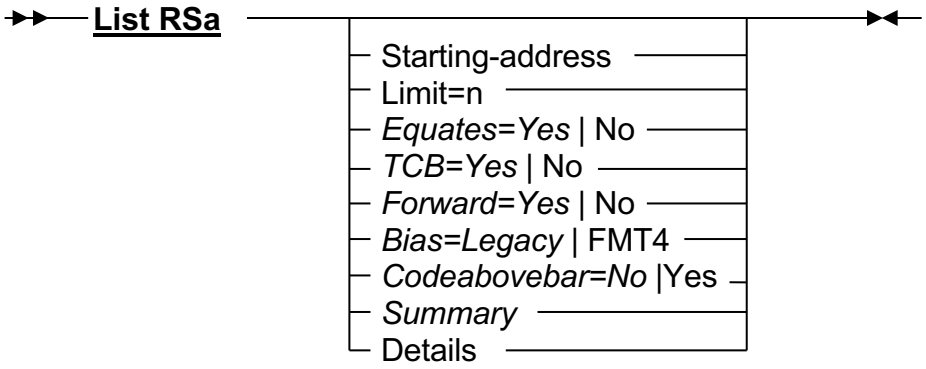
These two environments are normally the same, and issuing List PSW will give you what you need to know. HOWEVER, if z/XDC ever tells you that “THE ERROR LEVEL AND RETRY LEVEL ENVIRONMENTS ARE DIFFERENT”, then you need to pay attention. Some sort of recursive error may have occurred.

z/XDC displays the result of List Rn in several ways:

- As a raw hexadecimal number;
- As its EBCDIC equivalent.
- Different “chunks” of the register expressed as positive or negative integers.

Examples: L R3 ←Shows the contents of general Register number 3.
LI RW12 ← shows the contents of the “wide” (64-bit) format for General Register 12.

More Info: Enter Help COmmands LISt Generalregisters Individual. See also Help EXEcutionlevels.



Interrogate the Register Save Area chain.

Starting-address The address of a register save area located in the chain or the address of a Task Control Block. If this parameter is omitted, then the location pointed to by retry-level Register 13 will be used. If omitted, then the address pointed to by the retry-level R13 will be used as a default.

Limit= A decimal number between 1 and 9999. The default is "99".

Equates= A Yes here causes z/XDC to create equates that map each RSA entry. This is the default. No does not do so.

TCB= A Yes here (default) shows the first save area associated with this or any other Task Control Block (using the LIST TASKS command first allows you to use the equates "TCB#n" in this command).

***Forward*=** Yes (the default) means that z/XDC will attempt to follow the RSA forward from its starting point. No won't.

Bias= Legacy (the default) gives z/XDC a clue that the RSA chain is of the "old style". FMT4 gives z/XDC a clue that the save area chain is a Format 4 save area.

Codeabovebar= Specifying "No" (default) means that z/XDC will ignore the high word of the R14/R15 value. Specifying "Yes" means that z/XDC will use the entire 8-byt value of R14/R15.

Summary Produce only one line per RSA in the command's output.

Detail Produces multiple lines of output for each save area chain value. It also interprets the characters you may see in "Summary" and may describe any errors found for any save area entry.

Limit= A decimal number between 1 and 9999. The default is "99".

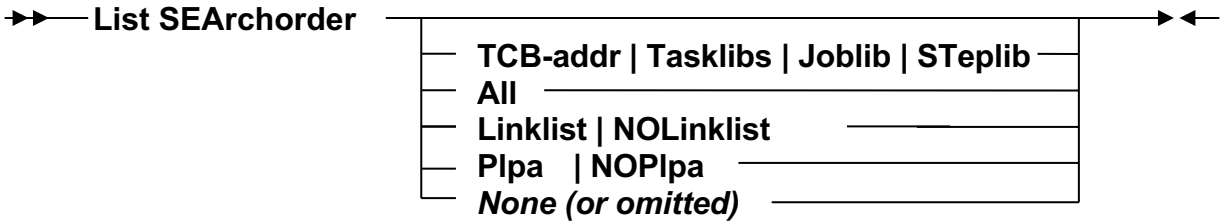
Remarks: This command generates a chain of standard 72-byte register areas. For each save area found, z/XDC displays its address, the return address found in the R14 "slot" and the entry point address found in the R15 slot. The return and entry addresses are, when possible, formatted and displayed as offsets into a load module and/or csect.

For each RSA found, the command shows:

- A sequence number;
- Flags (see help for an explanation);
- The RSA format;
- The RSA length;
- The return address found in the R14 "slot";
- The entry point address or the return code found in the R15 slot;
- Comments about each entry (if Detail has been specified);
- Any errors encountered (again, if Detail has been specified).

Examples: LI RS TCB#3 DE ← Shows a detailed display of save area chains beginning from TCB #3.
LI RSA RB#1 de ← shows a detailed display of save area chains beginning with the equate RB#1.

More Info: Enter Help COmmands LISt RSa.



Display the load module search order (as usedby the ATTACH[X], LINK[X], LOAD and XCTL{X} assembler macros for any Task Control Block in any accessible address space.

TCB-addr The address of any accessible Task Control Block. Tasklibs is an alias for this parameter. So is Joblib. So is STeplib.
All Forces all dsnames to be displayed.
Linklist Shows the search order for the Linklist.
PLPA Display PLPA dsnames
None (or omitted) Prevent all dsnames from being displayed. “N” or “NO” is an abbreviation.
NO<whatever> A NO in any of the parms suppresses the output of each parameter.

Remarks: Here at Cole Software, we often refer to the search order and how z/XDC finds adata and dsect maps. Now it’s pretty clear. The parameters discussed above also depict the search order that z/XDC uses internally. That’s also good to know.

Examples: L SEA TCB#3 ← After issuing LIST TASKS, the TCB#3 equate is used to generate the search order for the third Task Control Block in the address space.
LI SEA LINK ← shows the search order in the linklist.

More Info: Enter Help COmmands LISt SEArchorder.

List SECurity

Display information about z/XDC's security interface.

This command accepts no parameters.

Remarks: z/XDC has a highly evolved security interface, because with z/XDC the user may do "interesting" things in z/OS. z/XDC therefore makes heavy use of SAF calls to protect system integrity. Up until now, z/XDC's security interface was opaque and known only to the Security Officer. Now you can SEE it.

This command displays:

- The Security system in use; ← May be RACF, ACF2 or Top Secret.
- Rules Syntax; ← Shows whether z/XDC-related resource names start with the qualifier "XDC".
- Security userid; ← Shows the ownerid in whos name z/CC makes its security calls.
- Security Trace (if enabled); ← The SET SECURITY command can be used to displayinformation about all security calls made by z/XDC. This may be enabled or disabled.
- MAX Cache expiration. ← Shows how much time remains in the life of the youngest cache entry in z/XDC's internal security cache.

More Info: Enter Help COmmands LISt SECUrity. See also Help COmmands SEt SECUrity.

List SESsions CAPs

Report the status of all Licensed Features in the system and a list of current z/XDC debugging sessions.

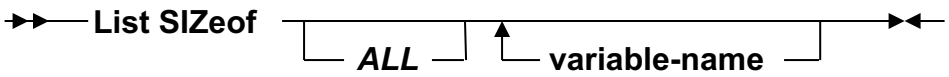
This command accepts no parameters.

Remarks: Any Cole Software debugging session may partake of several License Features. They are:

1. Base/XDC ← The “engine” of Cole Software’s products.
2. Asm/XDC ← Allows source statement debugging in z/OS Assembler language.
3. Auth/XDC ← Allows the user to debug APF-authorized code (and other cool things).
4. c/XDC ← Allows source statement debugging in IBM C language.
5. dump/XDC ← Ports Cole Software’s debugging tools into the IPCS, or dump-reading environment.

The output of this command shows which Licensed Features are in use by users, as well as the upper limit of the Licensed Features and the amount of their usage since the last IPL.

More Info: Enter Help COmmands LISt SESsions. The LIST CAPs command is an alias for this command.



Display the size of all or any specific C variables.

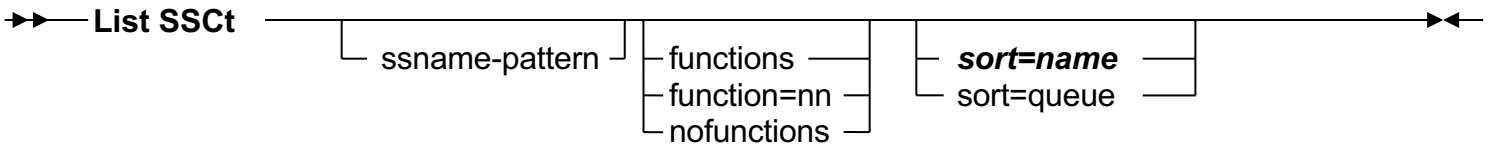
ALL/omitted The size of every variable is displayed, in bytes.
Variable-name The name of one or many variables separated by blanks.

Remarks: When an early c/XDC customer requested this command, we wondered why we hadn't thought of it ourselves. Since then, a SIZEOF field appears in every LIST VARS command output, but is generally off the terminal display, on the right side. We did this because some variable names are very long and we didn't want to truncate them.

In this situation, you can issue RIGHT <nnn> where "<nnn>" is a number of columns to move the display so as to see the sizes, or you can fool around with the Set VDisplay command to change the column boundaries. I'm scared of that command, so I haven't fooled with it yet. Vaya con Dios.

Examples: L SIZ estring ← the size of the estring variable (70 bytes) is displayed. estring is a variable in the CSAMPLE program that we distribute with c/XDC.
Li SIZof multiDouble ← the size of the array multiDouble is shown. multiDouble is another variable inside CSAMPLE.

More Info: Enter Help COmmands LIS SIZeof. See also Help COmmands Right, Help COmmands Left and (if you dare) Help COmmands Set VDisplay.



Display Subsystem Control Tables that currently exist in the system.

Ssname-pattern	Any name or fragment of a name with wildcard characters ("*" and "?"). The question mark substitutes a single character in a name. The asterisk substitutes one-to-many characters in a name. Use asterisk and you'll get 'em all.
functions	Displays all subsystem function exits.
function=nn	Qualify the display of functions by their number.
nofunction	Has the same effect as NOT specifying "Function" or "Function=nn". Whatever.
sort=name	Sort the display by subsystem name. This is the default.
sort=queue	Sort the display in the order in which the system passes control to each subsystem.

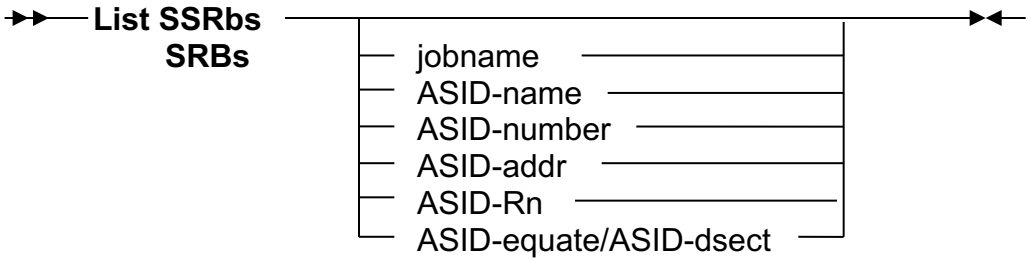
Remarks: Subsystem Control Tables identify registered subsystems in z/OS. This command will show:

- The equate name generated for the SSCT by z/XDC ("SSCT#n");
- The name of the subsystem;
- Whether it is active;
- Whether there is an active or standby Subsystem Vector table (SSVT);
- Its status;
- A comment field that tells you:
  - Whether it is the primary subsystem;
  - Whether it's dynamic:
  - Whether it responds to SETSSI commands.

Finally, Dave shows the number of subsystems defined, the number of systems matching a given name pattern and the number of systems displayed. Complete as always, Dave!

Examples: L SSC FUN=10 ← Display SSCTs with a function value of "10"
L SSCT ← Show 'em all. Same as "LI SSCT *".

More Info: Enter Help COmmands LISt SSCt.



Display Suspended Service Request Blocks (SSRBs) for any address space.

jobname	Qualify the display by job name, or TSO userid.
ASID-name	A keyword, like “HOME”, “PASID”, “ESASID”, “IASID”, etc.
ASID-number	The number assigned to the address space in the Address Space Vector Control Table.
ASID-addr	An address-expression that points to an ASID number.
ASID-Rn	A register that contains and ASID number.
ASID-equate/ASID-dsect	An equate or a dsect that has been assigned to represent storage in an address space.
omitted	Default. Displays SSRBs in the current address space.

List STGlayout

Display the storage layout of the current system.

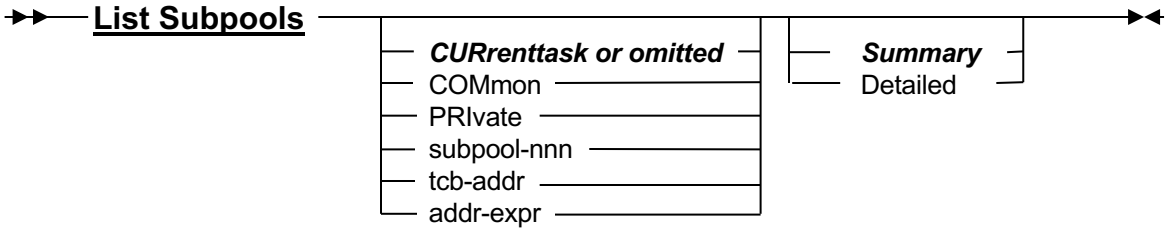
This command accepts no parameters.

Remarks: While you can get the storage layout of any system by issuing LIST EQUATES, and looking at the addresses of the built-in equates that z/XDC generates, the output of this command is far more concise. This is almost TOO obvious, but should be mentioned anyway:

- If z/XDC is active, the storage layout for the current system is displayed.
- If dump/XDC is active, the storage layout for the system that the current dump was taken on is displayed.

Thus, using this command under dump/XDC could be very useful, because the system upon which the dump was generated probably doesn't have the same layout as anyone's "home" system.

More Info: Enter Help COmmands LISt STGlayout.



Display the organization of storage subpools within the Home Address Space.

CURrenttask	Default. Since it's the default, this parameter can be ignored. Displays subpool information for the current task in the address space.
COMmon	Displays subpool extent information for the Systems Common Storage Area and System Queue Area. "Global" is a synonym.
PRIvate	Displays subpool extent information for the Private Area in the address space. "Region" and "Local" are synonyms.
Subpool-id	A numeric value up to three digits in width. z/XDC will return a display for only that subpool extent number (example: "78" or "230").
TCB-addr	The starting address of a Task Control Block. z/XDC will return a display for the given task. If you execute the List TAsks command, you may thereafter use the equates generated automatically by the List Tasks command (i.e. "TCB#n").
addr-expr	z/XDC will display the subpool extent that the address expression resolves within (example, "PSW!" or "R15?").

Summary	Default. z/XDC displays totals for all subpool allocations.
Detailed	z/XDC displays a single line for each and every extent of subpool allocation.

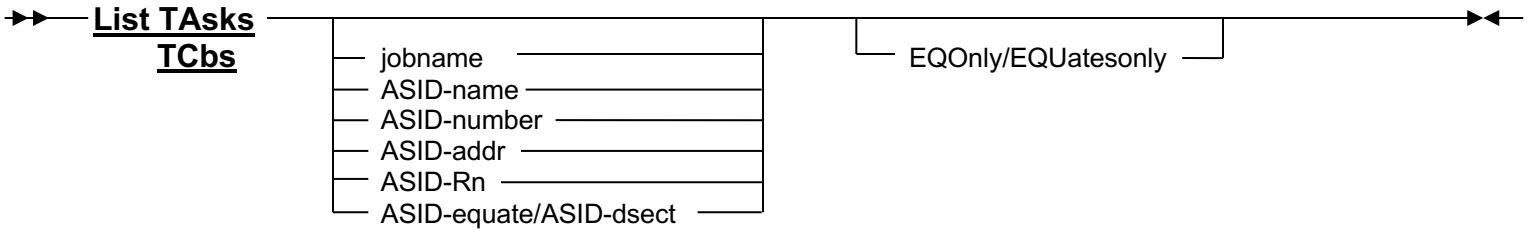
Remarks: When issued, this command will display the starting and ending addresses of each subpool extent, the amount of space allocated, the amount remaining unused, the Key of the extend and the ownership of each extent by task control block number.

This command can be very useful for tracking the allocation of memory by a program (or perhaps its mis-allocation).

Examples:

L S 78	←z/XDC will return the allocations of memory within subpool 78.
L S TCB#3	← If a LIST TASKS command has been executed, you may use the TCB#n equate and see subpool allocations for any given task.

More Info: Enter Help COmmands LIS SUbpools. See also Help COmmands LISt SUbpools Reports.



Display the organization of Task Control Blocks within any accessible address space.

jobname Qualify the display by job name, or TSO userid.
ASID-name A keyword, like “HOME”, “PASID”, “ESASID”, “IASID”, etc.
ASID-number The number assigned to the address space in the Address Space Vector Control Table.
ASID-addr An address-expression that points to an ASID number.
ASID-Rn A register that contains and ASID number.
ASID-equate/ASID-dsect An equate or a dsect that has been assigned to represent storage in an address space.
omitted Default. Displays tasks in the current address space.
EQUonly/EQUatesonly This variant of the command does not display the result, but DOES create the relevant Automatic equates that List Tasks normally does. It might, for example, be very useful in a script.

Remarks: The List TAsks command is one of z/XDC’s most powerful commands. If the List TAsks and then the List RBs commands are entered in sequence, the user can gain a very good understanding of where a program is, and why. List TCbs is an alias for this command.

When entered, List TAsks gives an easy-to-understand graphic display that includes:

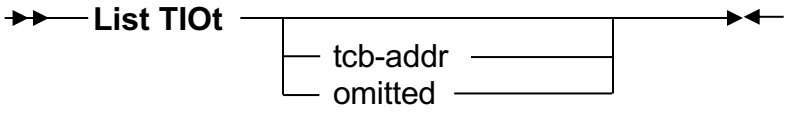
- An equate named “TCB#1-n” for each entry that is created automatically to “name” each Task Control Block;
- The program name running under each Task Control Block;
- The hierarchy of each task by indentation;
- The current task, identified by highlighting;
- The abend type for the current task.

Some of the parameters (those having to do with ASID-etc.) can only work when z/XDC is running APF-authorized.

Note: For dump/XDC users: This command, in conjunction with the LIST RBs command for any sub-task offers the dump/XDC user the ability to alter the Current eXecution Unit with the “CU” shortcut command. Thus, if you prefer to have THIS TASK/THIS RB be the scope of your inquiry, you have a handy too with which to do so.

Example:
L TA ← The task structure for the current address space is shown.
List TA JES2 ← If z/XDC is running APF-authorized, this will show the task structure for the JES2 address space.
L TA 1 ← Displays the task structure for the Master Scheduler Address Space.

More Info: Enter Help COmmands LIS TAsks. This command is really cool. You should read about it.



Display the dataset allocations for any Task I/O Table for any specific task in any accessible address space.

Tcb-addr The display shows Task I/OTable information for the given task. The equate “TCB#n” can be used for convenience after the List TAsks command has been issued.

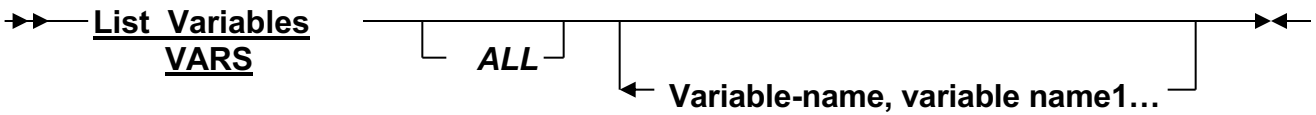
omitted The display shows TIOT information for the current task.

Remarks The List TIOt command is not actually a command, but is a source code module distributed with the z/XDC product (SRCXUCMD, which may be found in the XDCASM library). If the installation instructions are followed, this command can be made available for general usage.

Examples:

L TIO	← If z/XDC is running in Local Address Space Mode, then it displays the dataset allocations for the current task. OR:
List Tasks	← Execute this command. Then, issue:
L TIO TCB#3	← The List TAsks command creates equates TCB#1-n. Then, the List TIOt command displays allocations for the third Task Control Block in the address space.

More Info: Enter Help COmmands List TIOt.



Display all, or individual C variables.

ALL Default (or omitted). c/XDC displays all variables from all variable pools in only the current stack frame. This is the default action.
Variable-name The name(s) of individual variables. If the ALL parameter is given in conjunction with other variable names, then "ALL" is treated as a variable name.

Remarks: This command is central to c/XDC (and our support for other High Level Languages, in the future). If individual variable names are given, then c/XDC will display one or more lines for each variable to include:

- Simple scalar variables;
- Array names;
- Selected array elements;
- Structure names;
- Structures within arrays within structures within…;
- Unions.

The variables value will be displayed, formatted according to the variable's type.

Individual variables may be examined in several ways using new shortcut commands that you put in column 1 of the display line for each variable.

- "D" or "F" - Perform a Display of the variable. It will be shown in raw hexadecimal storage format. Format works the same way in this situation. Note that if you want to change a variable, you MUST first Display it (the "D" shortcut command), then use the "Z" shortcut command to alter it.
- You may use the "L" shortcut to "drill into" a variable, and "L" again to drill out. This behavior is most useful in the Working Window, and acts differently in a Watch, or Display Window.
- Asterisk ("*") – Display the variable in EBCDIC format.
- Vertical bar ("|") – Display the variable in ASCII format.
- Back-slash ("\") – Display a string, up to its null termination.
- Forward-slash ("/") – Display the entirety of a string, up to its length termination.

The default value in the "factory" Profile allows a LIST VAriables upper limit of 32767 lines of display(!). The List Variables Recurse Limit (per structure) is also 32,767 lines.

Examples: L VAR ← You see 'em all. Depending upon your program's complexity, his may result in a certain amount of processing time and a large display.
L VAR FRED ← You see only variable "FRED".
L VARS array[0][1][2-4] ← This variant drills down into a two- or three-dimensional array. It's expressed as a C programmer might expect, and in this case, c/XDC will display the "X" dimension as "0", the "Y" dimension as 1, and the "Z" dimension to show the 2nd, third and fourth elements in the array. There's more in the c/XDC Primer.

More Info: Enter Help COmmands LISt VARiables. See also Help SHortcutcommands.

List VDisplay

Display the current settings created by the Set VDisplay command (which affects the output of the List VARiables command).

This command accepts no parameters.

Remarks: SET VDisplay controls the output of the LISt VARiables command. These settings may be controlled with the SEt VDisplay command. That said, the default settings for variable display in c/XDC are pretty useful. You may never need to change them, but you may if you care to.

More Info: Enter Help COmmands LISt VDisplay. See also Help COmmands SEt VDisplay.

Display the set of thirty two 16-byte Vector Registers.

This command accepts no parameters. Well, except for the one below…

PRETEND You can see how z/XDC will display Vector Registers if and when your local processor supports them. You'll have to type the entirety of "PRETEND", as no abbreviation is allowed.

Remarks: With the Z13 processor, IBM introduced Vector Registers, a set of 32 16-byte registers which are intended for manipulating large floating point numbers. z/XDC supports these registers, and this command will show you the entire group of them.

However, you may or may not have the Vector Facility turned on for your processor. To query this, you can issue z/XDC's "LIST FEATURES" command, and look for the entry, "SIMD Instructions and vector registers". If this doesn't jump out at you, you can confirm the other way by issuing "LIST FEATURES NOTINSTALLED" command.

For processors that do not yet support vector registers you can issue the command with the PRETEND parameter to see what's in store for you.

You can also query individual Vector Registers with the List VR# command.

Examples: L VR PRETEND ← Look at the fake Vector Registers.

More Info: Enter Help COmmands LISt VRegs. See also Help Commands LISt VR#.

Display the contents of a specific Vector Register.

VRn A decimal representation of the desired Vector Register. The acceptable range is from 0-31.

PRETEND If you don't have the Vector Facility turned on in your local environment, the parameter will show you how z/XDC will display it. z/XDC also decodes the contents of such registers just as it does with Control Registers, so that you can their contents without having to decode them. PRETEND has no abbreviation.

Remarks: In z/OS V2R1, IBM introduced Vector Registers, a set of 32 16-byte registers which are intended for manipulating large floating point numbers. z/XDC supports these registers , and this command will show you the entire group of them.

However, you may or may not have the Vector Facility turned on for your processor. To query this, you can issue z/XDC's "LIST FEATURES" command, and look for the entry, "SIMD Instructions and vector registers". If this doesn't jump out at you, you can confirm the other way by issuing "LIST FEATURES NOTINSTALLED" command.

For processors that do not yet support vector registers you can issue the command with the PRETEND parameter to see what's in store for you.

You can also query the entire set of Vector Registers with the List VRegs command.

Examples: L VR0 PRETEND ← See what Vector Register zero will look like whenever you get the Vector Facility turned on in your environment.

More Info: Enter Help COmmands LISt VR#.

List VSEttings

Display c/XDC's current settings for maximum array dimensions and variable stack depth.

This command accepts no parameters.

Remarks: This command will display the Array Depth (1-64 dimensions) and the Variable Stack Depth (1 to 256 levels).

c/XDC comes with its "factory default" Profile set to the maximum values that are supported for a tn3270 connection anyway, so why alter them?

To investigate these settings in your profile, Enter "Profile" on the command line and select the sub-category "Display/Change High Level Language (HLL) Settings".

More Info: Enter Help COmmands LIS VSEttings. See also Help Commands SEt VSettings.

List VStack

Review the contents of the Variable Stack Pools.

This command accepts no parameters

Remarks: This command is implemented in support of c/XDC. If you don't have the c/XDC feature installed, this command won't help you.

An LE compliant subroutine generally has access to up to three variable pools:

- A Global Pool of constants;
- a Subroutine Pool of variables declared for the current routine, subroutine or function;
- a Local Pool of variables declared only for the current block of code.

As one subroutine calls another, new Subroutine Pools and Local Pools are created, and the variables declared in the calling routine are generally unavailable to the called routine. The management of these variable pools is accomplished through structures known as Lanquage Environment Stack Frames.

The LIST VSTACK command displays information about these Stack Frames as follows:

Stack Frames are displayed from oldest to newest.

For each frame:

- The resume address is displayed for the routine that owns the frame;
- Information is displayed about each variable pool that is owned by the frame;
- Shortcut Entry Fields are provided that can be used to display resume address locations or variable pool storage.

For each pool, an automatic equate is generated conveying the following information:

- The sequence number of the stack frame;
- The type of variable pool (Global, subroutine or local);
- The location of the variable pool;
- The length of the variable pool.

If you're an experienced z/XDC Assembler user, then you will see a similarity – even a parallel - between the List VStack and List TAsks commands.

More Info: Enter Help List VStack.

List WTor

Display whether (or not) z/XDC's Cross Domain Facility will issue a WTOR command to the system console for signon wait cancellation.

This command accepts no parameters.

More Info: Enter Help COmmands List WTor. See also Help COMmands SEt WTor, Help CDF and Help Messages DBC640.

List XDc MAIntenance

Display the version level of the z/XDC product.

This command accepts no parameters. "List MAIntenance" is an alias.

Remarks: Much like the "Help/About" information given in Windows-based software, this command offers the version, and maintenance level of the currently running z/XDC product.
One of the most important parts of the output of this command is the Update Level, which will tell you which maintenance version of the product you are running.
You will see something like this: **PEM-2404D**. This tells us that Peter Morrison released this maintenance in the years 2024 ("24") in the month of April ("04") and this was the fourth release in that month ("D"). Armed with this information you will help us all if you report a problem with our products.

If you report a problem to Cole Software for resolution, we will need this information in order to help you.

More Info: Enter Help COmmands LISt MAIntenance. See also Help COmmands LISt XDc.

Display the cross-memory execution environment for any program running under any Request Block in the system.

rb-addr The address of a Request Block located in any accessible address space. If you have previously issued the LIST RBS command, you may use the resulting “RB#n” equates as parameters for this command.

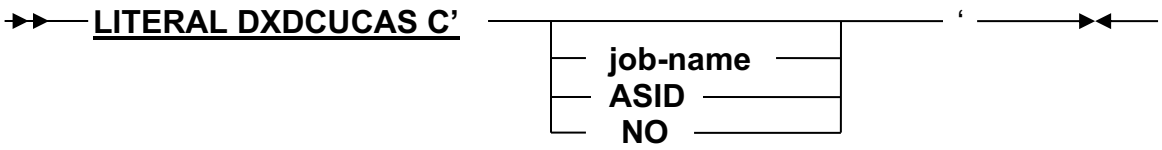
Note also that in order to see “other” address spaces you must be able to execute z/XDC APF-authorized (and with the appropriate security system rules in place). Once these conditions are fulfilled, you may issue SET ASID <job-name/ASID number, and enter Foreign Address Space Mode to look around inside another address space. Neat stuff!

omitted The cross-memory characteristics of both the error-level and retry level environments are displayed.

Remarks: This command displays:

- The Home Address Space Number;
- The current Task;
- The current Request Bloc;
- The Primary Address Space Number;
- The Secondary Address Space Number;
- The Address Space Control Mode (ASC-MODE);
- The Execution Space:
- The PSW Key mask (PKM);
- The authorization index (AX);
- And the Extended Authorization index (EAX) for both the retry and error levels.

More Info: Enter Help COmmands LISt XMs. See also Help COmmands SEt ASID.



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.

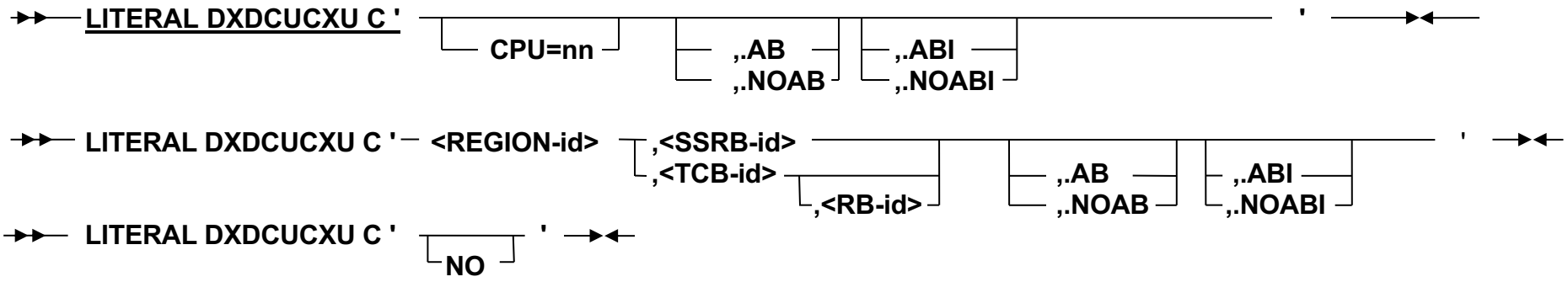
C'ASID' Pass an Address Space Identifier to d/XDC. It must be a hexadecimal value not wider than two bytes beginning with a decimal value (from 0-9). It may consist of numeric digits, alphabetic characters (including the hex digits A-F) or national characters ("@", "#" or "$"). This will become the Current 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. HOWEVER, if you know about dump/XDC's "CU" shortcut command you may change the CXU at will, at any time. This means that you may never need to issue this LITERAL.

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. See also Help SHortcutcommands.



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.

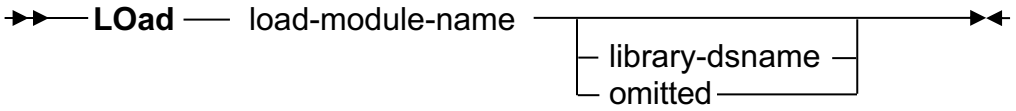
'NO' or ' ' This undoes any previous statement of DXDCUCXU, reverting to dump/XDC's automatic resolution of the CXU (if one existed in the first place).

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.



Issue the LOAD SVC for a given module.

load-module-name The name of the load module to be loaded. See Help COmmands SYntax Dsnames for information on how to specify a load module's name to z/XDC.

library-dsname The name of the load library dataset containing the desired load module. When stated, library-dsname must be a fully qualified, UN-quoted dataset name. Two-byte variable symbols are allowed to provide flexibility for current date time userid, jobname and more:

- ** - is replaced by a single asterisk;
- *D – is replaced by the current date formatted as "Dyymmdd";
- *J – is replaced by the Home Address Space's jobname;
- *O – is replaced by the Home Address Space's ownerid (from security). If no ownerid exists, then z/XDC proceeds as if *J had been specified instead;
- *P – is replaced by the current TSO profile prefix string. If no profile prefix exists, then z/XDC proceeds as if *U had been specified instead;
- *T – is replaced by the current time of day formatted as "Thhmmss";
- *U – is replaced by the current TSO userid. If no TSO useid exists (when z/XDC is running in batch) then z/XDC proceeds as if *O had been specified instead;
- *X – is replaced by z/XDC's current "clone" name (usually "XDC");
- *Y – is replaced by the current 4-digit year formatted as "Yyyyy".

Omitted The load module is loaded from standard system libraries (JOBLIB/STEPLIB, LPALIB, LINKLIB, etc.).

Remarks: The module to be loaded may now be either a PDS or a PDSE. When invoked, LOAD will search for a suitable copy of the module as follows:

- The address spaces Job Pack Queue is searched for a reentrant or reusable copy. If found then its use count is increased by 1;
- If a library name is specified in the LOAD command then it is searched;
- The task-, step- or job library is searched;
- The Systems Link Pack Queue is searched (the Link Pack Queue contains the MLPA and the FLPA but NOT the PLPA;
- The System's Pageable Link Pack Area (PLPA) is searched;
- The System's link-list library concatenation is searched.

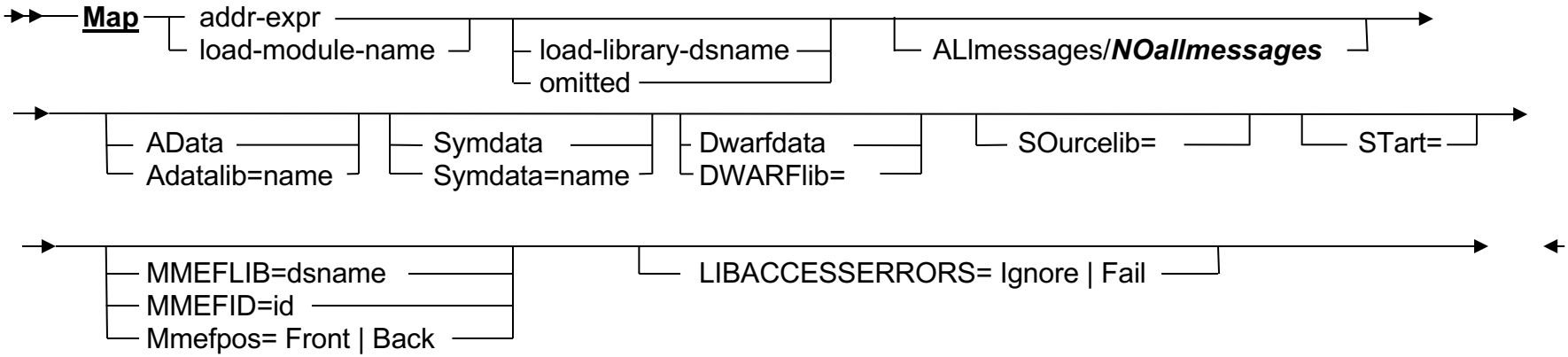
If it is necessary for the module to be loaded from disk, then it is brought into one of the following types of storage:

- If the module is not reentrant or if the library from which it is being loaded is not authorized, then the module is loaded into TCBKEY key storage (usually Key 8 storage).
- If the module is reentrant and if the library from which it is being loaded is authorized, then the module is loaded into Key 0 storage.

This command may not be used in Foreign Address Space Mode.

Examples: LO IEBGENR ←A copy of module IEBGENR is loaded from SYS1.LINKLIB.
LO MYPROG,*U.MYLIB.LOAD ← This command will load MYPROG from MYUID.MYLIB.LOAD (z/XDC substitutes the current TSO userid string for"*U".

More Info: Enter Help COmmands LOad. See also Help COmmands SYntax Dsnames.



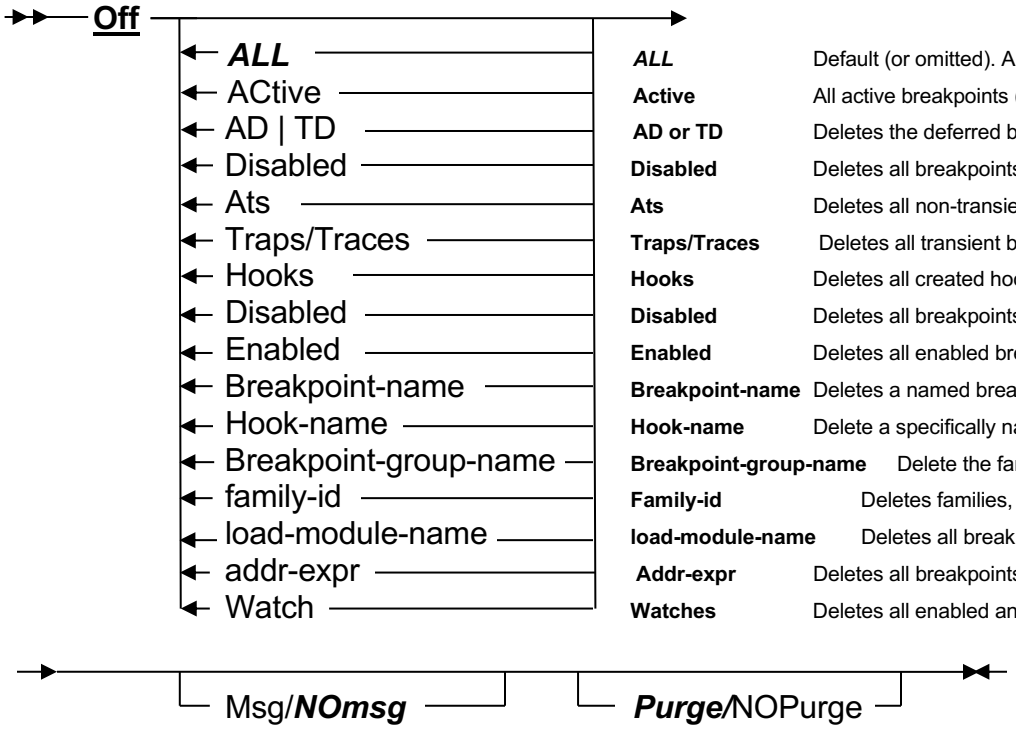
Build symbol maps or source image maps for High Level Language load modules.
Helper dialog: "MAP ?"
Shortcut command equivalent: "M"

addr-expr	An address expression that resolves to a place in the desired load module. z/XDC will return symbol or source image information for the module that the address expression occurs in. You can state module.csect if you prefer. When the module is mapped, the addr-expr is examined again and a csect map is loaded.
load-module-name	If stated as a pure load module name, then ONLY a module map will be loaded. To load a csect, end the module name with a dot, or use an address expression that resolves inside the csect or specify a csect name (ie, "module.csectname").
load-library-dsname	A load library from either a PDS or PDSE as a catalogued, fully qualified, UN-quoted name. May NOT be the name of an ADATA library or GOFF file. Two-byte variable symbols are allowed for each qualifier. See Help COmmands SYNtax DSNames
2nd parm omitted	z/XDC will attempt to located symbol or source image information on its own.
Allmessages/noallmessages	Map can generate a LOT of messages as it searches for your map. The default action, "noallmessages" surpresses them.
AData	This parameter coerces the Map command to create ADATA information (SYM data is not considered).
Adatalib=name	A dataset or library name where z/XDC is to search for ADATA. If unspecified, then the current MAPLIBS list is searched. See Help MAps AData MAPlibs for more information.
Symdata	This parameter coerces the Map command to search for SYM records, rather than ADATA (ADATA is not considered).
Symdata=name	The name of a load library that contains the load module to be mapped, using SYMDATA.
Dwarfdata	Coerces z/XDC to load DWARF Data. ADATA and SYM data are not considered.
DWARFlib=	The name of a partitioned dataset (PDS or PDSE) that contains the a member whose name matches the csect to be mapped. It must contain DWARF data.
SOurcelib=	A synonym for DWARFlib=libr. This parm will assist with setting up a LIBRARYLISTS redirect (see Help COmmands LIBrarylists).
STart=	The load point of a privately loaded module (one that has no CDE). When you use "ST=", the first parameter of the command must be the name of the load module, not an address-expression. For example "MAP MYMOD START=24CDC".

Using MAP with XDCMMEF program output:

When you receive a dump from a down-stream or customer site, and when that dump was generated on a system other than your own, you often have to guess at the layout of the foreign system's storage. The freeware program XDCMMEF available on the Cole Software website (https://www.colesoft.com/support/module-extract-utility) can be downloaded and run on the foreign system. This extracts "boundary" information (no content is included) and is highly compressed. If the people at the foreign system send you the output of XDCMMEF, then dump/XDC can fold this into the dump, giving you a much clearer picture of the foreign system's layout. NORMALLY, dump/XDC can fold XDCMMEF's output into the dump seamlessly. Other times you may have to help it along. The following parameters help you to accomplish that.

MMEFLIB=ds	Causes dump/XDC to create a DEFINE MMEFDSN to add to the MMEF datasets list.



ALL	Default (or omitted). All breakpoints are deleted.
Active	All active breakpoints (except ATX breakpoints and HOOKs) are removed.
AD or TD	Deletes the deferred breakpoints created by ADeferred and TDeferred commands.
Disabled	Deletes all breakpoints that have been previously disabled (via the "X" line command).
Ats	Deletes all non-transient breakpoints (those defined by AT and ADeferred commands).
Traps/Traces	Deletes all transient breakpoints (those defined by TRACE, TRAP and TDeferred).
Hooks	Deletes all created hooks.
Disabled	Deletes all breakpoints that have been previously disabled (via the "X" line command).
Enabled	Deletes all enabled breakpoints.
Breakpoint-name	Deletes a named breakpoint such as "AT0000nc" or "TR0000nc".
Hook-name	Delete a specifically named HOOK.
Breakpoint-group-name	Delete the family of breakpoints such as "AT0004".
Family-id	Deletes families, such as "*0003" regardless of their "AT" or "TR" prefix.
load-module-name	Deletes all breakpoints in the load-module (except ATX breakpoints).
Addr-expr	Deletes all breakpoints at this address.
Watches	Deletes all enabled and disabled Trace Watch commands.

Remove breakpoint(s) with any specified level of granularity.

Line mode equivalent: "O"

msg/NOmsg Enable or suppress z/XDC's messages as it executes OFF.

Purge/Nopurge Purge (default) will remove the breakpoint(s) from existence. Nopurge will merely disable them, leaving their definitions intact. You may later re-enable them with a subsequent "X" short-cut command.

Remarks The OFF command removes breakpoints with a surprising amount of granularity. However, keep in mind that one can also use the "O" short-cut command to accomplish the same result from a LIST BREAK display and/or wherever a breakpoint definition appears in a formatted z/XDC display. Also, you may use the "X" Line Command to "toggle on/off" a breakpoint definition, retaining the breakpoint for later use without deleting it.

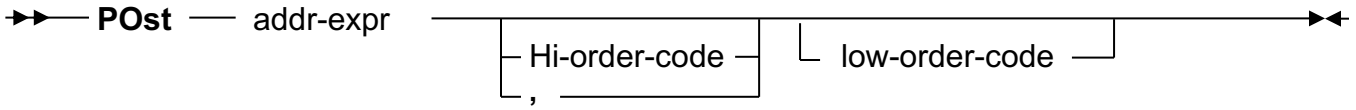
Examples:

O ←All breakpoints are deleted.

O AC ←All non-deferred breakpoints are deleted (except ATX breakpoints – which are persistent until explicitly deleted by name or the "O" shortcut command).

OF HOOK ← All dynamically created Hooks are deleted (not static ones created by the #XDCHOOK macro or cxdchook() function).

More Info: Enter Help COmmands Off. See also Help COmmands SEt Breakpoints. See also Help COmmands LISt BREakpoints.



Post an Event Control Block with a given code(s).

addr-expr An address expression that resolves to a valid ECB.
hi-order-code Two hexadecimal digits stating the code to store into the left-most byte of the ECB.
, Comma – A “place-holder” for the first parameter, which would make the next parameter the low-order-code.
low-order-code an address expression that is automatically resolved into a three-byte value stored into the low-order three-bytes of the ECB. If omitted, X’000000’ is stored.

Remarks: If you ever need to Post an ECB manually in z/XDC, here’s the tool with which to do so. Apparently one may not issue POST in 64-bit storage. Under the covers, z/XDC is using its ZAP command to make this change.

This command may be used in foreign address space mode.

Examples: POST .MYECB,FF ← The value X’7F000000’ is stored into the ECB located at “.MYECB”.
POST R10?,,R5% ← the EDB pointed to by R10 is to be posted. The value X’40’ is posted into the first byte of the ECB. The 24-bit address pointer contained in R5 is stored into the last three bytes of the ECB.

More Info: Enter Help COmmands POst.

PROfile

Access the Profile Facility.

This command accepts no parameters

Directly invoke the PROFILE facility's panel structure. This form of the PROFILE command accepts no parameters.

Remarks: A great deal of z/XDC's session and window characteristics can be controlled from the Profile Facility.

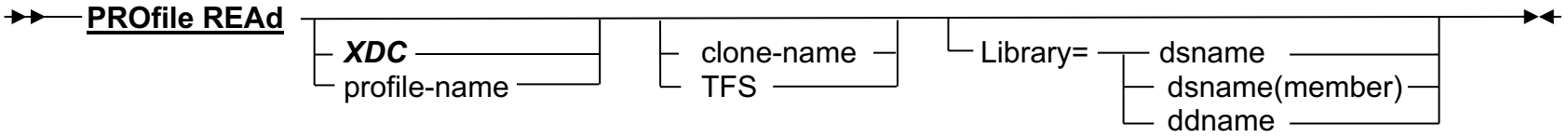
Behind the Profile facility, there are a huge number of SET commands that affect session characteristics. Profile allows you to review and change them symbolically, rather than having to know about the SET commands. It's a great time-saver.

Profiles control PFKey settings, screen colors, display window layouts, session logging and a great many other things. You can create a desired session layout and save this by name (via the Profile SAVE command). Later, you can recall this or any other profile by its unique name (via the PROfile REAd command). If desired, you may recall the "factory default" Profile at any time by issuing "PROfile RESet".

If you use c/XDC, then a very good profile to invoke is PROFILE RESET CWIDE (for wide terminal geometries), or PROFILE RESET C80 (for narrow terminal geometries).

You can have many named profiles, each of which is suitable for a particular debugging scenario (as many as will fit in the user's ISPF profile library). If the user wishes to use his or her favorite profiles when working on batch jobs via the Cross Domain Facility, then you must add a DD card (XDCPROF or ISPPROF) to the batch job's JCL that points to the ISPF profile library containing the desired profile(s).

More Info: Enter Help COmmands PROfile. See subsequent pages in this reference to learn about Profile's other parameters.



Retrieve a previously-saved profile for use in the debugging session.

XDC Default action. If no profile-name is invoked, then the default system profile is read.

profile-name The name of a previously saved profile, 1 to 5 characters in length. The profile name may consist of alpha, numeric or the national characters ("**$**", "**#**", and "**@**"). Why only five characters? That's because z/XDC appends the prefix "XDC" (or the current "clone name") to all of your profiles. If profile-name is omitted, then the user's current default profile is loaded.

clone-name z/XDC allows different versions of itself to have names other than "XDC". If you have created a clone of your z/XDC system (such as "X21" for Release z2.1, or "X1D" for Release z1.13) then you may invoke a profile that was defined under that clone-name.

TFS z/XDC writes profiles only to profile libraries (such as ISPPROF), but can **read** installation-specific default profiles from profile libraries **and** tables (such as ISPTLIB). Specifying "TFS" tells z/XDC to load profile-name from ISPTLIB.

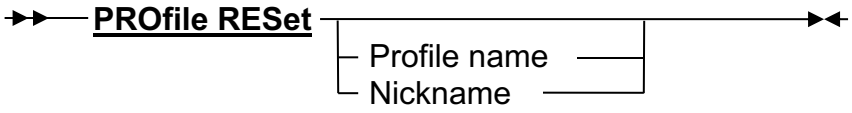
Library= If you want to load a profile from other than in the standard ISPC allocations, then you may specify one.

- **Dsname** Any FB-80 partitioned dataset.
- **Dsname (member)** Normally, profiles may be loaded only from library members named "xxxPNAME" (in your case, probably "XDCPNAME"). This allows for different "clone names" for z/XDC. However, using this parameter blows that away, and as long as the contents of (member) are z/XDC-generated Profiles, you are good to go. Because you have some compelling reason to do so, or you just prefer living a complex life.
- **DDname** If you have a profile library allocated to some arbitrary ddname, you can load your profile from this allocation instead of the dfailt (//XDCPROF, //ISPPFOR and //ISPTLIB).

Remarks: If you have created an ungainly mess on your screen, and you wish to get our "factory default" profile back, you may enter "PROfile RESet".

Examples: PRO REA TEST ←The user wishes to recall a previously-defined profile that is named "TEST1".
PROFILE REA FNOOB X1D ←The user wishes to recall profile "FNOOB" from the X1D clone of z/XDC .

More Info: Enter Help COmmands PROfile REAd. See also Help PRofiles Tlibprofiles. See also Help Commands LISt PROfile (which displays your current profile).



Summon the z/XDC "factory default" profile for use in the debugging session.

Remarks: z/XDC's "factory default" profiles are hard-coded within the product itself. In order to react to the proliferation of larger screen geometries and the advent of High Level Language support in z./XDC a number of new default profiles have been created.
If no name or nickname is provided, z/XDC will issue PROFILE RESET XDC, and give you the "factory default".

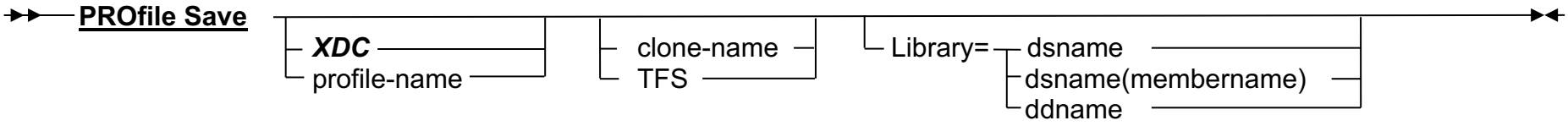
Profile name Can be any of the four names below:

- **A80** Intended for the Assembler environment with column width of 80 columns.
- **AWIDE** Intended for the Assembler environment with column width of 100 or more columns.
- **C80** Intended for the IBM C/C++/Metal C environment with column width of 80 columns.
- **CWIDE** Intended for the IBM C/C++/Metal C environment with column width of 100 or more columns.

Nickname Can be any of the four "nicknames" below:

- **ASM** z/XDC will configure the profile for Assembler using A80 or AWIDE depending on what it "senses".
- **C** z/XDC will configure the profile for C using C80 or CWIDE depending on what it "senses".
- **CEE** A synonym for nickname "C".
- **XDC/omitted** Pretty cool: z/XDC will configure for ASM or C depending on what it "senses".

More Info: Enter Help COmmands PROfile RESet. See also, the PROfile REAd command.



Store the current Profile for use in a later debugging session.

XDC If the word "XDC" or if this parameter is omitted, then current profile settings are saved as the user's default.

profile-name The name that you wish to assign to the profile, from 1 to 5 characters in length. The profile name may consist of alpha, numeric or the national characters ("**$**", "**#**", and "**@**").

clone-name z/XDC allows different versions of itself to have names other than "XDC". If you have created a clone of your z/XDC system (such as "Z21" for Release z2.1, or "X1D" for Release z1.13) then you may save a profile under that clone-name.

TFS z/XDC writes profiles only to profile libraries (such as ISPPROF), but can **read** installation-specific default profiles from profile libraries **and** tables (such as ISPTLIB). Specifying "TFS" tells z/XDC to load profile-name from ISPTLIB.

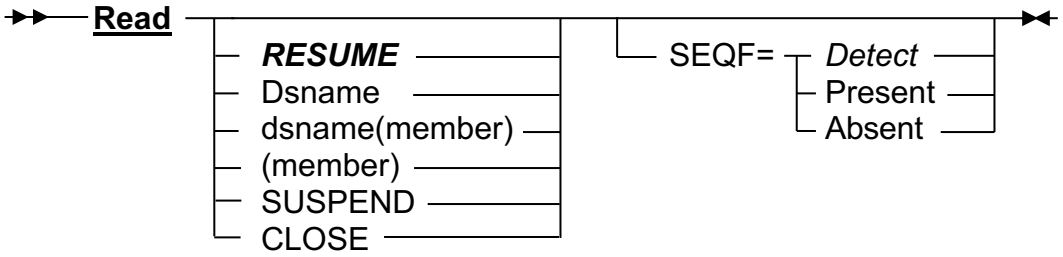
Library=

- **dsname** The name of a library other than the standard ISPPROF allocation.
- **dsname(membername)** The name of any desired member within dsname. When membername is specified, neither profilename or clonename may be given.
- **ddname** Cause the profile to be saved into any DD allocation other than the defaults (XDCPROF, ISPPROF or ISPTLIB).

Remarks: Profile is pretty cool. Imagine setting a hook or a breakpoint in a module and associating a PROFILE READ command with it. When the GO command executes the hook, or hits a breakpoint, your screen changes to the profile layout that is perfect for working on that part of your code.

Examples: **PRO S TEST1** ← The user wishes to save the existing session definitions under the name of "TEST1".
PROFILE SAVE FARFEL TFS ← The user wishes to save the existing session definitions under the name of "FARFEL" into ISPTLIB as a system default.

More Info: Enter Help COmmands PROfile Save. See also Help COmmands PROfile.



Cause z/XDC to obtain and execute commands from a previously created script.

RESUME Default. If a dataset has already been opened for reading, then was suspended (due to an error), execution is resumed.

dsname The name of a sequential dataset containing subsequent z/XDC commands to be read.

dsname(member) The name of a specific member of a partitioned dataset. It must be fully qualified and unquoted.

(member) If a previous Read command referenced a member of a partitioned dataset, then specifying Read (member) causes z/XDC to access the same PDS for the indicated member.

SUSPEND If a dataset is currently being processed, then execution is suspended. This parameter can only be issued from within the Read dataset itself.

CLOSE If a Read dataset has been suspended, then this parameter will close the dataset. If the user wishes to re-initiate a Read dataset that had previously been SUSPENDed, then the Read CLOSE command must be entered before the original READ command may be entered again.

SEQF= If you have decided to number the separate statements in your script, then by convention sequence numbers take up 8 decimal digits. But with the decline of the importance of sequence numbers, this may be vestigial. In any case, you can let z/XDC handle your sequence numbers for you, and NOT screw up your script by attempting to treat sequence numbers as commands in the script. SO, bottom line: it's best NOT to use sequence numbers in scripts. If you already have them in your scripts, you'll have to deal with them.

SEQF has two synonyms, each involving more key-strokes: SEQUENCEFIELD, and SEQFIELD.

SEQF= may have three parameters:

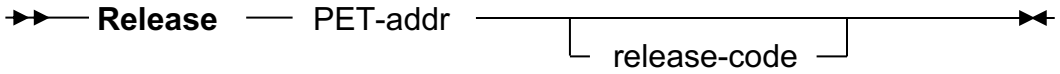
- ***Detect*** This is the default. z/XDC will do its best to determine the presence or absence of sequence fields.
- **Present** The sequence field is present in all records, and will be ignored by the READ command.
- **Absent** The sequence field is absent, and z/XDC will attempt to process everything it finds in the script dataset.

Remarks: Whenever you find yourself typing a series of commands over and over, DON'T. Take the time to create a script of those commands, and issue the READ command to execute them en masse! Scripts save tons of time! In many, if not most of our customers' shops there are one or two users who build and maintain scripts. Go make friends with that person.

Some rules about scripts:

- Scripts are "lists" of declarative commands. There's no conditional logic, but Dave insists that you can get a surprising amount done with simple declaratives (See Help SCripts Ourscripts for examples).
- If you want to investigate conditional logic (outside of scripts), check out HELP REXX.
- Scripts CAN be recursive. There is no reason that a script cannot itself issue the READ command.
- A script will not survive the "GO" command. If you want other things to happen after a "GO" command, make a second script.

More Info: Enter Help COmmands Read. See also Help SCripts Ourscripts and Help Scripts Ryoscripts (for "write your own").



Release a Pause Element Token.

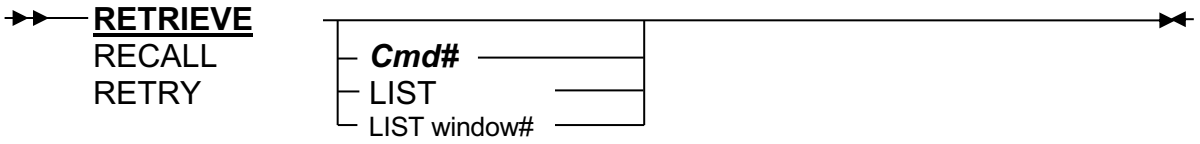
PET-addr An address expression resolving to a previously-created Pause Element Token.
Release-code A hexadecimal number from 0 to FFFFFF. Alternately you may use a decimal number with an N appended at the low end (example: “123N”).

Remarks: If z/XDC is executing non-authorized, then only PETS allocated in the same address space and with an authority level of IEA_UNAUTHORIZED may be released.

This command may be used in Foreign Address Space Mode.

Examples: RELease A0000000 100 ← Release a PET where the PET may be found at address A0000000. A release code of X’100’ is used.
REL A0000000~ASID(MYJOB1) 100 ← Release a PET where the PET can be found at address A0000000 in the address space “MYJOB1”. A release code of X’100’ is used.
REL 0000F1D0 Release a PET at F1DO. Good boy! Go for a walk, 61904!

More Info: Enter Help COmmands RELease.



Retrieve a previously-issued command from a Save Area to the command line, or retrieve ALL commands in the debugging session.

Omitted Default action: The most recently issued command is brought to the command line for editing or processing (by pressing Enter). In the "factory default" profile, RETRIEVE is set to PFKey 12.

Cmd# A pure integer either positive or negative. Returns the nth command in the debugging session. HOWEVER, using RETRIEVE 1 has the same action as using RETRIEVE without an integer (displays the last command entered). If you want to access a PRIOR command issue RETRIEVE –n (a negative integer). Think of the retrieve Save Area as a "stack". Retrieving a command from somewhere in the stack (RETRIEVE -10, for example), resets the pointer in the stack, and subsequent RETRIEVE commands will reference the updated pointer. The next RETRIEVE –n will go n back from the previous RETRIEVE location in the stack.

LIST This is SUPER handy! RETRIEVE LIST brings up ALL prior commands in the debugging session (I haven't been able to find an upper limit). The user may Select a command (with the "S" shortcut command) and bring it to the command line for further editing or processing (by pressing Enter). This is so cool that I've made it my default action for PFKey12 in all my z/XDC profiles.

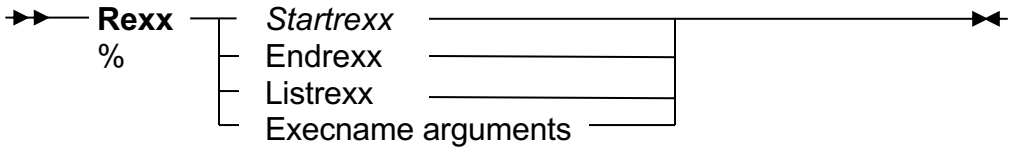
LIST Window# z/XDC assigns a number to each Display (or "Watch) Windows as they are created. The third Display Window created for the session would be number 3. If you issue RETREIVE LIST 3, then you will see a list of all the commands issued within that Display Window. I'm not sure I'd use this much of the time, but it's there.

Remarks: RETRIEVE may not be abbreviated, which is a minor annoyance. When Dave implemented RETRIEVE LIST, that got my attention. Suppose you wanted to set up a breakpoint or trace command and associate other display commands with it. And, suppose further that you'd like to navigate your program's logic and continue to issue the same display commands as you went along. This is NEAT: T BY 'LIST PSWE F;;LIST REGS;;F PSWE! 10;;RETRIEVE'

The command string above traces to the next Branch mnemonic that WILL branch. Then z/XDC displays the formatted PSW,(two semi-colons create a black line of "whitespace), the General Registers and shows you a Format of your code for ten lines. Then it issues a RETRIEVE, which brings the whole string back to the command line. You press enter and when you're done ALL this information will be in the debugging session log. That's hugely cool.

Examples: RETRIEVE ← Brings the most recently-issued command back to the command line for editing/processing.
PROFILE LIST ← Displays the entire "stack" of commands issued for this debugging session .

More Info: Enter Help COmmands RETRIEVE.



Initialize, shut down, Issue List Rexx (in an impractical way) or execute a REXX exec under z/XDC.

Startrexx Ignore this parameter. The Rexx command will automatically initialize the interface. So, don't bother.

Listrexx The same as issuing the LIst REXx command. Displays information about the REXX/XDC Interface.

Endrexx Ignore this parameter as well. If the z/XDC END command is processed the environment will be shut down – unless you enter END KEEP.

Execname arguments NOW we get to the good stuff. You name the exec to be run. You may specify name, or name(member). Arguments may be passed to the exwc, BUT argument may not contain semi-colons or colons unless the string is enclosed within single quotes (‘). That is because z/XDC is already making use of semi-colons and colons.

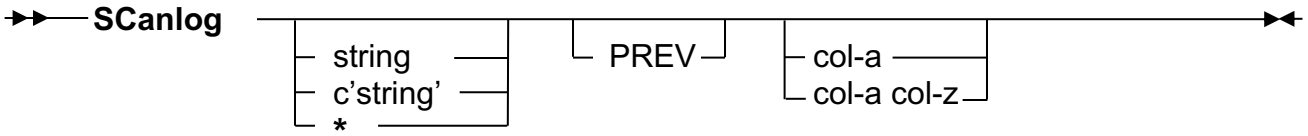
Remarks: The % character is a synonym for the REXX command.

z/XDC's REXX interface provides several built-in functions that your user-written exec can call. These functions:

- Control the source of REXX input and output;
- Parse address expressions;
- Extract or zap data in storage or registers;
- Extract or alter the retry-level PSW (whoa!);
- Send message to z/XDC's Display;
- Request input from a user.

Examples: Sorry, I can't find any examples to copy from.

More Info: Enter Help COmmands REXx. See also Help Rexx XDCServices. See also Help Rexx.



Search the session log for a particular string of characters.

String The character string to be searched for. May be enclosed in quotes. If the string contains spaces or commas, then quotes MUST be used.

c'string' C indicates that the string you're looking for is to be case-sensitive. You must enclose the case-sensitive string with either single or double quotes on each end of the string.

- String may contain any characters, including blanks, commas and the quote character.
- If a quoted string does contain the quote character, then it must be double as in the word don''t. This, however, will make the search case insensitive. So I suppose I wouldn't try to include phrases in my string that are conjunctions or are possessive.

***** Or omitted. Search for a previously stated character string again. That is, reissue the search for the "same string" again.

PREV Forces z/XDC to search for the string "upward" or "previously" in the log.

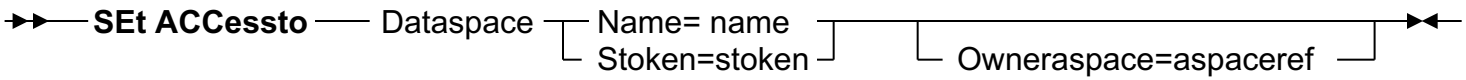
col-a Column areas to scan. If col-a is stated without col-z then SCanlog will search for the string value beginning ONLY in column-a.

col-z If col-a is and col-z are both specified, then the string will be found if the first character falls between the two column values.

Remarks: Once set, column limits remain unless reset. That is, if the user wishes to use other column definitions later they must be entered separately.

Examples:
SC 'COLE' PREV ← Searches for the character string "COLE" upward in the session log.
SC 'COLE' PREV 24 33 ← Searches for the character string "COLE" upward in the session log, and occurring only within columns 24-33.

More Info: Enter Help COmmands SCanlog. See also Help COmmands SYntax Characterstrings.



Establish z/XDC's access to a dataspace.

Dataspace Signals the z/XDC parser that a dataspace name will be specified.

Name= The name of the dataspace.

Note: The Name and Stoken operands are mutually exclusive. Use one or the other.

Stoken= A hexadecimal number exactly 16 digits long, including leading zeroes.

Note: The Stoken and Owneraspace are mutually exclusive. Use one or the other.

OWneraspace= If omitted the home address space is assumed as the owning address space.

If stated, this indicates the address space that owns the dataspace. "aspaceref" can be :

- The address of an ASCB;
- A four-digit ASID-number in hexadecimal (or decimal number if suffixed by "N");
- A keyword such as HOME, PASID, etc.;
- An address expression that resolves to an ASID.

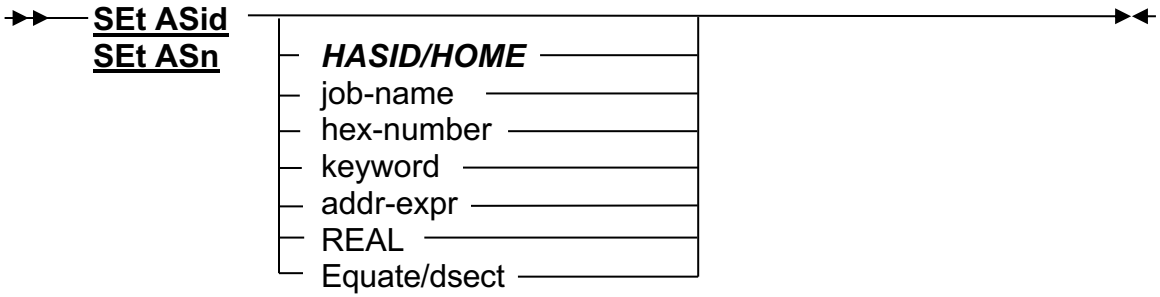
Remarks: This command issues an ALESERV macro that attempts to add the dataspace to z/XDC's home task's Access list (DU-AL). A couple of z/XDC commands may be used to access Stoken values: LIST DSPACES and LIST ACCESSLISTS.

Examples: SE ACC d na=BOB ← if the current address space has any dataspaces named "BOB", then z/XDC will add them to its access list.

SET ACCESSTO DATASPACE NAME=00000IEF OWNERASPACE=JES2 ← This got rejected (for me) by ALESERV – probably for a very good reason. I've got no business messing with JES2, do I?

S ACC D N=00000IEF O=JES2 ← Just for fun, here is the minimum abbreviation of this command.

More Info: Enter Help COmmands SEt ACCessto Dataspace. See also Help COmmands LISt DSpaces and Help COmmands LISt ACCESSLists.



Use Foreign Address Space Mode to change from one address space to another.

HASID/HOME **(or omitted)** If one is currently in another address space, z/XDC will return the user to the "home" address space. So, "SET ASID" gets you "home".

job-name The name of a particular job, such as "JES2", **"**VTAM", "CICS" and "DB2".

hex-number The 1 to 4-digit hexadecimal number that identifies an existing address space (an index into the Address Space Vector Table).

keyword **HASID** - The user's "home address space". Mentioned above as the default. **PASID** - The retry level of the Primary address space. **SASID** - The secondary address space. **EPASID** - The error-level instance of the Primary address space. **ESASID** - The error-level instance of the Secondary address space.

Addr-expr A address expression that points to an ASID number.

REAL The user wishes to display (and perhaps zap) real storage. This choice is NOT for amateurs, and (of course) is subject to security rules AND your personal discretion. Tread carefully here.

Equate/dsect An equate or dsect that has been assigned to represent storage in an address space.

Remarks: **The user must have APF-authorization in order to use this command (as well as appropriate permission from the security system).** Foreign Address Space Mode (AKA "FASM") allows the user to investigate programs running in other address spaces on an ad-hoc basis. This is an extremely powerful and useful facility.

If you wish (and if allowed by your security system), you may access another address space using FASM, and HOOK a running program, then get out of FASM and connect to it using Cross Domain Facility (Option 3 from the z/XDC Startup Panel). Now HOLD ON: This is powerful/scary stuff, and you really must know what you're doing when you use FASM to hook another address space. Proceed with caution. If you crash anything important, it will be on YOU, not z/XDC.

More Info: Enter Help COmmands SEt ASId. Also Help CDf Fasm, and Help FUNctions ASId. In addition, see Help COmmands SYntax ASids. While you're at it, see Help Security.

SEt AUTH scbaddress

Make a problem-state z/XDC debugging session APF-authorized.

scbaddress An address expression that resolves to a STAE Control Block (SCB) queued from any TCB located in any address space in the system (Security System permitting).

Remarks: SET AUTH is the first command in a two command process by which an already *APF-authorized debugging session* can make a non-authorized debugging session authorized. The other command in the process is to switch to the target debugging session that you issued SET AUTH within, and issue “GO NOWHERE”. GO NOWHERE causes Recovery Termination Manager to get control and the target debugging session will thereafter run APF-authorized. This does NOT confer APF-authorization on your program – only that z/XDC itself will be authorized.

SET AUTH will not work for those cases in which the target debugging session is not running as an Abend Recovery routine. This includes:

- z/XDC sessions which are debugging SRB routines (which are already running authorized)
- z/XDC sessions which are debugging routines protected by Functional Recovery Routines (which are already running authorized).
- z/XDC sessions which are running as Trap Handlers due to TRAP2-type breakpoints.

In other words, SET AUTH will work only for debugging sessions that are running as Problem State, with ESTAE-type recovery routines.

Examples: How to do this process, from Dave hisownself:

First, use LIST TASKS [aspacename] to produce a tasks report and create TCB#n equates for all tasks in the target address space. This is running non-authorized.

Then examine the tasks report to decide under which task your target debugging session is running. You might find the LIST RBS TCB#n command to be helpful here.

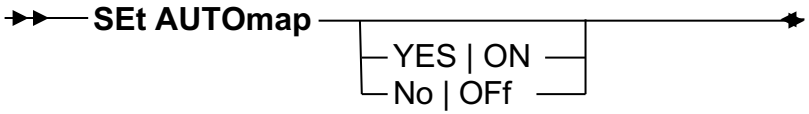
Then use LIST ESTAES TCB#n to produce an SCB report and create SCB#n equates for each ABEND recovery routine's STAE Control Block currently queued to that task.

Now examine the SCB report to decide which SCB describes the z/XDC debugging session that you want to change. It usually will be the one for the exit named XDC or XDCCALL and captioned as being ACTIVE.

Then issue the appropriate SET AUTH SCB#n command to begin the authorization process.

Finally, navigate back to the target debugging session and issue “GO NOWHERE”.

More Info: Enter Help COmmands SEt AUTH. See also Help COmmands GO



Enable or disable the AUTOMAP feature in z/XDC.

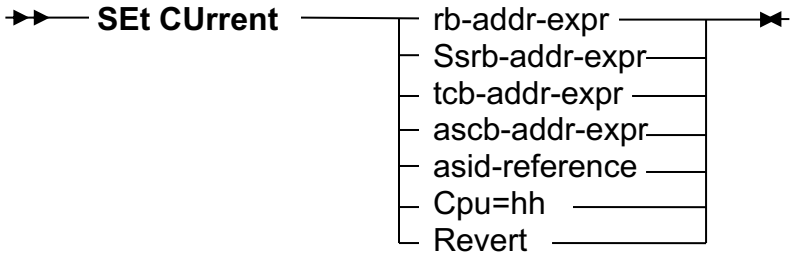
Yes | ON Enables the Automap feature.
No | Off Disables the Automap feature.

Remarks: This recent but valuable innovation allows z/XDC to automatically attempt to map one's source code (in ADATA for Assembler, or DWARF for C). If ADATA or DWARF libraries are established in z/XDC, then you will never have to issue SET MAPLIBS or MAP again. And, if your code passes from Assembler to C and back again, MAP always works. This is really convenient, and comes set ON in our factory default profile.

Mapping is a hugely good thing in z/XDC. The ability to see your source code is one of the great advantages – IF you have access to ADATA and/or DWARF in your environment.

SEt AUTOmap can be found in your Profile. To access it, enter "Profile", then navigate to "Display/Change AUTOMAP PRINT and Other Settings".

More Info: Enter Help COmmands SEt AUTOmap. For a deeper dive, see Help COmmands Map.



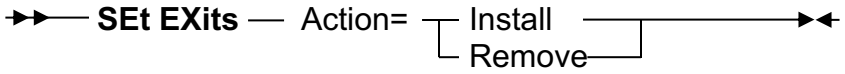
Change the Current eXecution Unit while in a dump/XDC session.

rbaddr-expr	The CXU is set to the designated Request Block, and the Current Address Space is set to the home address space.
ssrb-addr-expr	The CXU is set to the designated suspended Service Request Block, and the Current Address Space is set to the home address space.
tcb-addr-expr	The CXU is set to the newest Request Block for this Task Control Block, and the Current Address Space is set to the home address space.
ascb-addr-expr	The CAS is set to the nominated address space. The CXU is not set.
asid-reference	The CAS is set to the nominated address space. The CXU is not set.
CPU=hh	A two-digit Hexadecimal number that identifies a CPU engine. IF you are working on a standalone dump, you can designate whatever thread was running on the specified CPU as the CXU.
REVERT	Undo a previous SEt CUrrent command. dump/XDC restores the original CXU that IT found in the dump.

Remarks: dump/XDC uses two concepts to identify which portion of a dump was running at the time of the dump. First, it identifies the Current Address Space (the "CAS) and then it looks for the Current eXecution Unit within the CAS. It does this stepwise process during the initializing process and is most often correct. HOWEVER, you have the ability to over-ride dump/XDC's selection with the SEt CUrrent command – and with the "CU" shortcut command. You may also set the CAS and CXU with an IPCS Literal. I think SEt Current and the "CU" shortcut command are far easier.

Examples: LIST TASKS <address-space>, then LIST RBS TCB#n, then SET CURRENT RB#n ← This is the easiest way for me to do this.

More Info: Enter Help COmmands SEt CUrrent. See also Help Dumps EXEcutionthread. See also Help SHortcutcommands.



Enable or disable z/XDC's default miscellaneous user exits.

Install Turn 'em on.
Remove Turn 'em off.

Remarks: You can install or remove z/XDC's exits manually (if z/XDC doesn't load them automatically for you). This module is named, "XDCEXITS". If z/XDC has already loaded XDCEXITS, the SEt EXits command will have no effect. Once a SEt EXits command has been accepted, the exits module will take effect AFTER the next Trace, STep or GO command.

Examples: SE EX A=I ← Install XDCEXITS.

More Info: Enter Help COmmands SEt EXits. See also Help EXIts.

SEt ILc

Control whether z/XDC will perform instruction length checking when the user overtypes Assembler mnemonics. This is z/XDC's default action. This command is the converse of the Set NOIlc command.

This command accepts no parameters

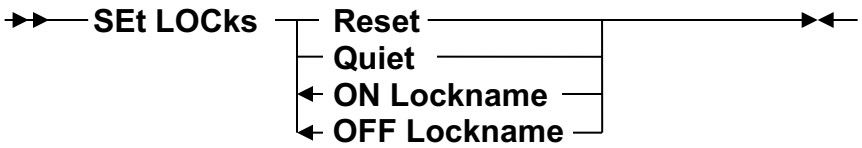
Remarks: When the user chooses to overtype an instruction mnemonic with another mnemonic using the "Z" line command (for the ZAP command), z/XDC is normally set to "refuse" the request if the instruction being typed in is of a different length than the original instruction. In other words, Set ILc will not allow instruction mnemonics of different length to be overtyped onto other mnemonics. This is usually a Very Good Thing.

For example, with Set NOIlc in effect one may overtype an "MVC" mnemonic with a "ST" - an instruction of shorter length. z/XDC makes no provision for "shifting" the contents of memory to deal with the change in the length of this instruction, so the result would probably not be what the user intended. Set ILc is the system default.

Set ILc and Set NOIlc have no effect on zaps that are made directly to the hex portion of a display. These commands have effect only when mnemonics are overtyped.

Set ILc can be found in your Profile. To access it, enter "Profile", then navigate to "Display/Change READ ZAP and Parse Order Settings".

More Info: Enter Help COmmands SEt ILc. Also, see Help Commands SEt NOIlc.



Control the set of system locks that z/XDC will acquire or re-acquire if execution of the user's SRB routine is resumed.

Reset Restores all locks to the state found when z/XDC last got control.

Quiet In normal operation, z/XDC will issue a List LOCks command after a Set LOCks command executes. The Quiet parameter suppresses the List LOCks command.

Lockname z/XDC supports these locknames:

- LOCAL – For ON processing if a CML Lock was to be turned on, it will now be left off. For OFF processing, if any CMS lock was to be turned on then it will be left off.
- CML – For Cross-Memory Local locks. For OFF processing you merely specify "CML".
- CML=asid - For ON processing, this form must be used. "Asid" specifies an address space name, asid number or any other parameter recognized by z/XDC's SEt ASid command.
- CMS
- CMSEQDQ
- CMSLATCH
- CMSSMF
- CMSALL

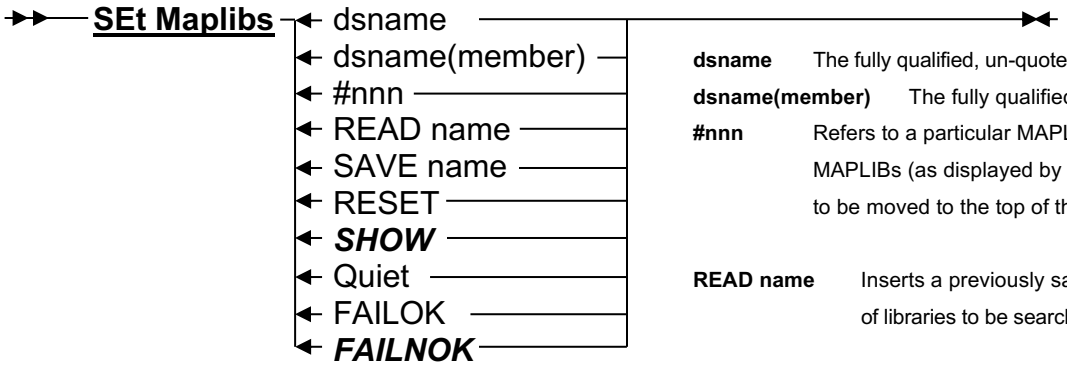
  These are various cross memory services locks currently recognized by IBM's SETLOCK macro. For On processing, these locks are mutually exclusive. If any on is turned on, then all the others are turned off. For OFf processing, no special notes are needed. There are no dependent locks to be turned off.
- CPU — This is the CPU lock. It does not have any prereq., dependent or mutually exclusive locks. You just specify it.

On/OFF Specifically-named locks are toggled on or off individually. This parameter and the ON/OFf sub-parameter may be entered any number of times. You may specify ON/OFF either before or after lockname. Stick to one usage convention or you risk confusing yourself.

Remarks: This command may only be used when z/XDC is running as an FRR (Functional Recovery Routine). All parameters are optional but ONE parameter must be specified. If used with "lockname" you may specify ON or OFF preceding OR following the lock name.

Examples: Set LOCks ON LOCAL ← The local lock is turned on
S LOC R ← Reset locks to the state found when z/XDC last had control.

More Info: Enter Help COmmands SEt LOCks. See also Help DEbugging Frr. See also Help COmmands LISt LOCks. See also Help Debugging LOstlocks.



dsname The fully qualified, un-quoted name of a sequential or partitioned dataset.

dsname(member) The fully qualified, un-quoted name of a member of a PDS.

#nnn Refers to a particular MAPLIB's unique number in the list of active MAPLIBs (as displayed by the List MAPlibs command). Causes #nnn to be moved to the top of the search order in the list. You must preface this parameter with "#".

READ name Inserts a previously saved MAPLIBs list into the front of the currently active list of libraries to be searched for ADATA information. It does not replace the active list.

Create or add to the list of datasets that z/XDC will search for ADATA (source statement) information.

SAVE name Causes the currently defined list of MAPLIBs to be saved under a name, so that they can be "READ" later.

name For both the READ and SAVE parameters, "name" is 1-8 characters, including "_", "@", "#" and "$".

RESET The currently active MAPLIBS list is cleared. Previously saved lists of MAPLIBs are not deleted.

***SHOW*/Quiet** Determines whether the list of defined MAPLIBs will be displayed when the List MAPlibs command is issued, or not. SHOW is the default. NOSHOW is a synomym for Quiet.

FAILOK/*FAILNOK* If SEt Maplibs is being executed from within a script that is processed by z/XDC's READ command, then this parameter controls whether the READ command will abort on error.

Remarks: z/XDC automatically maps your program if it can find the ADATA anywhere in your search order. SET AUTOMAP ON is active in the "factory default" profile that comes with the product, so once you create a default MAP library(s) you won't have to either issue SET MAPLIBS or the MAP command again. It's a great convenience, but if the ADATA dataset cannot be found in your search order, then you will find SET MAPLIBS very useful.

After SEt Maplibs is issued, z/XDC's MAP command will read source statements for use in Formatted displays. Lists of MAPLIBs can be saved in the user's profile as the user's default. Lists of MAPLIBs may also be established by the systems programmer as a system-wide default.

Here is the command sequence to make a set of maplibs your personal default, in your default profile:

1. DELETE MAPLIB DEFAULT ← This wipes out anything you had already!
2. SET MAPLIBS <as.many.datasets.as.you.like>
3. SET MAPLIB SAVE DEFAULT
4. PROFILE SAVE

Another way to avoid having to issue SET MAPLIBS is to add a DD name of XDCMAPLB to your JCL. z/XDC will search this dataset name for ADATA and will automatically map it if the dataset contains your ADATA.

Examples: S M DBCOLE.XDCZ17.XDCADATA ← When a subsequent MAP command is issued, this dataset will be searched for ADATA information.

More Info: Enter Help COmmands SEt Maplibs. For important overview information, enter Help MAPs AData.

SEt NOIlc

Prevent instruction length checking when the user overtypes (and changes) Assembler mnemonics. This is the converse of the Set ILc command.

This command accepts no parameters

Remarks: When the user chooses to overtype an instruction mnemonic with another mnemonic using the "Z" line command (for the ZAP command), z/XDC normally will "refuse" the request if the instruction being typed in is of a different length than the original instruction. That is z/XDC's default behavior.

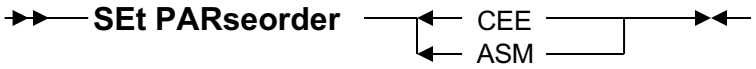
Set NOIlc will allow instruction mnemonics of different length to be overtyped onto other mnemonics. Vaya con Dios.

For example, one may overtype an "MVC" mnemonic with a "ST", an instruction of shorter length. z/XDC makes no provision for "shifting" the contents of memory to deal with the change in the length of this instruction, so the result would probably not be what the user intended. Set ILc is the system default. *If you use Set NOIlc, you had better know what you're doing.*

Set ILc and Set NOIlc have no effect on zaps that are made directly to the hex portion of a display. These commands have effect only when mnemonics are overtyped.

Set ILc can be found in your Profile. To access it, enter "Profile", then navigate to "Dipslay/Change READ ZAP and Parse Order Settings".

More Info: Enter Help COmmands SEt NOIlc. Also, see Help Commands SEt ILc.



Control the order in which language “protocol” (Assembler vs. C) z/XDC and c/XDC will use to parse a string.

CEE z/XDC and c/XDC will evaluate the string in terms of C/C++ and Metal/C languages first.
ASM z/XDC and c/XDC will evaluate the string in terms of Assembler language first.

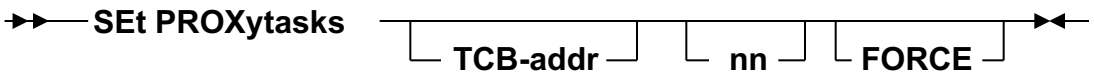
Remarks: The syntax rules for variables differ from one language to the next. Consequently, one parser cannot be used to parse a variable for all languages. Set Parseorder allows you to set which Parsers are called for parsing a string, the order in which they are called and which Parser will generate an error message should they all fail. z/XDC’s CEE and CWIDE profiles handle these settings for you and you can poke around to see how they work.

This setting can also be found in your Profile. If you’re curious, enter “PROfile”, and navigate to “Display/Change READ ZAP and Parse Order Settings”.

If you happen to be a C-centric programmer using the c/XDC Feature, then you can issue PROFILE RESET CWIDE (presuming a wide terminal geometry). Parsing and screen layouts will change to favor your work in C.

Examples: SE PAR CEE ← z/XDC will attempt to interpret the string as C-specific.
SET PARSEORDER ASM ← z/XDC will “bias” towards Assembler language as it parses a string.

More Info: Enter Help COmmands SEt PARseorder. See also Help COMmands LISt PARseorder.



Create Proxy Tasks that z/XDC may use in debugging Service Request Block-mode programs.

TCB-expr An address expression that resolves to the location of a Task Control Block that will anchor the Proxy tasks. If you issue "LIST TASKS" you may subsequently use the TCB#nn equates that are defined by List Tasks as equates for TCB-addr.

nn The number of Proxy tasks that you wish to create. The acceptable range is 1-10. The default value is 2.

FORCE Before attaching Proxy tasks to a given TCB, z/XDC checks to see if the task is in the process of ending or abending, or if it has ended or is frozen. If so, then this command will fail. You can override this decision with the Force parameter, which will cause z/XDC to attempt to attach the Proxy Tasks anyway.

Remarks: When debugging Service Request Block-mode programs, z/XDC runs as a Functional Recovery Routine. When you debug FRR code, z/XDC runs in two "phases":

- The "local" phase is the address space where the SRB-mode code runs.
- The "remote" phase is the Home address space where z/XDC can attach a Task to run the debugging session. This is why SEt PROXytasks exists.

When a breakpoint or abend is sensed in the SRB-mode program, z/XDC suspends the SRB, deletes locks, and communicates with debugging address space to request debugging services. When you issue a "GO" or resume the SRB, locks are set, and the SRB resumes in its normal environment.

If z/XDC is started APF-authorized it will automatically ATTACH two Proxy Tasks for you – BUT z/XDC can choose an "unfortunate" task for the purpose. Hence the need for this command.

Set Proxytasks allows you to prepare for debugging SRB-mode code. You may specify up to ten Proxy Tasks that will receive control when z/XDC is debugging your SRB-mode program.

There is no "LIST PROXYTASKS" command. Instead, issue List TAsks, and you'll see all tasks, including Proxy Tasks.

Examples: S PROX ← 2 Proxy Tasks will be defined for the current Task in the Task Control Block chain for the Address Space.
L TA;S PROX TCB#3 9 FORCE ← You issue List Tasks (to get the TCB#n equates defined) and then demand that z/XDC create 9 Proxy Tasks under TCB#3, with the FORCE parameter.

More Info: Enter Help COmmands SEt PROXytasks. See also Help DEbugging Frr.

Control the addressing mode bit of the Program Status Word (PSW) or the PSWE (the “error level” PSW).

24 24–bit addressing mode is enabled.

31 31-bit addressing mode is enabled.

64 64-bit addressing mode is enabled.

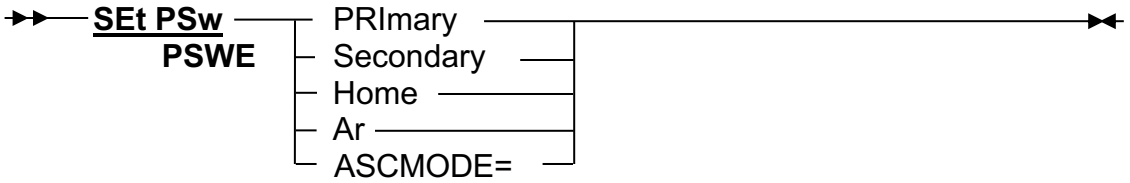
Amode=nn For those who love additional typing, type the keyword “Amode” and then “nn”, where “nn” is one of these: 24/31/64.

Remarks: z/XDC offers many ways in which to affect the state of the Program Status Word. See the following pages for other ways in which the PSW may be changed. Set PSW and its functions are very powerful, and worth handling with caution.

Note: If all you want to do is change the Instruction Location Counter (the next instruction to execute), then you can issue “ZAP PSW <addr-expr>” or use the “J” shortcut command to change the next instruction (“jumping” the PSW’s ILC). All the other changes to the PSW are accomplished with the Set PSw command.

To interrogate the current state of the PSW or PSWE, issue “List PSW(E) Format”.

More Info: Enter Help COmmands SEt PSw Addr-mode. For an overview, see also Help Commands SEt PSw.



Set the Address Space Control-mode bit of the Program Status Word (PSW) or the PSWE (the 64-bit PSW).

PRimary Set the ASC-mode bit in the PSW to Primary (normal) mode.
SEcondary Set the ASC-mode bit in the PSW to Secondary mode.
Home Set the ASC-mode bit in the PSW to Home mode.
Ar Set the ASC-mode bit in the PSW to Access Register mode.
ASCMODE= If you like typing, you may preface each parameter for this command with "ASCMODE=".

Remarks: **In Primary mode**, both instructions and data are fetched from the Primary Address Space.
In Secondary mode instructions are fetched from the Primary Address Space, and data is fetched from a Secondary Address Space.
In Home mode, both instructions and data are fetched from the Home Address Space. z/XDC must be running APF-authorized before this mode can be selected.
In Access Register mode, instructions are fetched from the Primary Address Space, while data is fetched either from the Primary Address Space or permitted data spaces and address spaces that are linked to the user program via access registers

Set PSW and its functions are very powerful, and worth handling with caution.

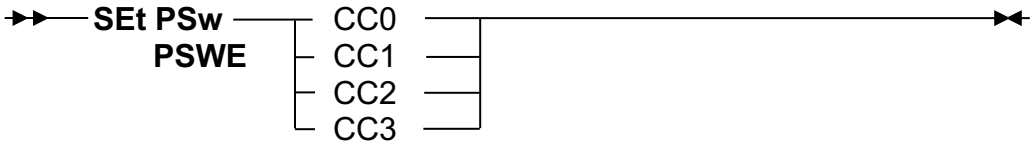
Note: If all you want to do is change the Instruction Location Counter (the next instruction to execute), then you can issue "ZAP PSW <addr-expr>" or use the "J" shortcut command to change the next instruction ("jumping" the PSW's ILC). All the other changes to the PSW are accomplished with the Set PSw command.

To interrogate the current state of the PSW or PSWE, issue "List PSW(E) Format".

Here is a chart that may help you understand what is possible:

	Primary	**AR**	**Secondary**	**Home**
Problem state	Y	Y	Y	N
Supervisor State	Y	Y	Y	Y

More Info: Enter Help COmmands SEt PSw Asc-mode. For an overview, see also Help Commands SEt PSw.



Set the condition code-bit in the Program Status Word (PSW) or the PSWE (the 64-bit PSW).

CC0 The condition code is set to binary zeroes. Aliases are CC=0, CCEQUAL, or CCZERO.

CC1 The condition code is set to a value of binary 1 (B'01'). Aliases are CC=1, CCLOW, CCMINUS or CCMIXED.

CC2 The condition code is set to a value of binary 2 (B'10'). Aliases are CC=2, CCHIGH or CCPLUS.

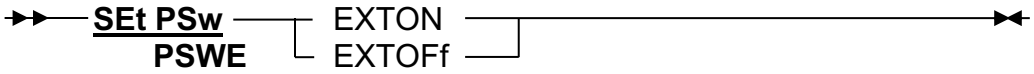
CC3 The condition code is set to a value of binary 3 (B'11'). Aliases are CC=3, CCONES or CCOVER.

Remarks: z/XDC allows the user to directly control the state of the Program Status Word. Set PSW and its functions are very powerful, and worth handling with caution.

Note: If all you want to do is change the Instruction Location Counter (the next instruction to execute), then you can issue "ZAP PSW <addr-expr>" or use the "J" shortcut command to change the next instruction ("jumping" the PSW's ILC). All the other changes to the PSW are accomplished with the Set PSw command.

To interrogate the current state of the PSW or PSWE, issue "List PSW(E) Format".

More Info: Enter Help COmmands SEt PSw Cc. For an overview, see also Help Commands SEt PSw.



Enable or disable the External Interrupt Mask in the Program Status Word (PSW) or the PSWE (the 64-bit PSW).

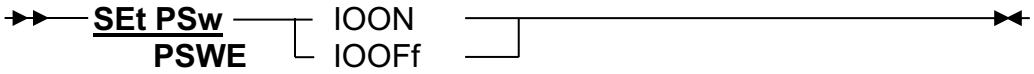
EXTON External interrupts are enabled. Aliases of this parameter are EXTENABLE, EXT=ON and EXT=ENABLE.
EXTOFF External interrupts are disabled. Aliases of this parameter are EXTDISABLE, EXT=OFF, and EXT=DISABLE.

Remarks: z/XDC allows the user to directly control the state of the Program Status Word. Set PSW and its functions are very powerful, and worth handling with caution.

Note: If all you want to do is change the Instruction Location Counter (the next instruction to execute), then you can issue “ZAP PSW <addr-expr>” or use the “J” shortcut command to change the next instruction (“jumping” the PSW’s ILC). All the other changes to the PSW are accomplished with the Set PSw command.

To interrogate the current state of the PSW or PSWE, issue “List PSW(E) Format”.

More Info: Enter Help COmmands SEt PSw Ext. For an overview, see also Help Commands SEt PSw.



Enable or disable the I/O Interrupt Mask in the Program Status Word (PSW) or the PSWE (the 64-bit PSW).

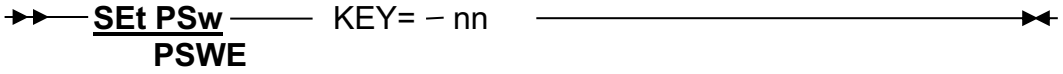
IOON I/O interrupts are enabled. Aliases of this parameter are IOENABLE, IO=ON and IO=ENABLE..
EXTOFF I/O interrupts are disabled. Aliases of this parameter are IODISABLE, IO=OFF and IO=DISABLE..

Remarks: z/XDC allows the user to directly control the state of the Program Status Word. Set PSW and its functions are very powerful, and worth handling with caution.

Note: If all you want to do is change the Instruction Location Counter (the next instruction to execute), then you can issue "ZAP PSW <addr-expr>" or use the "J" shortcut command to change the next instruction ("jumping" the PSW's ILC). All the other changes to the PSW are accomplished with the Set PSw command.

To interrogate the current state of the PSW or PSWE, issue "List PSW(E) Format".

More Info: Enter Help COmmands SEt PSw Io. For an overview, see also Help Commands SEt PSw.



Control the storage access key of the Program Status Word or the PSWE (the "error level" PSW).

KEY Indicates to z/XDC that you wish to alter the PSW's KEY value. If you like typing you may add an equal character and state "Key=nn".

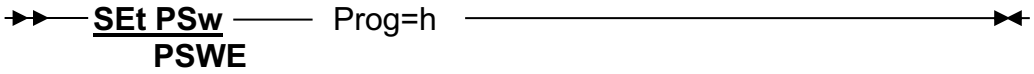
nn A decimal numeric value from 0 to 15, or a hexadecimal value from 0 to F.

Remarks: If z/XDC is running non-authorized, then this command can be used to set only those key values that are permitted by the retry level's PSW key mask. If z/XDC is running APF-authorized then you may set the key to any value in the range of 0 to 15.

Note: If all you want to do is change the Instruction Location Counter (the next instruction to execute), then you can issue "ZAP PSW <addr-expr>" or use the "J" shortcut command to change the next instruction ("jumping" the PSW's ILC). All the other changes to the PSW are accomplished with the Set PSw command.

To interrogate the current state of the PSW or PSWE, issue "List PSW(E) Format".

More Info: Enter Help COmmands SEt PSw Key. For an overview, see also Help Commands SEt PSw.



Enable or disable the four Program Mask bits in the Program Status Word (PSW) or the PSWE (the 64-bit PSW).

h A hexadecimal digit . z/XDC sets the four Program Mask bits in the retry level PSW to the 4-bit value represented by the given hex digit.

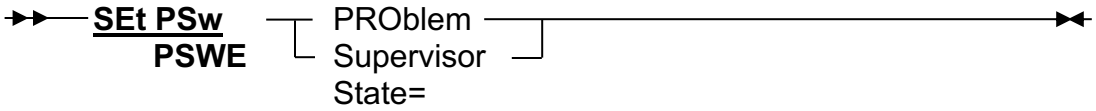
Remarks: z/XDC allows the user to directly control the state of the Program Status Word. Set PSW and its functions are very powerful, and worth handling with caution.

An alias for “Prog=” is “PGm=”.

Note: If all you want to do is change the Instruction Location Counter (the next instruction to execute), then you can issue “ZAP PSW <addr-expr>” or use the “J” shortcut command to change the next instruction (“jumping” the PSW’s ILC). All the other changes to the PSW are accomplished with the Set PSw command.

To interrogate the current state of the PSW or PSWE, issue “List PSW(E) Format”.

More Info: Enter Help COmmands SEt PSw Prog. For an overview, see also Help Commands SEt PSw.



Control whether the Program Status Word is set to Problem or Supervisor state.

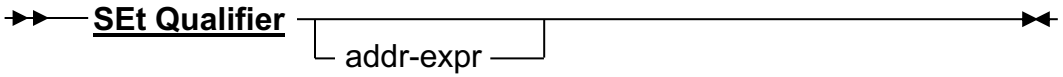
PRoblem The Problem state-bit in the PSW will be set on. When the user's program resumes, it will do so in Problem State.
SUpervisor The Problem state-bit in the PSW will be set off. When the user's program resumes it will do so in Supervisor state.

Remarks: z/XDC can set the state-bit to Supervisor state ONLY when z/XDC is running APF-authorized. If you want to, you can specify the keyword "STATE=", but it's not required so I didn't diagram it. Note: Set PSW and its functions (especially this one) are very powerful. Please use with caution.

Note: If all you want to do is change the Instruction Location Counter (the next instruction to execute), then you can issue "ZAP PSW <addr-expr>" or use the "J" shortcut command to change the next instruction ("jumping" the PSW's ILC). All the other changes to the PSW are accomplished with the Set PSw command.

To interrogate the current state of the PSW or PSWE, issue "List PSW(E) Format".

More Info: Enter Help COmmands SEt PSw Exec-State. For an overview, see also Help Commands SEt PSw.



Establish, reset or clear a default module name and csect name.
Shortcut command equivalent: “Q”

addr-expr Any address expression. z/XDC resolves the virtual address of the address expression and uses the current module and csect name in other z/XDC commands, such as Format. If addr-expr is omitted, then any previously defined qualifiers are cleared.

Remarks: An address space is a BIG place, and it is useful to be able to set a convenient frame of reference. For example, if one wishes to directly reference a label within a MAPped csect by name, one must normally issue the entire address expression that points to it in virtual memory. That complete address expression would be modulename.csectname.label, which is a LOT to type. The Set Qualifier command establishes a “default” module and csect name. Thereafter, one may refer to any label within that range of addresses as .label-name (the dot is significant).

You may also establish your default module and csect by using the “Q” shortcut command. Format anywhere you like within the desired csect, tab to column one anywhere in the working window, enter a “Q” and press Enter.

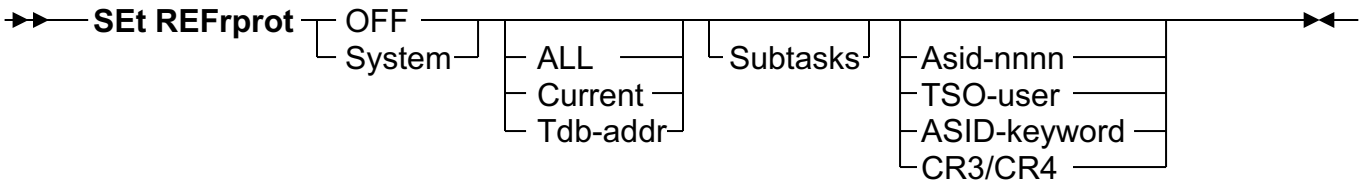
LE C and C++ programmers: If you issue LIST LKEDMAP <address-expression> then select an entry point and enter a “Q” there, you may refer to statement numbers in your C program by “.#nnn” where “nnn” is a given statement number. Now you can set breakpoints at statement numbers!

However, the pound sign (“#”) may only be used in your MAIN() function. All other csects must be prefaced with an alphabetic character (“A”, “B”, “C” etc.) as those csects appear in the load module.

Metal C programmers: The same applies but the pound sign (“#”) doesn’t exist. In this case your MAIN() function is “A” through “Z”. So in your case you’d issue SET QUALIFIER (or use the “Q” shortcut command) then manipulate statement numbers as “.Annn” or “Bnnn”, etc.

Examples: S Q PSW! ← The module name and csect name of the location pointed to by the instruction address portion of the PSW is established as the default.

More Info: Enter Help COmmands SEt Qualifier.



If REFRPROT is enabled for a system, directly control the way in which refreshable modules are loaded.

OFF Turn REFRPROT off for any task in any accessible address space.

System Set the task-level REFRPROT setting(s) for the targeted task to match the system-wide setting.

ALL Set REFRprot will target all tasks in the targeted address space (not including system tasks, such as Region Control, Started Task Control and the System Dump Task).

CURRENT Target only the home address space's current task.

Tcb-addr Target a specific task. TCB-addr must point to the starting address of a Task Control Block. The List TAsks command would be handy for this, allowing you to specify a TCB#n equate as a parameter for Set REFRprot.

Subtasks Used with CURRENT or TCB-addr parameters, this parameter will target all sub-tasks for the specified "parent" task.

ASID-nnnn Identify the targeted address space with the 4-digit hexadecimal number that corresponds to the address space (from the Address Space Control Block and/or the Address Space Vector Control Table). Once identified, alter the REFRPROT setting for that address space.

TSO-user Identify the targeted address space by using the name of a TSO user's address space, and alter the TSO-user's address space's REFRPROT settings.

ASID-keyword Identify the targeted address space by using a keyword such as "HOME", "PASID", "SASID", etc. Once identified, alter the ASID's REFRPROT settings. See Help COmmands SYntax ASids for more information.

CR3/CR4 Identify the targeted address space by using a Control Register. You may also specify "ECR3" and "ECR4" (the error-level Control Registers). Once identified, alter the target address space's REFRPROT settings.

Remarks: Beginning with z/OS V1R9, refreshable modules will ALWAYS be loaded into Key-0 storage regardless of whether they are loaded from an APF-authorized library and regardless of whether or not the RENT attribute is also on. This command allows you to alter the REFRPROT settings within the system. This would allow you to debug Refreshable programs in Key 8 storage and you would NOT have to be running z/XDC APF-authorized to do so (which is normally required). So, it's a convenience for the user. There is also a setting in z/XDC's Debugging Startup Panel where you can affect the REFRPROT setting.

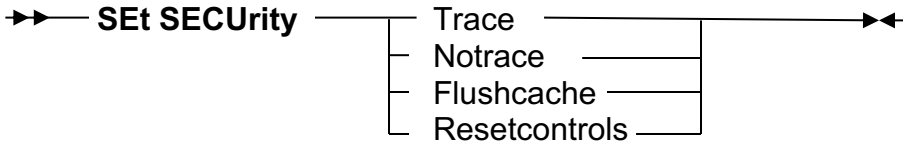
These settings can be reviewed with z/XDC's List REFrprot command.

Rules for this command:

i. One or the other of the OFF and SYSTEM operations is required. All other operands are optional. Varying defaults will be taken for omissions.
ii. Parameters may be given in any order.
iii. Parameters appearing above within the same column are mutually exclusive.
iv. The SUBTASK and ALL operands are mutually exclusive.

Examples: S REFR OF HOME ← Turns REFRPROT off for the home address space's current task.
Set REFRPROT SYSTEM ALL JES2 ← Reset the REFRPROT settings to the system-wide setting for JES2's jobstep task and all of its descendent tasks.

More Info: Enter Help COmmands SEt REFRprot. See also Help COmmands LISt REFRprot. See also Help COmmands SYntax ASids.



Manipulate z/XDC's security interface.

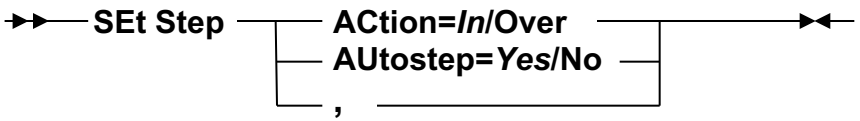
Trace Enable a security calls trace. Security traces will be displayed at the user's terminal along with z/XDC's other output.
Notrace Disable a security calls trace.
Flushcache Immediately expire z/XDC's internal security cache. The maximum lifetime of any cache entry is no more than two minutes. See Help SECurity Cache for more information.
Resetcontrols Normally, z/XDC displays some security-related events only once per debugging session. Resetcontrols allows these messages to be displayed again.

Remarks: remarks
A security trace may also be run by adding a ///XDCTRSAF DD DUMMY JCL statement.

This command may/may not be used in foreign address space mode.

Examples: Se SECU T ← initialize security trace messages at the console.
Se SECU F ← clear the security cache.

More Info: Enter Help COmmands SEt SECUrity See also Help SEcurity. See also Help Security CAche.



Control c/XDC's interaction with prolog/epilog C code and subroutines or functions.

Action= The default value, "In" specifies that c/XDC will normally process C statements one at a time, and will follow execution into a subroutine call or function, stopping at the first statement in the subroutine or function. If "Over" is specified, then c/XDC will allow subroutines/functions to execute entirely and will capture control upon return. This setting is overridden with the interactive command Step. It's best to leave AC=In as the default.

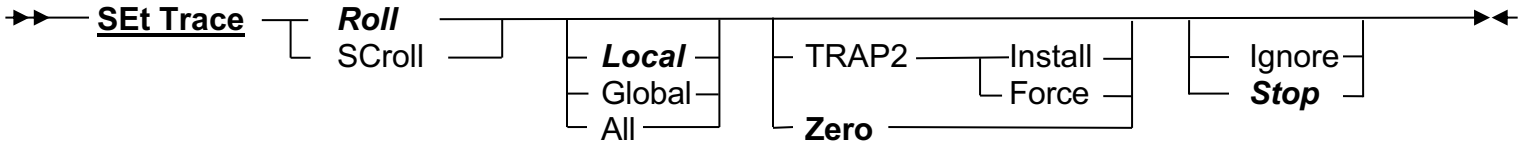
AUtostep= The default value, "Yes" specifies that c/XDC will automatically skip stepping through C Prolog and epilog code. This is usually the desired result. If you specify "No" instead, then c/XDC will dutifully step through this code. Leave this setting alone as well. Alias of "Yes" is "On". Alias of "No" is "Off".

Note: You can influence the settings for STEP and Autostep both from your profile (Profile/Display/Change High Level Language (HLL) Settings) AND you can change it on the z/XDC Startup Panel. Thanks, Mike and Dave for this enhancement.

Remarks: The comma means that you can twiddle both parameters in the same command string, separating the parameters by a comma. That said, I don't see any need for you to change these settings under normal conditions. However, you can if you care to.

Examples: S ST AC=I ← This establishes c/XDC's default behavior. I just did you a favor.
SET STEP AU=No ← Lucky you. You get to step through C Prolog and Epilog code.

More Info: Enter Help COmmands SEt Step. See also Help COmmands LISt STEp and Help COmmands STep



Control the scope and action of the Trace command. Also control how z/XDC identifies the current instruction in the Working Window.

Roll Default setting. This causes the trace command to move "down" the screen, highlighting the next instruction as it goes.

SCroll This causes the "next" instruction to appear at the top of the working window, removing from view the previous instructions.

Local Default setting. Keeps the "scope" of a trace within the Private Area. In other words, z/XDC won't follow a trace to protected storage or common storage or from any common storage module to any other common storage module.

Global/All This causes the Trace command to follow "anywhere" it may go – such as a subroutine or supervisor call outside the home address space. This setting is to be used with caution and forethought (do you really want to set a breakpoint in an SVC?), and requires that the session be APF-authorized. "ALL" is an alias for "GLOBAL".

TRAP2 You may now issue SET TRACE TRAP2 and z/XDC will perform breakpoints with TRAP2 instructions rather than the classic X'00'. This buys some efficiencies. It also ELIMINATES the dichotomy between the "Error Level environment" vs the "Retry Level environment". You may also choose TRAP2 in the Profile Facility under Profile/ Display/Change FORMAT TRACE and FIND Settings. See HELP WHATSNEWS Z22 TRAP2 for important information.

Install If z/XDC's TRAP2 handler has not yet been installed, this will cause it to be installed immediately.

Force Will cause z/XDC's TRAP2 handler to be installed even if another TRAP handler is found. Note: There are considerations that I won't go into here. Please read Help COmmands SEt Trace Trap2 for more information.

Zero z/XDC uses X'00" to stop program execution by way of S0C1 abends, which z/XDC then captures. This is the "classic" way that z/XDC worked for many years. It is what fostered the requirement that Dave keep track of the dichotomy between the error-level and retry-level environments. I imagine that was a substantial effort on his part.

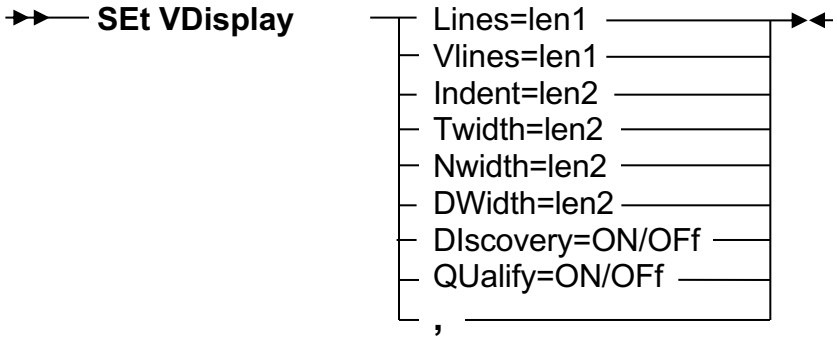
Ignore When using z/XDC's tracing commands, and when a #DIE trap is reached, IGNORE causes z/XDC not to stop execution at the #DIE trap.

STop Causes z/XDC to stop when a trace command reaches a #DIE trap.

Remarks: **If you want to see your current Trace settings, you can issue "List TRace".**

Examples: S TR SC ← The user wishes to keep the "next" instruction at the top of the screen.
S TR T ← z/XDC stops using X'00' for breakpoints and begins to use TRAP2 instructions.
S TR G ← If permitted by your security system, Set Trace Global is now in effect. z/XDC will now trace into Alien Memory. **BE CAREFUL. This will allow you to trace right into system routines, which you may NOT want… You have been warned.**

More Info: Enter Help COmmands SEt Trace. See also Help COmmands LISt TRace



Note: Len1 may range from 1 to 32767. Len2 may range from 1 to 255.

Control the output of the List VAriables command.

Note: Each of the parameters may be given all at once, with each parameter separated by a comma and space.

Lines= Limits the total number of output lines that the List VAriables command will produce. The default value is 32,767.

Vlines= Limits the maximum number of members within a structure that the List VAriables command will produce. The default value is 255.

Indent= Specifies the indentation value for each level of a structure's display. The default value is 2 columns to the right.

Twidth= Specifies the width of the "Type" column in the display before wrapping occurs. The default value is 25 columns.

NWidth= Specifies the width of the name column in the display before wrapping occurs. The default value is 25 columns.

DWidth= specifies the width of the data column in the display before wrapping occurs. The default value is 60 columns.

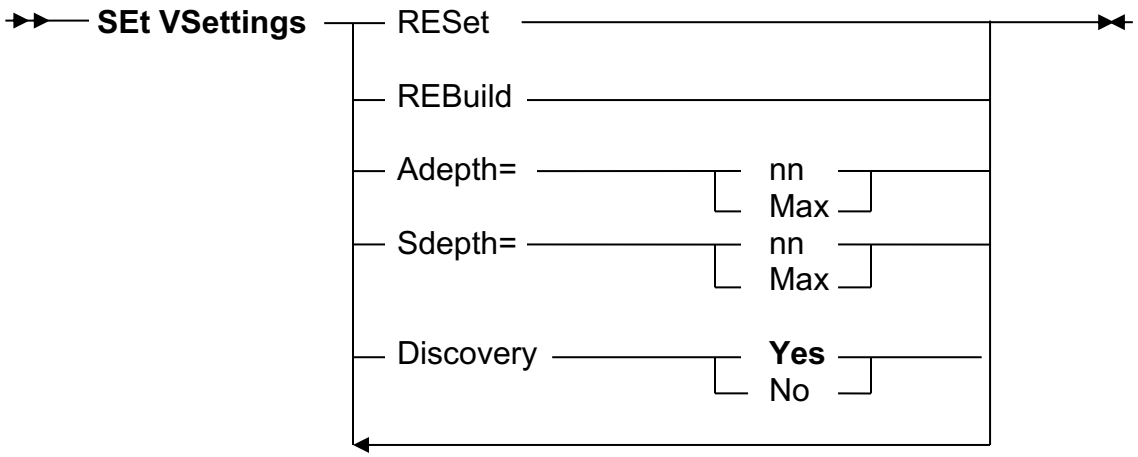
DIscovery= May be ON/Yes or OFf/No. This parameter controls whether (or not) c/XDC's List VAriables command will automatically expand structures. The default is "ON/Yes".

QUalify= May be ON/Yes or OFf/No. This parameter controls whether (or not) the fully qualified variable name is displayed. If "OFF" or "NO" then only the basic variable name is displayed. The default is "ON/Yes".

Remarks: Variable discovery is a huge part of c/XDC's architecture. During this work we came up with default values for Set VDisplay that we think are very good engineering choices. You may never need to alter these settings. Generally, it's not worth the brain damage to mess with this command. However, we give you the flexibility to do so.

Examples: S VD I=500 ← c/XDC will display a maximum of 500 lines in the output of the LISt VAriables command.
SE VD DI=OFF ← c/XDC will NOT automatically expand structures in the output of the LISt VAriables command.

More Info: Enter Help COmmands SEt VDisplay. See also Help COmmands LISt VAriables.



Cause c/XDC to reexamine or rebuild its understanding of variables and/or change limits on Array dimensions and Structure depth.

RESet Causes c/XDC to reexamine its knowledge of variable pools. If new variables are found, c/XDC will display "DBC303I New C variables discovered."

REBuild Discards and rebuilds all variable pools. This would be useful if you think that c/XDC has "lost its way".

Adepth= Sets the maximum number of array dimensions that c/XDC will display. The allowable range is 1-64. Adepth=Max sets the limit to 64 dimensions.

Sdepth= Sets a limit on the depth to which stack frames can be listed by the List Vstack command. SDepth=Max sets the limit to 256 structures.

Discovery When set to YES, this parm directs c/XDC to examine all retry-level environments looking for changes to the user program's defined C-language variables and the storage locations assigned to them. Variable discovery is a costly process and can affect c/XDC's performance. If set to "NO" then variables will not be discovered, and the user will have to subsequently issue the SET VSETTINGS RESET command. This can improve c/XDC's performance when working with large, complex applications.

Remarks: c/XDC's default values for these settings are at maximum, and with Discovery=YES. Therefore the only reason one might alter these values is in the presence of a very large programming environment wherein the max values would take up way too much memory. In that case you might also elect to reduce these values from the default (Adepth=64 and Sdepth=256).
ON THE OTHER HAND, and if you're debugging your batch program in batch mode (through our Cross Domain Facility) you could set REGION=0M and not worry about storage. Performance might still be affected, but….
In most cases it's not worth the brain damage to mess with this command. But, it's YOUR world.

More Info: Enter Help COmmands SEt VSettings

SEt Zap ***Normal***
Sprot

Control whether z/XDC will allow the user to alter store-protected storage, or not.

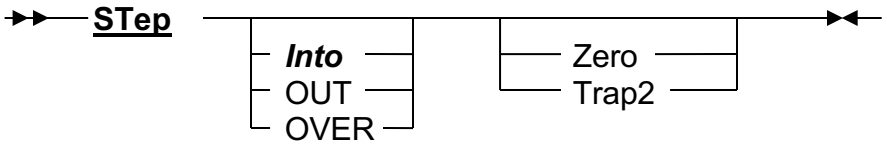
This command will work only when z/XDC is running APF-authorized.

Remarks: z/XDC normally will not allow the APF-authorized user to alter store-protected memory. However, if Set Zap Sprot is invoked, the user may alter store-protected memory. **While Set Zap Sprot is in effect, it is incumbent upon the z/XDC user to proceed with caution**, as *it is possible to crash a system in the time it takes to lift a finger from the Enter key.*

A prudent human will avoid self-inflicted wounds. There's your life-lesson for the day.

You can influence Set Zap in your Profile. Issue "Profile", then navigate to "Display/Change READ ZAP and Parse Order Settings".

More Info: Enter Help COmmands SEt Zap. Also, see Help Commands SEt NOIlc.



Execute one High-Level Language statement (such as C, C++ or Metal C).

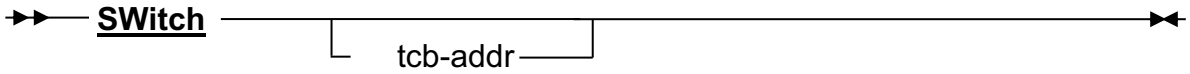
omitted Default action. Execute a single statement. If the next statement happens to be a subroutine, execution will begin there.
INto Step INTO a subroutine. If execution is NOT at a subroutine, c/XDC merely issues a STEP command, returning control at the next statement.
OUT Step OUT OF a subroutine.
OVER Allow a subroutine to execute, and return control after its execution
Zero Perform the STep command and force z/XDC to use a X'00' to do so.
Trap2 Perform the STep command and force z/XDC to use a TRAP2 instruction (X'01FF') to do so.

Remarks: The Step command allows you to trace a single High Level Language statement, which can encompass many machine instructions. It's one of the most useful features in c/XDC. Remember, c/XDC's interaction with High-Level Languages doesn't mean that z/XDC isn't available to you. You can still SET FORMAT BOTH to see raw object code AND source statements together and you can still issue TRACE, and breakpoint commands.

At this writing STEP INTO is a system default, and Dave Cole gives a pretty convincing argument why this is so in Help COmmands STep. He also says that if you want to change this behavior you can, but he neglects to tell you how. I did the research. You can change this with SEt STep.

Examples: ST ← Execute one High Level Language statement
ST OUT ← leave the current subroutine and return to the calling routine.

More Info: Enter Help COmmands STep.



When in a multi-tasking debugging environment, switch to debug a different sub-task. Shortcut command equivalent: "S"

Tcb-addr An address-expression that resolves to a Task Control Block in the address space, such as the equae "TCB#n". See below.

Remarks: The Switch command can be used in multi-tasking debugging situations when multiple tasks have either abended or reached breakpoints and, therefore, are all awaiting debugging services.

z/XDC Permits the user to hold a debugging conversation with only one task at a time. When multiple tasks request debugging services, z/XDC forces the additional tasks to wait. The Swtich command causes z/XDC to pass control of the debugging conversation from the current task to another task.

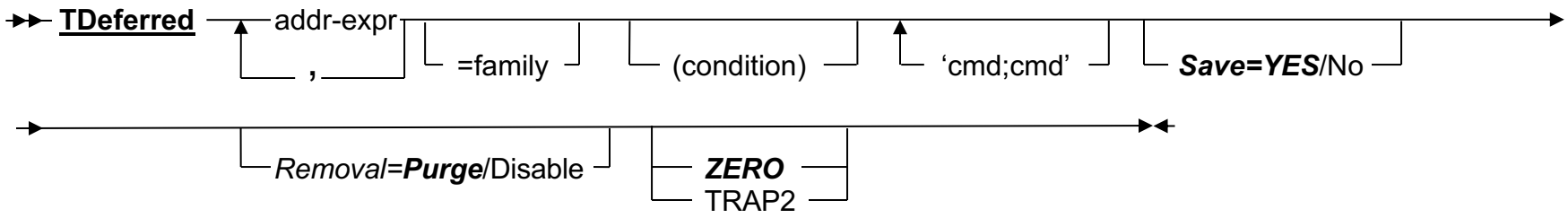
The "S" Shortcut command is the simplest way to use Switch. First, issue the LIST TASKS command, then select the desired sub-task, put an "S" next to it and press Enter. Nice!

Alternatively, after issuing the LIST TASKS command, one can enter SW TCB#n, using the automatic equates generated by z/XDC as a by-product of the LIST TASKS command.

If tcb-addr is omitted, z/XDC will switch to the "next" task for which it is the ESTAE. The order of what is the "next" task is random, BUT follows an order through the sub-tasks in the address space in a "repeatable circuit".

Examples: SWITCH TCB#2

More Info: Enter Help COmmands SWItch.



Create a transient deferred breakpoint in the desired module.

addr-expr Address expression(s) that names a module, a module.csect and (if desired) the offset where the deferred breakpoint is to be placed. The "base" of the addr-expr MUST be the primary name of a load module. Separate multiple address expressions with commas OR spaces.

=family A breakpoint family may be specified (a series of breakpoints named alike). The breakpoint will then be assigned to that family of breakpoints.

(condition) A conditional expression may be specified to further control under which situations the breakpoint is to be taken.

'cmd;cmd' An additional z/XDC command (or commands) that are to be executed when the breakpoint is taken. New for z/XDC Release z2.1, you no longer may use colons to associate commands with traces or breakpoints. Instead you enclose the command string within single quotations and separate commands within a string with a semi-colon. See Help Whatsnew Z21 Syntaxchange! for more information.

Save= When a conditional expression or an associated command string is stated, z/XDC normally saves them for us as defaults in subsequent break-pointing commands. Omitting this parameter defaults the action to Save=Yes. Save=No turns this off.

Removal= Default action is "Purge". This would certainly apply to transient breakpoints (which are automatically deleted when executed once). "Disable", on the other hand would deactivate the breakpoint (changing the op-from X'00' or X'01FF' back to its original value). This retains the breakpoint, and in the LIST BREAK command's output you will see that the breakpoint is "DISABLED". You can enable or disable a breakpoint using the "X" shortcut command next to any breakpoint. PURGE will cause the breakpoint to be automatically deleted when it is executed. In this case it would be just as easy to create a TRANSIENT breakpoint.

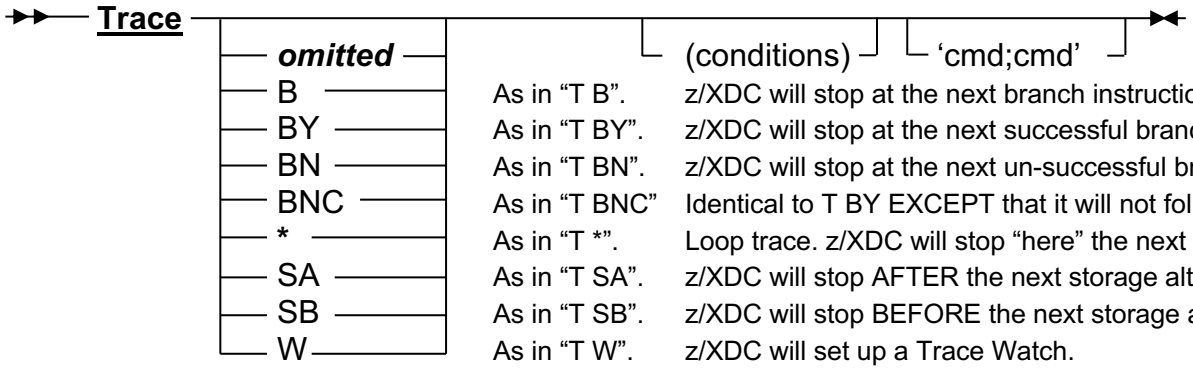
ZERO This will create a "classic" breakpoint of X'00'. It will cause a S0C1 abend as z/XDC has done for decades.

TRAP2 This invokes z/XDC's TRAP2 support, in which case z/XDC will create a X'01FF' at the breakpoint and will use the TRAP2 facility to create breakpoints.
Note: for more information on z/XDC's TRAP2 support, enter "Help Whatsnew z22 Trap2".

Remarks: This command creates a breakpoint in a module that will be loaded into memory sometime in the future. This is a "scheduled" breakpoint. A breakpoint created by ADeferred remains in existence for the life of the debugging session or until it is explicitly removed. If you desire a one-time-only deferred break-point, see TDEFERRED.
This command may not be used in Foreign Address Space Mode.

Examples: AD MYPGM+3DC ← Create a Deferred breakpoint at offset "+3DC" into module MYPGM whenever it loads.
AD MYPGM.MYCSECT+3DC ← Create a Deferred breakpoint at offset "+3DC" into module MYPGM and csect MYCSECT whenever it loads. You would have had to issue a DMAP command for MYPGM prior to this so that z/XDC can "find"csect MYCSECT.
AD =5 MYPROG+3DC +4 +8 (R1,EQ,00) :L REGS:L AREGS ← OK, this is a toughie. Set a Deferred breakpoint into Family "5" at MYPROG+3E8 but only if Register one is equal to all zeroes. If and when this Transient Deferred breakpoint fires, then issue "LIST REGS and LIST AREGS.

More Info: Enter Help COmmands ADeferred. For conditions see HELP Breakpoints Conditional and Help Commands Syntax Breakpoints Conditions.



Follow the execution path of a program.

Omitted z/XDC will return control after a single instruction.

(conditions) A counting condition or a value-testing condition.

'cmd;cmd' Another or many z/XDC commands to be executed when the trace takes effect. New for z/XDC Release z2.1, you no longer may use colons to associate commands with traces or breakpoints. Instead you enclose the command string within single quotations and separate commands within a string with a semi-colon. See Help Whatsnew Z21 Syntaxchange! for more information.

Remarks: Trace is one of the core commands with in z/XDC, so much so that the "factory default" Profile has PF9 set to "T", and PF10 set to "T BY".

Neat feature: Whenever z/XDC "stops" at a Branch-type instruction it automatically evaluates the branch. If the branch will be taken, then the branch target is displayed next to the instruction within parentheses. If the branch will fall through, then z/XDC displays "(NO BRANCH)". Nice.

More information on the parameters of the Trace command may be found above in this booklet in the section entitled, "z/XDC's Trace Commands".

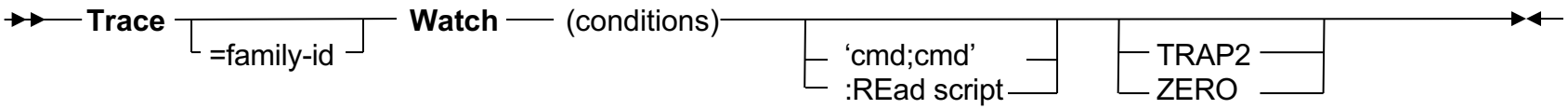
A Trace Watch is a special case that sets up a conditional statement that is not tied to any breakpoint or trace, but is evaluated WHENEVER z/XDC gains control. Trace Watch is the very next command discussed in this book.

Dave wrote T BNC because sometimes in his AUTOTRACE script, z/XDC would go down into the system and never resurface (down the "rabbit hole"). This often made the AUTOTRACE Script useless. Now he's incorporated the T BNC command into a new script called AUTOTBNC. See Help Scripts Ourscripts AUTOTbnc for more information.

This command may not be used in Foreign Address Space Mode.

Examples: T ← Execute to the next sequential instruction
T BY 'LIST PSW F:LIST REGS:RETRIEVE' < ← Trace to next successful branch. When there, also show the PSW (broken down by fields) as well as the general registers. Finally a RETREIVE command is processed which puts this command string back on the command line. Then all you have to do is keep pressing Enter to keep running the command. I LOVE THIS.

More Info: Enter Help COmmands TRACe, and Help Commands TRAce Watch. See also Help COmmands SYntax Breakpoints.



Monitor, and report on a conditional statement in the background.

=family-id A specified family-id. Like Breakpoints and Traces, a Trace Watch may be part of a family.

(conditions) A value-testing conditional statement. See Help COmmands Syntax Breakpoints CONditions Value, or look in the earlier part of this book.

'cmd;cmd' After a Trace has been performed, you may "associate" other z/XDC commands that will be executed after the trace completes. New for z/XDC Release z2.1, you no longer may use colons to associate commands with traces or breakpoints. Instead you enclose the command string within single quotations and separate commands within a string with a semi-colon. See Help Whatsnew Z20 Syntaxchange! for more information. A very handy command to associated would be "'ALARM'", which would ring the terminal display's bell.

:**Read script** There is no reason that you couldn't associate a z/XDC script with a Trace Watch command. Then MANY cool things could happen when your Trace Watch's value-testing conditional statement proves true. See Help COmmands REAd, and Help Scripts for more information.

Trap2 The Trace Watch will use a TRAP2-type breakpoint regardless of the current SET TRACE TRAP2 | ZERO command.

Zero The Trace Watch will use a X'00'-type breakpoint regardless of the current SET TRACE TRAP2 | ZERO command.

Remarks: A Trace Watch is a conditional expression of the value-testing variety that is not tied to any particular trap or trace. Instead of being checked as a result of a trap or a trace, a Trace Watch is evaluated EVERY TIME that z/XDC gets control (which could be hundreds of times per second). So, a Trace Watch can be considered a "fire-and-forget" weapon of debugging. Trace Watches can also be used to look for compound conditions.

Trace Watches DO NOT STOP EXECUTION when they trigger, but they DO signal the truth of the conditional expression the next time that z/XDC gets control after the condition proves true. After implementing Trace Watch, enough customers complained about Trace Watch's inability to stop program execution that Dave has implemented two new Boolean parameters for conditional expressions: "TRUE" or "FALSE".

If you set a *separate* trace or trap command with (FALSE) as the conditional expression AND then you define a Trace Watch, THEN the WATCH will be evaluated at this/these trace(s) or breakpoint(s) BUT control of the program will NOT be interrupted unless the WATCH tests TRUE.

ALL WATCHs are evaluated at each breakpoint or trace command. Any number of WATCHs may be defined.

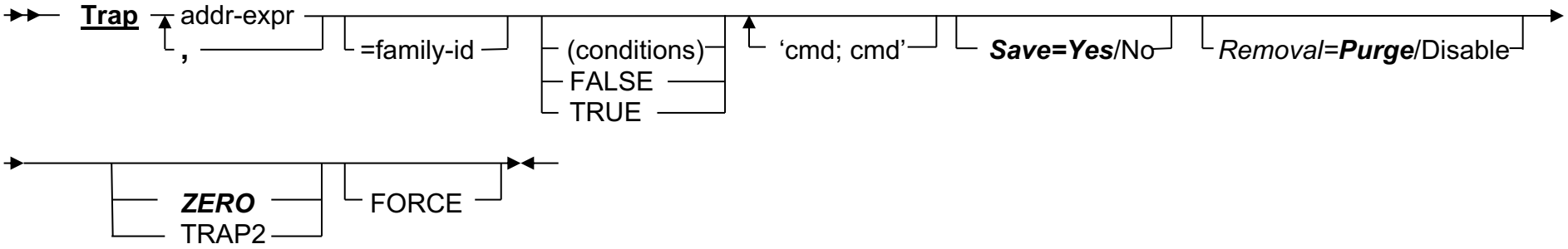
A WATCH may also have one or more z/XDC commands "associated" with it, and whenever the conditional express in the WATCH tests TRUE, then those commands will be executed.

It is possible for multiple WATCHES to test TRUE at the same time.

This command may not be used in Foreign Address Space Mode. But if you were to HOOK the address space and go back in through Cross Domain Facility, then you COULD issue this command.

Examples: T W (R1?,NE. ' FRED') 'ALARM' ← Monitor program execution. When Register 1 no longer points to an Equate called "FRED" then trigger the associated command, which in this case is ALARM.
T W (.PTR,NE,00CDFE2A)'ALARM ' ← A label in the program (.PTR) contains the initial value X'00CDFE2A', which for some reason is critical. Tell the z/XDC user when this label gets changed.

More Info: Enter Help COmmands TRACe W. See also Help Breakpoints Conditional Watch. See also See also Help COMmands Syntax Breakpoints CONditions .



Set a transient breakpoint at a specified location in virtual memory in order to stop program execution at that location. Shortcut command equivalent: "T"

addr-expr Specify the location where you wish to place the breakpoint. You may specify multiple address expressions separating them with a comma OR a space. Multiple breakpoints created at the same time will be assigned to the same family (a numeric value).

=family-id Assign this breakpoint to a particular family.

(conditions) Specify a Counting or Value-Testing conditional statement (See Conditional Expressions in this booklet or type Help COmmands Syntax Breakpoints CONditions).
FALSE will bypass the trap, or cause the Trace to advance one instruction. **TRUE** will cause the trap to be accepted or the Trace operation will stop. FALSE/TRUE were introduced in order to cause a TRACE WATCH to stop program execution – which it couldn't do originally.

'cmd' Specify other z/XDC commands that you would like to have executed WHEN the breakpoint is reached. Enclose the command string within single quotations. Separate individual commands within a string using a semi-colon.

Save= When a conditional expression or an associated command string is stated, z/XDC normally saves them for us as defaults in subsequent breakpointing commands. Omitting this parameter defaults the action to Save=Yes. Save=No turns this off.

Removal= Default action is "Purge". This would certainly apply to transient breakpoints (which are automatically deleted when executed once). "Disable", on the other hand would deactivate the breakpoint (changing the op-from X'00' or X'01FF' back to its original value). This retains the breakpoint, and in the LIST BREAK command's output you will see that the breakpoint is "DISABLED". You can enable or disable a breakpoint using the "X" shortcut command next to any breakpoint. PURGE will cause the breakpoint to be automatically deleted when it is executed. In this case it would be just as easy to create a TRANSIENT breakpoint.

ZERO This will create a "classic" breakpoint of X'00'. It will cause a S0C1 abend as z/XDC has done for decades.

TRAP2 This invokes z/XDC's TRAP2 support, in which case z/XDC will create a X'01FF' at the breakpoint and will use the TRAP2 facility to create breakpoints.
Note: for more information on z/XDC's TRAP2 support, enter "Help Whatsnew z22 Trap2".

FORCE Allowed only for the Trace, and Trace Watch commands. Will allow the Trace command to be used even when a trap save area corruption has occurred. If no trap save area corruption has been detected, this operand is ignored.

Remarks: This is one of z/XDC's core commands; one that you will use in almost every session. TRAP is an alias of this command. You can also specify "ATX" to create a "hard-to-remove" breakpoint (that must be OFF'd explicitly). This command may not be used in Foreign Address Space Mode.

Examples: **T R14? +4 + 6 .JOBTOP1 +2 'LREGS;F R15?'** ←Set six transient breakpoints, then execute a List Registers and a Format at the location pointed to by the contents of R15 whenever the breakpoint(s) is/are taken.

T myprogram.mycsect+23C ← Set a single breakpoint at offset 23C into the csect "mycsect" of the module "myprogram".

T .loopstart (R1,NE,7FFFFFFF) 'ALARM WHOA! IT HAPPENED!' ← Set a breakpoint at a label in the program that will only stop execution if the contents of Register One are no longer equal to X'7FFFFFFF' Then, automatically create an alarm condition to alert the programmer.

T R5+10 (R3+3,AND,02,EQ,02) Set a breakpoint that will not be accepted until the 31st bit (i.e., bit number 30) of R3 is on.

More Info: Enter Help COmmands AT. For conditions see HELP Breakpoints Conditional and Help Commands Syntax Breakpoints Conditions.

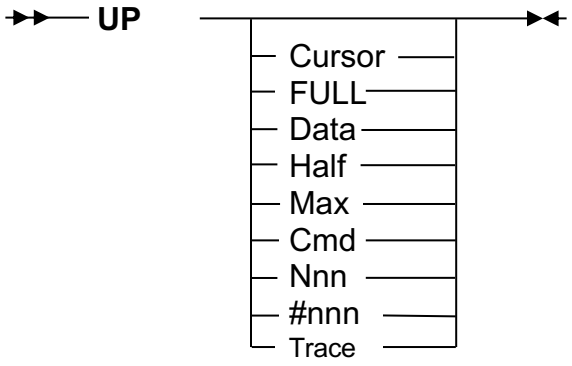
Issue a TSO command from within a z/XDC debugging session.

Command parameter(s) The desired TSO command
the parameters for the TSO command, if any.

Remarks: This command will cannot be used when using z/XDC to debug a batch job. When the TSO command is entered, z/XDC will suspend its operation until the specified TSO command completes. Then the current debugging session is resumed.

Examples: TS LISTD 'SYS1.LPALIB' ← A LISTD command is issued. Debugging of the current debugging session resumes when the LISTD command completes.
TSO ISPF ← The current debugging session is suspended and an ISPF session is started. When the ISPF session ends, the current debugging session is resumed.

More Info: Enter Help COmmands TSo.



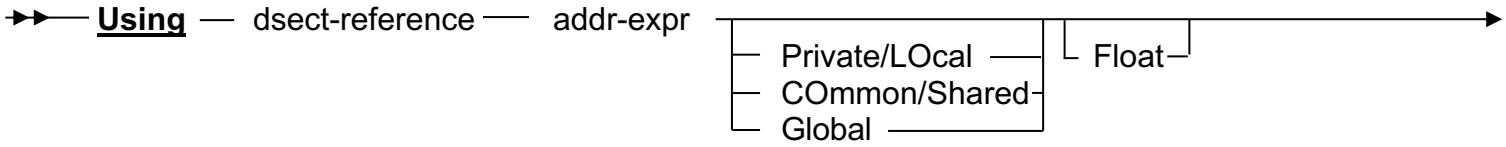
Move the display window up a specified distance through the session log.

omitted	Scroll up by the window's default scroll amount (as controlled by the SEt WIndow command).
Cursor	The window is scrolled to the line on the display that contains the cursor. This parameter may only be entered via a PFkey setting.
Full	Scroll the distance of a full window. "Page" is a synonym for this parameter.
Data	Scroll so that the previous screen's top becomes the bottom of the screen.
Half	Scroll one-half of the vertical distance of the screen.
Max	Scroll to the beginning of the session log.
Cmd	Scroll up to the next command.
nnn	Scroll up a specified number of lines.
#nnn	Scroll up to a specifically numbered command in the log.
Trace	May only be used in the Working Window. If you have previously issued DOWN to navigate to an later time (down) in the Session Log, then UP Trace will display the PREVIOUS time z/XDC got control and painted a screen. Dave Cole points out a useful idea: From wherever you are now in the session log issue UP TRACE;RETRIEVE. This will allow you to see your debugging session run backward like a movie! That might be worth trying out some time.

Remarks: Every parameter in this command will accept simple arithmetic that alters the "value" of the UP command's output. That said, PageUp/PageDown work pretty well if you're too lazy to enter the UP (or DOWN) commands.

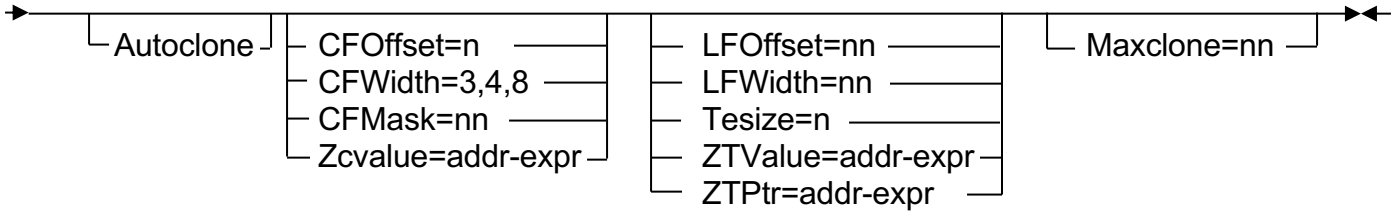
Example: DOwn Half 10+5 ← Same as entering "DOwn Half 15".
DO CMD-5 ← goes down one command in the session log (presuming that you're located "above" the current command) minus five lines.
DO 12-3+4 ← Same as "Down 11".

More Info: Enter Help COmmands DOwn.

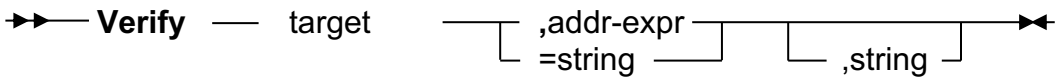


Place a dsect in memory at a fixed address or set it to "float" with the resolution of addr-expr.
Helper Dialog: "USING ?"

dsect-reference	A field in the dsect that will be used as the origin address. Sometimes the origin address is NOT the first field in the dsect.
addr-expr	The address in memory where the user wishes to place the dsect.
PRIVATE/LOCAL	"PRIVATE" and "LOCAL" are aliases of each other. When used, they force the dsect to be treated as residing in the Private Area, even if it is actually assigned to a Common Area location.
COMMON/SHARED	A common or shared dsect is owned by ALL spaces, address spaces, data spaces and real storage. You can override z/XDC's determination of a dsect's placement by using this parameter. Common/shared dsects are not displayed when real storage or data space storage is displayed. Use "GLOBAL" for that purpose.
Global	A global dsect is owned by ALL spaces: address spaces, data spaces and real storage.
omitted	GLOBAL, COMMON, SHARED, PRIVATE and LOCAL all omitted. If the dsect resolves to a common storage location then it is treated as being Common. Otherwise it is treated as being private. This may be your "best" choice in most cases.
Float	The Float parameter specifies that the origin address of the dsect will move in memory according to the resolution of addr-expr. This is very handy for correctly placing dsects that might map an input or output record, or any other data structure that may "move" in memory.



Autoclone	Directs z/XDC to create a list of named equates that refer to pointers in a chain, or elements in a queue according to Autoclone's own parameter list. If all other parameters are omitted, then z/XDC assumes these defaults: CFOFFSET=0, CFWIDTH=4, CFMASK=7FFFFFFF, ZCVALUE=0 and MAXCLONE=10.
CFOFSET=n	The offset of the entry's chain field, starting from the beginning of the dsect. Can be expressed as a hexadecimal number OR a decimal number if followed directly by an "N". That is, "10" means 16, and "10N" means 10. The default offset is 0.
CFWIDTH=n	Defines the width of the chain field. Permitted values are 3, 4, or 8. the default width depends on whether or not the CFMASK option is used. If CFMASK is not given, then the default width is 4.
CFMASK=n	Defines a selection mask that is AND'd against the chain field's value to eliminate bits that are not part of the chain pointer. The width of the mast must match the width of the chain field (as defined by the CFWIDTH parameter).
ZCVALUE=	The "zero chain field" value. z/XDC's default is to consider a zeroed chain field the end of the chain. However, you can also give an address expression that will be compared to the contents of each entry's chain field. A match identifies the chain's last entry. You can also use the keywords "NEGATIVE" or "NOT POSITIVE" as operands of the ZCVALUE parameter. Both of these keywords cause any zero or negative value to signify the end of the chain. When testing for a negative value, the CFMASK= parameter is ignored. z/XDC checks the field's hi-order bit. If it is ON, then the chain is ended.
LFOFFSET=n	If the table entries have a length field, then this operand provides the offset of that field from the start of each table entry. The number can be expressed as a hexadecimal number OR a decimal number if followed directly by an "N". That is, "10" means 16, and "10N" means 10. The default offset is 0.
LFWIDTH=	If the table entries have a length field, then this operand provides the width of that field. LFWIDTH may range in value from 1 to 8. There is no default value for this parameter. If LFOFFSET= is stated, LFWIDTH must also be stated.



Compare the contents of one memory location with the contents of another, or the contents of memory with a stated value.

target The location in memory that you wish to compare. This can be an address-expression that resolves within the Private Area, the Common Area, a Dataspace, Foreign Address spaces, real storage, registers, the PSW or the computed values returned by certain built-in functions.

,addr-expr The address-expression whose value is to be compared against the first address-expression. You must use either a comma, or a blank to delimit the second address-expression (but not both!).

=string A stated value that you wish to compare against the resolved address-expression. When multiple strings are given, the strings are converted into binary form and concatenated together before being used.

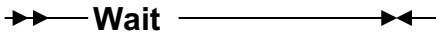
Remarks: When used, the Verify command will either work, or return message DBC106E VERIFY FAILURE. Structurally, VERIFY has the same features as ZAP.

z/XDC allows the user to create “scripts” that contain lists of z/XDC commands that can be executed en-masse upon invocation by z/XDC’s READ command. In order to allow such scripts to use the ZAP command, Dave Cole introduced the VERIFY command, so that the enterprising z/XDC user can write SUPERZAP functions in script format.

Folks, Verify can be used in far more interesting ways than I can document here (or understand!). Please read the Help Sub-system!

Examples: V R15+1=C’ABC’ ← Verifies that general register R5 contains X ’C1C2C3’ starting in its second byte.
VER PSW!=0A03 ← Verifies that the PSW points to a location that contains an SVC 3 instruction.

More Info: Enter Help COmmands Verify , Help COmmands Verify Targets, Help COmmands Verify Address and Help COmmands Verify String. Also, read Help COMmands Zap while you’re at it.



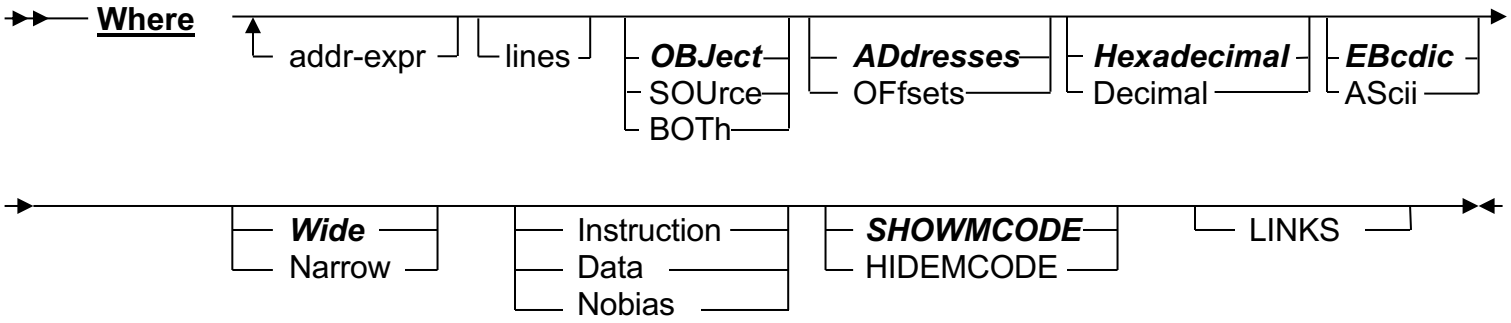
Resolve a conflict over the use of the terminal between z/XDC and a user or system task.

This command accepts no parameters.

Remarks: In the event of a terminal control conflict you may type the WAIT command repeatedly until it is received by the z/XDC task. z/XDC will then immediately cease trying to use the terminal and will wait until control is passed to z/XDC again.

I guess this is a “Shut up, z/XDC!” command.

More Info: Enter Help COmmands WAit. See also Help MUltitask Conversationmanagement/



Cause z/XDC to FORMAT the instruction pointed to by the Program Status Word. In other words, "Where am I?" Shortcut command equivalent, "W".

Note: **Just issue "W" from the command line or any acceptable space in column one of the display. You don't need all the optional parameters to make this command work.**

addr-expr The location in memory where the disassembly is to begin. If addr-expr is omitted, then the next screen-full of memory is disassembled. If addr-expr is omitted and other parameters are desired, then a comma must be used before stating the other parameters.

lines The number of lines the user wishes to see. The upper limit is 1000. If omitted, z/XDC will fill the current window.

SOurce *OBject* BOth If a source level map is available, source code can be shown as part of the formatted display. OBject is z/XDC's normal output. BOth should be obvious. These settings have no true default, as they can be set in the Profile facility.

Addresses The virtual address of each line of the display will appear on the left side of the panel.

Offsets The offset from the start of the data's object (load module, csect or dsect) will appear on the left side of the panel.

***Data*/Decimal** Displacement fields of machine instructions will be displayed as hexadecimal or decimal numbers as desired. Operative ONLY when source statement support is not being used.

***Ebcdic*/AScii** Memory will be interpreted in EBCDIC or ASCII format, as desired.

***Wide*/Narrow** Shows 32 bytes of memory (default) per line, or 16 bytes per line.

Instruction/Data/Nobias z/XDC makes a number of decisions in the Format process, many of which are dependent on a great many other variables. Consequently, the Where/Format command may, under certain circumstances, mis-interpret data as instructions if the format command is attempted within data rather than object code. The Instruction, Data and Nobias parameters affect z/XDC's determination in the Format command. Please see Help Commands Format for a thorough understanding of how to use these parameters.

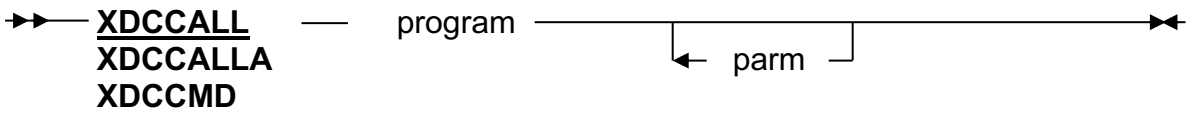
***SHOWMCODE*/HIDEMCODE** z/XDC will SHOW or HIDE macro expansions. If you use these parameters you MUST specify an address-expression.

Links When storage is being formatted via a disassembly, and if a line of display shows a snippet of data that is 3, 4, or 8 bytes long, then LINKS will attempt to resolve that snippet as a storage pointer. Then it will display that resolution as a comment on the display line. This is useful, but VERY CPU-EXPENSIVE.

Remarks: Whenever the user is far "afield" in debugging a problem, it is very useful to issue Where and get back to "where" the program is executing. While the Where offers all the parameters of the Format command, it's easiest just to tab over to column one of any display, key in a "W" and press Enter. Voila! You're back! The Where command also works from the command line. This is a very cool command.

Examples: W ← z/XDC will disassemble where the PSW points to. What could be simpler?

More Info: Enter Help COmmands Where. See also Help COmmands FOrmat.



Invoke z/XDC from the TSO READY prompt, or in the Batch.

program The name of a module.
parm The parameters of the module, separated from one another by spaces.

Remarks: This is how you invoke z/XDC, either interactively or in batch mode.

If you wish to execute APF-authorized, then you use the XDCCALLA form of the command. In fact, many z/XDC users use "XDCCALLA IEFBR14" merely to gain APF-authorization for the purpose of investigating system routines, or code running in the Private Areas of other address spaces.

XDCCALL is also used to invoke z/XDC in batch programs. To do that, alter your EXEC statement from

"// EXEC PGM=myprogram, PARM='parm1 parm1 parm2 …"

to

// EXEC PGM=XDCCALL, PARM='my program parm1 parm2 …"

If you need to invoke your program APF-authorized Use the XDCCALLA command in your EXEC statement.

When invoked, XDCCALL does several things:

- XDCCALL(A) loads into a task in the address space (TSO Private Area or batch address space);
- It ATTACHes your program as a subtask.
- It specifies that XDCCALL is the ESTAI for your program;
- It uses the #DIE command to create a S0C1 abend at the entry point of your code;
- It writes a "DEAD" message to your TSO display, or to the Operator Console if in batch.

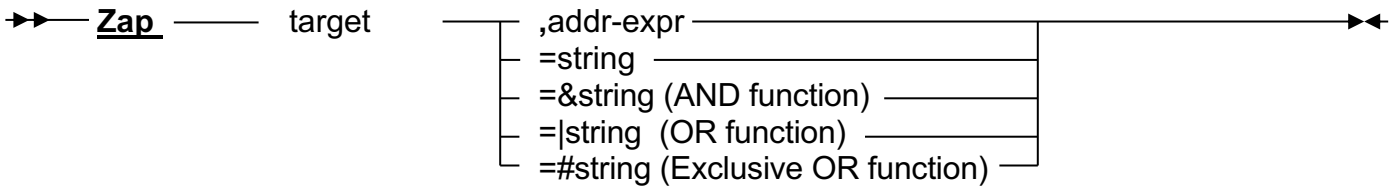
If in TSO, you can begin debugging. If in batch you'll use Option 3 from the z/XDC Startup Panel, and select your batch job (with an "S") to establish a connection to your batch job.

Use the XDCCMD alias of XDCCALL to establish a debugging session on a TSO Command processor. In this case, z/XDC passes parameters, if any, to the user's program in the format of a command buffer pointed to by a CPPL (pointed to by R1).

This command may not be used in Foreign Address Space Mode.

Examples: XDCCMD MYTSOCOMMAND ← the user wishes to work on a TSO command in the Foreground.
XDCCALLA IEFBR14 ← The user wishes to launch an APF-authorized z/XDC session for some cool purpose.

More Info: There is no reference for the XDCCALL "command" under HELP COMMANDS. Frank says, "That's because XDCCALL isn't a command. It's a callable module." So, please issue HELP XDCCALL for more information. See also Help Debugging Batchjobs.



Over-write the contents of one memory location with the contents of another memory location, or with a stated value. Shortcut command equivalent: "Z"

target The location in memory that you wish to over-write. This can be within the Private Area, the Common Area, a Dataspace, Foreign Address spaces, real storage, the registers, and the PSW. For zapping read-only storage, see Help COmmands Set Zap.

,addr-expr The address-expression whose value is to be written to the target. You must use either a comma, or a blank to delimit the second address-expression (but not both!).

=string A stated value that you wish overwrite the target with. There can be NO space between "target" and the =string value.

=&string The string is to be AND'd, byte by byte, into the "target address-expression."

=|string The string is to be OR'd, byte by byte, into the "target address-expression." You can use "|" or "¦".

=#string The string is to be Exclusive OR'd, byte by byte, into the "target address-expression."

Remarks: Zap is one of the most useful and powerful of z/XDC's commands. Accordingly, Zap must be used with caution, and care. You can zap literally anywhere if your security system will allow you to do so. **THIS MEANS THAT YOU CAN CRASH A SYSTEM WITH AN INCORRECT ZAP!**

To use Zap as a shortcut command, put a "Z" in column one of a z/XDC display, tab to the item you wish to change, and merely overtype the old value with the new value! You may overtype any field on the display in which a TAB operation stops the cursor.

If you wish to zap the PSW to change the execution location, then Zap will work on ONLY resume address portion of the PSW (which cannot be "overtyped" as just described). In that case ZAP may ONLY be issued from the command line. That is, "ZAP PSW <address-expression>". For other changes to the PSW, see the SET PSW command.

This command may be used in Foreign Address Space Mode. This is powerful, and scary. ***Vaya con Dios, pero tenga CUIDADO!***

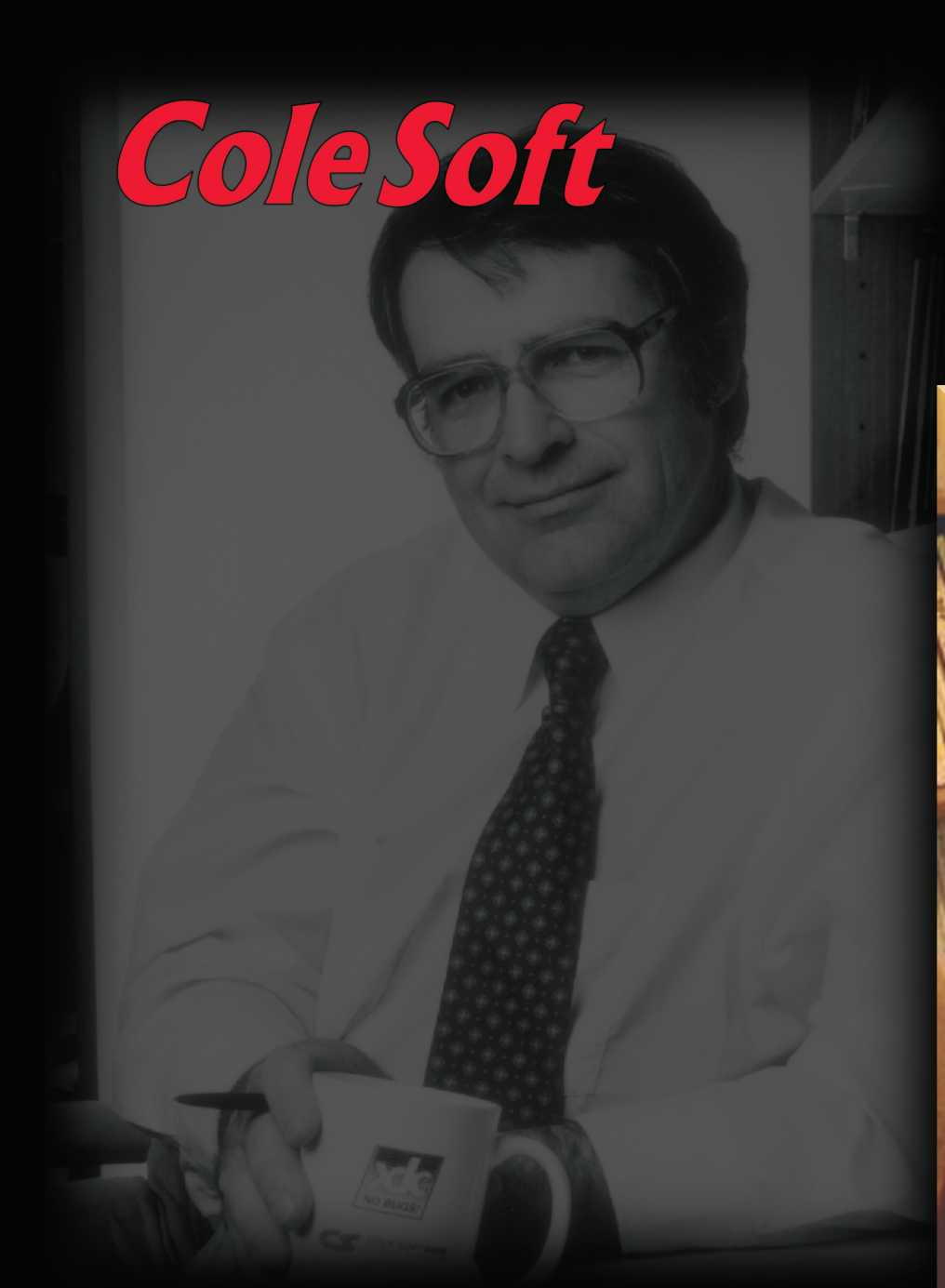
For zapping read-only storage, see Help COmmands Set Zap.

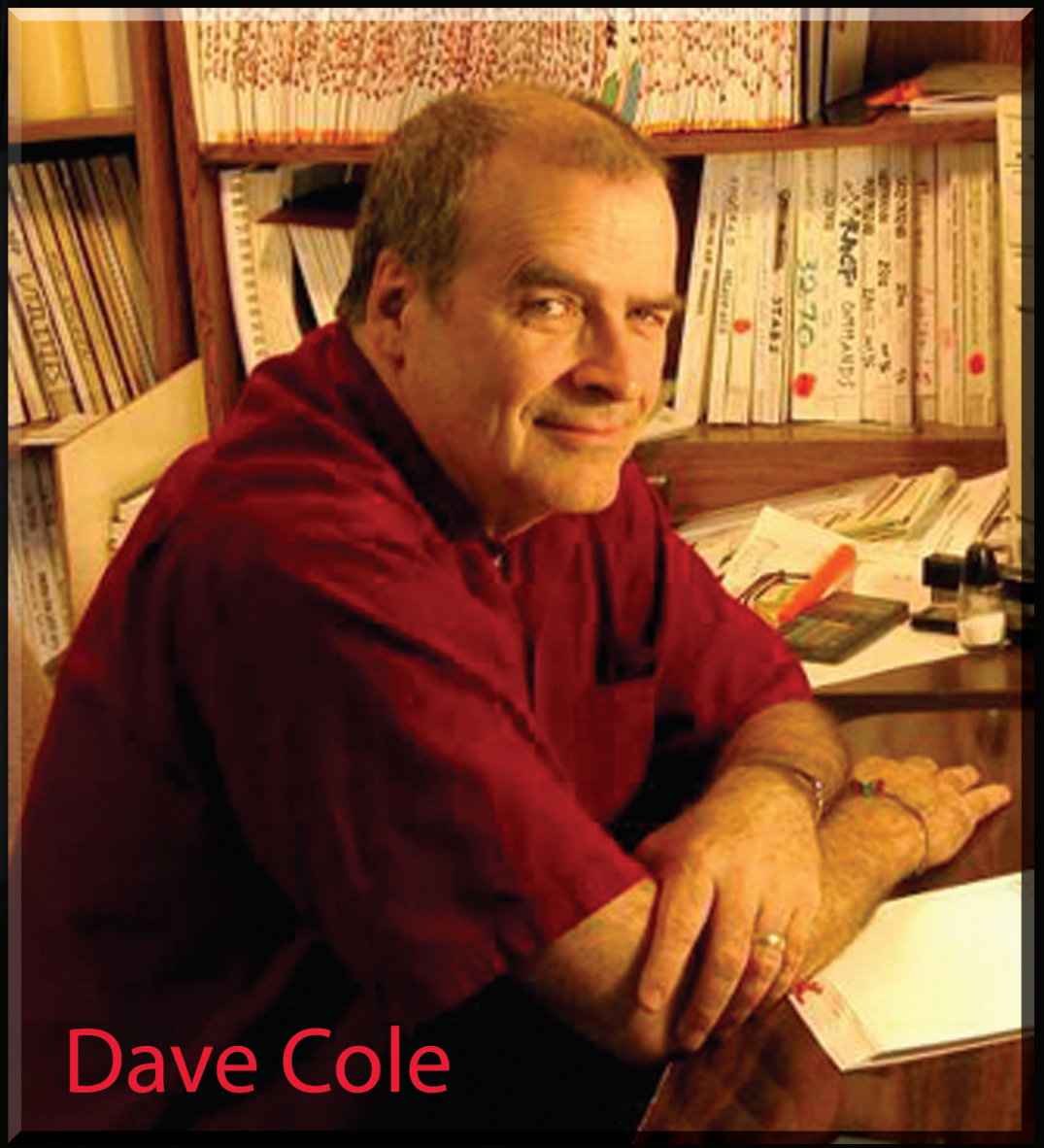
z/XDC is now case-agnostic and accepts mixed-case zap strings. For idle research, see Help COMmands SEt ASIS and LISt ASIS.

Examples: Z R0 0FFFFFFF ← Overwrites R0 with the new value of "0FFFFFFF".
ZAP .LABEL='SOME TEXT' ←write the upper case letters "SOME TEXT" into the address expression ".LABEL".

More Info: Enter Help COmmands Zap and Help COmmands Zap Targets. See also Help COmmands Zap Boolean, Help Commands Verify, and SET PSW. If you want to move "chunks" of storage around, see Help COmmands COpy.

ColeSoft





Dave Cole