colesoft™

# Z/XDC®

# INSTALLATION GUIDE

## Z/XDC® Release v3.1 for z/OS

David B. Cole

Z/XDC®

Z/XDC® is a member of the XDC® (Extended Debugging Controller®) family of products
COPYRIGHT "Izzi Software LLC" - 2010-2026. ALL RIGHTS RESERVED

Generated 09/03/2026, 8:42pm

online PDF Documents

# PREFACE

## PROPRIETARY LEGEND

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

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 The Product may not be published, reproduced, distributed or made available to third parties for any purpose without the expressed written permission of Owner or Licensor.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

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

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

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

## CONTACTING COLESOFT

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

Phone: **540-456-8210**
E-Mail: **sales@colesoft.com**
Home Page: **www.colesoft.com**

Our Technical Support contacts are:

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

Our Customer Services contacts are:

Phone: **540-456-8210**
E-Mail: **support@colesoft.com**
Home Page: **www.colesoft.com**

Our snail mail address is:

Address: **ColeSoft Marketing, Inc.**
**440 Monticello Ave Ste 1802**
**PMB 380764**
**Norfolk, Virginia 23510-2670**
**USA**

## ONLINE PRESENCE

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

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

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

## TRADEMARKS

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

**Extended Debugging Controller**®, **XDC**®, and **Z/XDC**® are registered trademarks of Izzi Software.

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.

## PRODUCT MANUALS

Z/XDC's documentation consists of the manuals shown below. The manuals are provided in PDF format. All manuals can be downloaded from www.colesoft.com under Support.

| | |
|---|---|
| Install Guide | Describes how to install Z/XDC |
| User Guide | Explains how to use Z/XDC |
| Commands Reference | Describes all of Z/XDC's commands |
| Messages Reference | Explains all numbered messages issued by Z/XDC |
| Maintenance and New Functionality Guide | Describes what is new if Z/XDC (updated constantly) |
| Primer | A basic tutorial to help the beginner get started with Z/XDC |
| Quick Reference | A comprehensive syntax reference for all of the Z/XDC commands |

## ADDITIONAL MANUALS

Z/XDC customers may make as many copies of this manual as they feel is necessary for the legitimate use of Z/XDC within their organization. Existing customers may download from our web site (www.colesoft.com/product-support/zxdc-support/zxdc-documentation) printable copies of all of Z/XDC's manuals. Each manual is available in PDF format.

# Contents


PREFACE . . . ii
PROPRIETARY LEGEND . . . ii
CONTACTING COLESOFT . . . ii
ONLINE PRESENCE . . . ii
TRADEMARKS . . . iii
PRODUCT MANUALS . . . iii
ADDITIONAL MANUALS . . . iii
CONTENTS . . . v
The Z/XDC Product Shipment Package . . . 1
The $README File . . . 1
Reintegration Summary . . . 1
The High level Qualifier (hlq) . . . 1
Z/XDC SYSTEM INTEGRATION . . . 3
Step 1: Adding Z/XDC's Permanent Product Libraries to Your System . . . 3
Adding XDCPLPA Load Modules to Common Storage . . . 3
Adding the XDCLINK and XDCLINKE Libraries to Your Linklist . . . 4
Adding XDCGUI to APF authorized list . . . 5
Copying XDC Load Modules into Existing Linklist Libraries . . . 5
Adding XDC Load Libraries to the Linklist Concatenation . . . 5
Leaving XDC Out of the Linklist . . . 6
Making XDCCALLA and XDCCMDA APF-Authorized . . . 6
Adding XDCCLIST . . . 8
Adding XDCMLIB, XDCPLIB and XDCTLIB to ISPF . . . 10
Copying Z/XDC's ISPF Elements into Existing ISPF Libraries . . . 10
Adding Z/XDC's ISPF Element Libraries to TSO Logon Procs . . . 10
Using XDCLBDEF to Access Z/XDC's ISPF Element Libraries Only When Needed . . . 11
Setting Up the XDCPANEL Panel or the XDCLBDEF Clist in ISR@PRIM (or ISRUTIL) . . . 11
Choosing Between Invoking XDCPANEL Directly or Indirectly (via XDCLBDEF) . . . 11
Adding the XDCCMDS Scripts Library . . . 12
Step 2: ReIPL (Maybe) - A Checklist . . . 13
Step 3: Defining Your License to Use Z/XDC . . . 13
Step 4: Defining Z/XDC to Your Computer's Security System . . . 14
For More Information About Z/XDC Security ... . . . 15
Z/XDC's Standard Security Method . . . 15
Using Z/XDC Prior to Creating Security Definitions . . . 16
Using Z/XDC in RACF Protected Systems . . . 17
RACF - Initial Setup: Deny All Accesses by Everyone . . . 17
RACF - Permitting Some Accesses to Some People . . . 18
RACF - Permitting Accesses: Examples . . . 18
RACF - Additional Information . . . 19
Using Z/XDC in ACF2 Protected Systems . . . 19
ACF2 - Initial Setup . . . 19
ACF2 - Permitting Accesses . . . 20
Using Z/XDC in Top Secret Protected Systems (TSS) . . . 22
TSS - Initial Setup . . . 22
TSS - Permitting Some Accesses to Some People . . . 23
TSS - Permitting Accesses: Examples . . . 23
TSS - Additional Information . . . 24
Step 5: c/XDC Requires an OMVS Segment . . . 24
Step 6: Installing Z/XDC's Service SVC, Legacy Hook SVC and System Interface . . . 24

About Z/XDC's SVCs . . . 25
Step 7: Installing Z/XDC's Server Job . . . 25
Bouncing server/XDC - Is it Safe? . . . 26
Setting up cdf/XDC . . . 26
Step 8: The Installation Verification Test . . . 28
Testing Authorized Debugging . . . 32
Step 9: cdf/XDC Installation Verification . . . 33
Step 10: Creating a Renamed Clone . . . 35
Considerations for Using a Renamed Z/XDC . . . 35
Step 11: Making hlq.XDCV31.XDCADATA the Default MAPLIB Library . . . 36
Step 12: Creating a Default Profile: TFSDPV31 . . . 37
Reasons to Create or not Create a System-Wide Default Profile . . . 37
How to Create Z/XDC's System-Wide Default Profile . . . 38
Which System-Wide Default Profile Values You Might Consider Changing . . . 39
Individual PF-key Assignments: . . . 39
Session Log File Allocation: . . . 41
Changing the Default Scripts Library (used by Z/XDC's READ command): . . . 42
//XDCPRINT File Characteristics: . . . 43
Step 13: Exit Routines . . . 43
A User Communications Interface (UCI) Exit . . . 43
Consolidated Exits . . . 44
An "Initialization" Exit . . . 44
A "Resumption" Exit . . . 44
A "Termination" Exit . . . 44
A "User SVC" Exit . . . 44
A "User Commands" Exit . . . 44
A "Front-end/Back-end" Routine . . . 44
Step 14: Setting up the gui/XDC Server . . . 44
Add the gui/XDC Server proc . . . 46
Adding a VHOST to z/OS HTTP Server . . . 46
Step 15: Installing the VS Code Extension . . . 50

# List of Figures


1 Adding XDCCALLA and XDCCMDA to IKJTSOxx . . . 8
2 ISPF Panel Mods for Invoking the XDCPANEL Panel . . . 12
3 Z/XDC's RACROUTE Macro Usage in RACF Protected Systems . . . 17
4 Z/XDC's RACROUTE usage in ACF2 Protected Systems . . . 19
5 Z/XDC's RACROUTE usage in Top Secret Protected Systems . . . 22
6 Typical XDCCDF member to be added to a VTAMLST library for cdf/XDC . . . 27
7 Typical SYSIN File Parameters for cdf/XDC . . . 27
8 Model Startup JCL for server/XDC . . . 27
9 Z/XDC's Startup Panel in ISPF . . . 29
10 Initial Screen Displayed by Z/XDC on a Classic 43x80 3270 Terminal . . . 30
11 Initial Screen Displayed by Z/XDC on a Terminal Set to Large Dimensions . . . 31
12 Error When Z/XDC is Not Properly Installed into ISPF . . . 31
13 Starting Z/XDC Authorized from its Startup Panel in ISPF . . . 32
14 Model JCL for Testing cdf/XDC . . . 33
15 Logon Screen for VTAM Connections to Z/XDC's Cross Domain Facility . . . 34
16 cdf/XDC Job Selection menu . . . 34
17 Z/XDC's Profile Menu System . . . 39
18 Default PF-key Definitions . . . 40
19 Profile Settings for Z/XDC's Session Log . . . 41
20 Profile Settings for READ, ZAP and PARSEORDER . . . 42
21 Profile Settings for AUTOMAP, PRINT and Other Settings . . . 43
22 Sample XGUIPROC proc . . . 46
23 Sample httpd.conf configuration statements . . . 47
24 Sample httpd.conf configuration statements . . . 48
25 Steps to install VS Code extension . . . 50

# List of Tables

1 Z/XDC's ISPF Clists, Panels, Tables and Message Elements . . . . . 10

# THE Z/XDC PRODUCT SHIPMENT PACKAGE

The Z/XDC "Product Shipment Package" can be obtained via download from our website: http://www.colesoft.com

- The Shipment Package is just a tersed file that must be downloaded to your z/OS System.
- There also are a couple of text files containing instructions and JCL that need to be downloaded as well.

The Shipment Package can be used:

- By new customers wishing to perform an initial product install of Z/XDC at the latest maintenance level
- By existing customers wishing to update any older product level, bringing it up to the latest maintenance level.

## The $README File

The instructions for installing or updating the Permanent Product Libraries are contained in a $README file. If you do not already have this file, you can pull the latest copy from our website at https://colesoft.com/support/maintenance-files. If you do already have the $README, make sure it's the latest version by comparing its Change History to that of the website's copy.

$README is a cookbook that takes you through the entire process of:

- Selecting a High Level Qualifier (hlq) for your Permanent Product Libraries,
- Downloading the Product Shipment Package,
- Extracting Product Libraries into Temporary Product Libraries,
- Using the Temporary Product Libraries to create or replace your Permanent Product Libraries.

At this point, you need to printout the $README and follow its instructions to (re)build your Permanent Product Libraries. Go ahead. Do it now. I'll wait...

———————-

Are you done? Good. You are now ready to begin the Integration Phase of the Z/XDC installation. That documentation follows below.

## Reintegration Summary

Important! If you are applying a maintenance update (and not an initial install), when you are done, there are certain reintegration steps that must **always** be done:

- You must always reintegrate all load modules into your system libraries and concatenations. For details, see:
  - "Adding XDCPLPA Load Modules to Common Storage" on page 3,
  - and "Adding the XDCLINK and XDCLINKE Libraries to Your Linklist" on page 4.
- If the maintenance update has made any significant other changes, those will have to be reintegrated as well. (The $README process provides guidance for identifying product elements that have changed.)

## The High level Qualifier (hlq)

The $README file explains how to (re)build your Permanent Product Libraries from the Shipment Package. The Permanent Product Libraries will have names that start with a High Level Qualifier that you chose when you built the libraries. The remainder of the installation process will use the Permanent Product Libraries to integrate Z/XDC into your system.

Throughout this Guide, the individual Permanent Product Libraries will be referred to as **hlq.XDCV31.XDCwhatever**, where:

- **hlq** is the High Level Qualifier that you chose when you created the Permanent Product Libraries.
- **XDCwhatever** identifies a specific library.

# Z/XDC SYSTEM INTEGRATION

So much for the easy part! There remains several tasks that need to be performed in order to integrate (or reintegrate) Z/XDC into your z/OS system. Let's get right to it.

## Step 1: Adding Z/XDC's Permanent Product Libraries to Your System

The following Z/XDC's Permanent Product Libraries need to be integrated into your system by either copying them into or concatenating them to your corresponding z/OS system libraries. The choice depends upon your local installation practices.

There are the four load libraries:

- **hlq.XDCV31.XDCLINK**
- **hlq.XDCV31.XDCLINKE**
- **hlq.XDCV31.XDCPLPA**
- **hlq.XDCV31.XDCGUI**

And six FB-80 libraries of various sorts:

- **hlq.XDCV31.XDCCLIST**
- **hlq.XDCV31.XDCCMDS**
- **hlq.XDCV31.XDCMLIB**
- **hlq.XDCV31.XDCPLIB**
- **hlq.XDCV31.XDCSRVER**
- **hlq.XDCV31.XDCTLIB**

Also, there are seven additional libraries that do not need to be integrated, but do need to be available for customer access:

- **hlq.XDCV31.XDCADATA**
- **hlq.XDCV31.XDCASM**
- **hlq.XDCV31.XDCDWARF**
- **hlq.XDCV31.XDCJCL**
- **hlq.XDCV31.XDCMACS**
- **hlq.XDCV31.XDCMMEF**
- **hlq.XDCV31.XDCSAMP**

The following discussions make several references to adding information to various members of your system's PARMLIB libraries. It is assumed that you have sufficient knowledge of PARMLIB to understand the discussion. If not, then please refer to IBM's Initialization and Tuning Reference Manual [1] for more information, or consult with your systems programmer, as appropriate. If necessary, you may also, of course, contact us for assistance.

### Adding XDCPLPA Load Modules to Common Storage

**hlq.XDCV31.XDCPLPA** contains load modules XDC and XDC31.

[1] All public IBM manuals are available online. Generally, they can be found easily with simple Google searches for their titles.

- XDC is Z/XDC's primary load module. It will reside in 24-bit storage.
- XDC31 contains that part of Z/XDC that resides in 31-bit storage.

These modules need to be placed either into the system's PLPA[2] or its DLPA[3] You can do so in any of the following ways:

- By copying the modules to an existing PLPA library, and then reIPLing with CLPA specified.
- By adding hlq.XDCV31.XDCPLPA's dsname to the PLPA library list, and then reIPLing.
- By issuing a **'SETPROG LPA,ADD,MODNAME=(XDC,XDC31),DSN=hlq.XDCV31.XDCPLPA'** operator command to cause the modules to be immediately loaded into DPLA.
- By adding a **COM='SETPROG LPA,ADD,MODNAME=(XDC,XDC31),DSN=hlq.XDCV31.XDCPLPA'** statement to an active COMMNDxx member of your PROCLIB libraries to dynamically load the XDC and XDC31 load modules into the system's DLPA at IPL time. *[This is the recommended method.]*

The XDC and XDC31 load modules will have to be reinstalled into the PLPA or DLPA every time you apply Z/XDC maintenance to the hlq.XDCV31.XDCPLPA library! That's because every maintenance update will make at least systemic changes to all load modules. The $README file contains instructions for doing this.

If you want to add the modules right now (without waiting for an IPL), you can use a SETPROG LPA,ADD operator command to load them into the DLPA immediately, and then a DISPLAY PROG commands to report the results. The commands are:

- SETPROG LPA,ADD,MOD=(XDC,XDC31),DSN=hlq.XDCV31.XDCPLPA
- DISPLAY PROG,LPA,MOD=XDC
- DISPLAY PROG,LPA,MOD=XDC31

These commands cause the XDC and XDC31 load modules to be loaded into common storage, making them immediately available for use. Then their locations are displayed.

If you have older instances of XDC and XDC31 already loaded into the DLPA:

- That's ok because the newer instances will supersede the older ones in the system's DLPA search order.
- But if you wish to recover wasted CSA storage, then after issuing the SETPROG LPA,ADD command to load the newer instances into the DLPA, you can then issue a SETPROG LPA,DELETE,operands command to delete the older instances of XDC and XDC31. The precise command is:
  - **SETPROG LPA,DELETE,MOD=(XDC,XDC31),FORCE=YES,OLDEST**
- Issue the above command repeatedly until it fails (in case multiple older instances exist).[4]

### Adding the XDCLINK and XDCLINKE Libraries to Your Linklist

Both the ***hlq***.XDCV31.XDCLINK and ***hlq***.XDCV31.XDCLINKE libraries contain those Z/XDC load modules that should be added to the system's linklist libraries.

- XDCLINK is a PDS that contains load modules that may reside in either PDS or PDSE libraries.

[2]The PLPA is the "Pageable Link Pack Area". It is a library of load modules that get permanently loaded into a common area of storage at IPL time. When changes are made to the modules in the PLPA library, those changes do not get reloaded into common storage until the next IPL with CLPA specified.

[3]The DLPA is the **Dynamic Link Pack Area**. The name refers to those load modules that have been brought into common storage via a SETPROG LPA,ADD operator command. When you need to add or change modules in common storage, using the SETPROG command allows you to do so without requiring an IPL.

[4]Use of the **OLDEST** operand keeps the command from removing the newest instance, i.e. the instance that you just loaded with the SETPROG LPA,ADD command. So repeating the *DELETE* command until it fails will remove only excess instances. Your newly added instance will be retained.

- (But see footnote [5] for one exception).
- XDCLINKE is a PDSE that contains program objects that cannot reside in PDS libraries.

You may either copy the XDCLINK and XDCLINKE libraries to existing linklist libraries or add the libraries to your linklist concatenation.

Alternatively, you may leave the XDCLINK and XDCLINKE libraries out of the linklist entirely and, thereby, require users to access the Z/XDC modules via //JOBLIB or //STEPLIB. If you elect to do this, though, then you must add both libraries to the system's APF libraries list (i.e. to the APF section of an active PROGxx member of the PARMLIB libraries).

### Adding XDCGUI to APF authorized list

**hlq.XDCV31.XDCGUI** contains the XDCLMGUI and XDCLMLPA programs.

- XDCLMGUI is the primary program for Z/XDC's GUI Server.
- XDCLMLPA contains authorized and cross memory functionality needed by XDCLMGUI. XDCLMLPA will be dynamically added to the DLPA by XDCLMGUI.

Neither of these programs should be copied or added to the LINKLIST or into the LPA.

The XDCGUI library needs to be added to the system's APF libraries list (i.e. to the APF section of an active PROGxx member of the PARMLIB libraries).

If you do not wish to IPL to add XDCGUI to the APF list, you can use the SETPROG APF operator command to dynamically add the library. Example:

**SETPROG APF,ADD,DSN=hlq.XDCV31.XDCGUI,VOL=volser**

### Copying XDC Load Modules into Existing Linklist Libraries

If you elect to copy the ***hlq***.XDCV31.XDCLINK and ***hlq***.XDCV31.XDCLINKE load modules into existing linklist libraries:

- After copying the Z/XDC load modules, perform an LLA REFRESH to make them usable (assuming the target linklist libraries did not create and expand into extra extents).
- This copying process will have to be repeated every time you apply Z/XDC maintenance.
- The copying process may cause your target linklist libraries to create and expand into extra extents. If so, that will cause programs that try to use those load modules to fail with I/O errors. But don't worry. It's recoverable.

You can use the SETPROG LNKLST operator command to create and activate a rebuilt linklist. The following commands will work:

**SETPROG LNKLST,DEFINE,NAME=TEMP,COPYFROM=CURRENT,NOCHECK**
**SETPROG LNKLST,ACTIVATE,NAME=TEMP**
**DISPLAY PROG,LNKLST,NAME=CURRENT**

If you have to do this, be aware that existing batch jobs and TSO sessions will not see the revised linklist, but new jobs will. TSO sessions, for example, will have to be bounced before they will have access to the new load modules.

### Adding XDC Load Libraries to the Linklist Concatenation

If you elect to add the hlq.XDCV31.XDCLINK and hlq.XDCV31.XDCLINKE libraries to the linklist concatenation, then:

[5] Except the XDCLCNSE load module. That must reside in a PDS. It may not reside in a PDSE.

- Add hlq.XDCV31.XDCLINK's and hlq.XDCV31.XDCLINKE's dsnames to the LNKLST section of an active PROGxx member of the PARMLIB libraries.
- If your active IEASYSxx member of PARMLIB has LNKAUTH=APFTAB specified, then the ***hlq***.XDCV31.XDCLINK and ditto.XDCLINKE libraries will have to be made APF authorized by adding their names to the APF section of an active PROGxx member of the PARMLIB libraries.
- Historically, when you change the linklist, you need to wait for an IPL before those changes are activated. However, these days you can use the *SETPROG APF,operands* and *SETPROG LNKLST,operands* commands to authorize the ***hlq***.XDCV31.XDCLINK and ditto.XDCLINKE libraries dynamically and activate the linklist changes immediately. Examples:

  **SETPROG APF,ADD,DSN=*hlq*.XDCV31.XDCLINK,VOL=volser**
  **SETPROG APF,ADD,DSN=*hlq*.XDCV31.XDCLINKE,VOL=volser**
  **SETPROG LNKLST,DEFINE,NAME=TEMP,COPYFROM=CURRENT,NOCHECK**
  **SETPROG LNKLST,ADD,NAME=TEMP,DSN=*hlq*.XDCV31.XDCLINKE,ATTOP**
  **SETPROG LNKLST,ADD,NAME=TEMP,DSN=*hlq*.XDCV31.XDCLINK,ATTOP**
  **SETPROG LNKLST,ACTIVATE,NAME=TEMP**
  **DISPLAY PROG,LNKLST,NAME=CURRENT**
  **F LLA,REFRESH**

  These commands are documented in IBM's z/OS: MVS System Commands[6].

### Leaving XDC Out of the Linklist

If you elect to leave the ***hlq*.XDCV31.XDCLINK** and ***hlq*.XDCV31.XDCLINKE** libraries and modules out of the linklist entirely:

- The ***hlq***.XDCV31.XDCLINK and ***hlq***.XDCV31.XDCLINKE libraries must be made authorized by adding their dsnames and locations (*volser*) to the APF section of an active PROG*xx* member of the PARMLIB libraries.
- You may issue a SETPROG APF command to activate the authorization immediately without requiring an IPL.
- Users will have to use //JOBLIB or //STEPLIB (or equivalents) in order to use Z/XDC. But be aware that when they want to use Z/XDC authorized, then the //JOBLIB or //STEPLIB concatenation must **NOT** include nonauthorized libraries. (Doing so "poisons" the concatenation, causing all libraries to be treated as being nonauthorized.)
- Also note, if you need to run Z/XDC authorized under ISPF, then you will NOT be able to use ISPF's **LIBDEF** facility to access the ***hlq***.XDCV31.XDCLINK and ***hlq***.XDCV31.XDCPLPA libraries. This is because **LIBDEF** does not support authorized libraries.

### Making XDCCALLA and XDCCMDA APF-Authorized

XDCCALL**A** and XDCCMD**A** are authorized copies of the (nonauthorized) XDCCALL and XDCCMD load modules. All four load modules are located in the ***hlq***.XDCV31.XDCLINK library. They all can be used both as TSO commands and as batch programs (// EXEC PGM=XDCCALLA, for example).

The **A variants** (XDCCALLA/XDCCMDA) allow users to debug APF authorized programs and TSO Commands. These modules have been built with the "AC=1" attribute and will need to reside in an APF-authorized library.

However, in order to use XDCCALLA and XDCCMDA as authorized commands within TSO, there is more to be done!

- You must also add their names to the AUTHCMD parameter in an active IKJTSOxx member of the PARMLIB libraries.

[6] All public IBM manuals are available online. Generally, they can be found easily with simple Google searches for their titles.

- Then in order to activate the new AUTHCMD list, you must use TSO’s PARMLIB command to cause TSO to refresh its AUTHCMD list in storage.

For details, see Figure 1 on page 8. For more complete information, see IBM's **Initialization and Tuning Reference** manual, the **TSO/E Customization manual** and the **RACF Command Language Reference** manual.

Use an editor to add XDCCALLA and XDCCMDA to the list of names for the AUTHCMD parameter (found in an active IKJTSOxx member of the PARMLIB libraries):

```
AUTHCMD NAMES( ...
        ...
        XDCCALLA XDCCMDA
        ...
        )
```

Use the PARMLIB TSO command to validity check and then activate your changed IKJTSOxx. The PARMLIB command can be issued either from TSO's READY prompt or from ISPF's Command Shell (**=6**) panel.

- To validity check your change, type: PARMLIB CHECK(xx)
- To activate your change, type: PARMLIB UPDATE(xx)
- To display your change, type: PARMLIB LIST(AUTHCMD)

where "xx" matches the last two characters of the name of an IKJTSOxx member of a PARMLIB library that you are updating.

If system security does not permit you to use the PARMLIB command, then you need to have a security officer give you UPDATE authority to the PARMLIB resource of the TSOAUTH class. If your security system is RACF, then the following commands can be issued from TSO's READY prompt or from ISPF's Command Shell (**=6**) panel:

- To display the resource, type: RLIST TSOAUTH PARMLIB ALL
- To give a user UPDATE authority, type: PERMIT PARMLIB CLASS(TSOAUTH) ACCESS(UPDATE) ID(*userid*)

Figure 1: Adding XDCCALLA and XDCCMDA to IKJTSOxx

Note that the above procedure is a step that is only needed for permitting XDCCALLA and XDCCMDA to run **authorized** in TSO. XDCCALLA also can run in a batch job (**// EXEC PGM=XDCCALLA,...**). In this case, it is irrelevant whether or not X DCCALLA's name is in **IKJTSOxx**'s **AUTHCMD** list, XDCCALLA will run authorized in the batch regardless.

If you do not wish to permit authorized debugging, then you can create security system rules to prohibit it. See **Step 4: Defining Z/XDC to Your Computer's Security System** on page 14 for more information.

### Adding XDCCLIST

The hlq.XDCV31.XDCCLIST library contains two clists that are used in an ISPF interface to Z/XDC. They are named XDCCLIST and XDCLBDEF.

- XDCCLIST is required. It is invoked internally by the Z/XDC Startup Panel (an ISPF panel) to allocate datasets prior to passing control to Z/XDC.
- XDCLBDEF is optional but recommended. It can be used if you wish to avoid adding the ***hlq***.XDCV31.XDCMLIB, ditto.XDCPLIB, ditto.XDCTLIB and ditto.XDCCLIST libraries to your TSO logon procs.

You may install these clists into your system in any of the following ways:

- You may copy both clists to an existing clist library.
- You may copy just the XDCLBDEF clist to an existing clist library and modify its default input parameters (the "CLIST(dsname)" operand) to point to the Permanent Product Library that contains the XDCCLIST clist.

- (Recommended) You may add the hlq.XDCV31.XDCCLIST library to your existing clist library concatenations.

If you copy the hlq.XDCV31.XDCCLIST library members to an existing clist library, then be aware of the following:

- The hlq.XDCV31.XDCCLIST library is distributed with RECFM=FB and LRECL=80. The clists within it, however, are structured so that they can be copied to a RECFM=VB clist library without damage.
- If your target clist library has RECFM=VB, then you must use ISPF's Move/Copy Utility (=3.3) to perform the copy. IEBCOPY cannot do the necessary record format conversion.
- This copying process will have to be repeated every time the factory default instances of the clists are changed by product maintenance. (The $README process provides guidance for identifying product elements that have changed.)

If you add the hlq.XDCV31.XDCCLIST library to your clist library concatenations:

- Your clist libraries must all be RECFM=FB and LRECL=80. z/OS does not support concatenations of these libraries when they have differing record formats.
- You must add a DD card for the hlq.XDCV31.XDCCLIST library to the //SYSPROC DD concatenation of every logon proc for every TSO user whom you expect to use Z/XDC.[7]
- Alternatively, if you use logon clists (e.g. PARM=%LOGON in the TSO logon procs), you can recode that clist to add the hlq.XDCV31.XDCCLIST library to the //SYSPROC DD concatenations.
- It does not matter where, in the //SYSPROC DD concatenation, the hlq.XDCV31.XDCCLIST library occurs (first, last, middle). Duplicate instances of the clists from other sources are not expected.
- The first library in any concatenation either must have the largest BLKSIZE or must specify DCB=BLKSIZE=value that is as large as or larger than the library having the largest BLKSIZE. The hlq.XDCV31.XDCCLIST library's BLKSIZE is 27,920.

[7] (Note, systematic use of a **//ddname INCLUDE procname** statement into your LOGON procs can ease this sort of burden considerably. See **z/OS MVS: JCL Reference** for more information.)

### Adding XDCMLIB, XDCPLIB and XDCTLIB to ISPF

The hlq.XDCV31.XDCMLIB, ditto.XDCPLIB and ditto.XDCTLIB libraries contain ISPF panels, messages and table modules (respectively) used for integrating Z/XDC into ISPF. (See Figure 3 on page 10.) You may either copy these libraries to existing ISPF libraries, or you may add them to your existing library concatenations.

| Element Name* | Type | Purpose |
|---|---|---|
| TFSPROF | Table | An ISPF profile that causes ISPF to pass all PF-keys and ISPF commands through for processing by Z/XDC. |
| XDCV31A | Panel | A dynamic panel used for nearly all of Z/XDC's displays when it is communicating to the user's terminal via ISPF's display service routines. |
| XDCV31B | Panel | A special purpose panel used in miscellaneous situations by Z/XDC's interface to ISPF. |
| XDCPANEL | Panel | Z/XDC's Startup Panel to be installed into ISPF so that users can more easily invoke Z/XDC. |
| XDCPHELP | Panel | Built-in Help for Z/XDC's Startup Panel (XDCPANEL). |
| XDCPHLPx | Panel | Various additional Built-in Help panels for Z/XDC's Startup Panel. |
| XDCM00 | Message | Error messages used by the Z/XDC Startup Panel. |
| XDCCLIST | Clist | Invoked internally by XDCPANEL to allocate datasets and then call XDCCALL [et.al.] to start the requested debugging session. |
| XDCLBDEF | Clist | Can be invoked by users to dynamically allocate Z/XDC's ISPF libraries and display the XDCPANEL panel. |

Table 1: Z/XDC's ISPF Clists, Panels, Tables and Message Elements

Alternatively, you may avoid adding the ***hlq***.XDCV31.XDCMLIB, ***hlq***.XDCV31.XDCPLIB and ***hlq***.XDCV31.XDCTLIB libraries to your ISPF and other system libraries and instead use the **XDCLBDEF** clist to dynamically allocate these libraries only when a user wants to run Z/XDC. (This is not recommended due to usage restrictions that arise using this method.)

### Copying Z/XDC's ISPF Elements into Existing ISPF Libraries

If you elect to copy ***hlq***.XDCV31.XDCMLIB, ***hlq***.XDCV31.XDCPLIB and ***hlq***.XDCV31.XDCTLIB into existing ISPF libraries:

- This copying process will have to be repeated every time the factory default instances of the library members are changed by product maintenance.[8]
- You will have to modify an invoking ISPF panel (such as **ISR@PRIM** or **ISRUTIL**) to invoke the **XDCPANEL** panel. (See **Setting Up the XDCPANEL Panel or the XDCLBDEF Clist in ISR@PRIM (or ISRUTIL)** on page 11 for details.)

### Adding Z/XDC's ISPF Element Libraries to TSO Logon Procs

If you elect to add the ***hlq***.XDCV31.XDCMLIB, ***hlq***.XDCV31.XDCPLIB and ***hlq***.XDCV31.XDCTLIB libraries to your ISPF library concatenations:

- You must add a DD card for the Z/XDC libraries to the //ISPMLIB, //ISPPLIB and //ISPTLIB DD concatenations of every logon proc for every TSO user whom you expect to use Z/XDC.[9]
- Alternatively, if you use an ISPF startup clist, you can recode that clist to add the Z/XDC libraries dynamically to the ISPF library concatenations.

[8] The $README process provides guidance for identifying product elements that have changed.

[9] Note, systematic use of **//ddname INCLUDE procname** JCL in your LOGON procs can ease this sort of burden considerably. See **z/OS MVS: JCL Reference** for more information.

- It does not matter where, in the ISPF library concatenations, the Z/XDC libraries occur (first, last, middle). Duplicate instances of the ISPF elements from other sources are not expected.
- When you add a library to a concatenation, remember that the first library in any concatenation either must have the largest BLKSIZE or must specify DCB BLKSIZE=value that is as large as or larger than the library having the largest BLKSIZE. The Z/XDC library BLKSIZEs are all 27,920.
- Modify an invoking ISPF panel (such as **ISR@PRIM** or **ISRUTIL**) to invoke the XDCPANEL panel. (See **Setting Up the XDCPANEL Panel or the XDCLBDEF Clist in ISR@PRIM (or ISRUTIL)** on page 11 for details.)

#### Using XDCLBDEF to Access Z/XDC's ISPF Element Libraries Only When Needed

If you elect to use the XDCLBDEF clist to dynamically allocate the ***hlq***.XDCV31.XDCMLIB, ***hlq***.XDCV31.XDCPLIB, ***hlq***.XDCV31.XDCTLIB and ***hlq***.XDCV31.XDCCLIST libraries, then:

- Read the commentary within XDCLBDEF to understand how it works.
- Change the default values of the MLIB, PLIB, TLIB and CLIST parameters (in the XDCLBDEF clist) to reference the actual dsnames of the hlq.XDCV31.XDCMLIB, ***hlq***.XDCV31.XDCPLIB, ***hlq***.XDCV31.XDCTLIB and ***hlq***.XDCV31.XDCCLIST libraries, respectively.
- You may nullify any of the MLIB, PLIB, TLIB and CLIST parameters by coding an asterisk [*] for its value. Example: CLIST(*) causes the XDCLBDEF clist not to allocate a ***hlq***.XDCV31.XDCCLIST library and not to issue an ALTLIB ACTIVATE command for it.
- Optionally, you may modify an invoking ISPF panel (such as ISR@PRIM or ISRUTIL) to invoke the XDCLBDEF clist instead of the XDCPANEL panel. (See **Setting Up the XDCPANEL Panel or the XDCLBDEF Clist in ISR@PRIM (or ISRUTIL)** on page 11 for details.)
- Alternatively, the user can simply run the XDCLBDEF clist from ISPF's Command Shell (**=6**).

## Setting Up the XDCPANEL Panel or the XDCLBDEF Clist in ISR@PRIM (or ISRUTIL)

The **XDCPANEL** panel provides an interface allowing ISPF users a quick and easy way to invoke Z/XDC to debug their programs. To make it available to your users, you need to choose an invoking panel (such as **ISR@PRIM** or **ISRUTIL**) and then modify it as shown in Figure 4 on page 12. Either way, all roads lead eventually to XDCPANEL.

**XDCPANEL** can be invoked in several ways:

- From another panel (such as ISR@PRIM or ISRUTIL),
- By modifying said panel to invoke XDCPANEL either directly [PANEL(XDCPANEL)] or indirectly [CMD(%XDCLBDEF)],
- Or by requiring your users to run the XDCLBDEF clist from ISPF's Command Shell (=6). See **Choosing Between Invoking XDCPANEL Directly or Indirectly (via XDCLBDEF)** (next) for important considerations.

In order to install XDCPANEL, see Figure 4 below and proceed as follows:

- A line needs to be added to the calling panel's )BODY section to show the user what option code he needs to type to invoke the XDCPANEL panel. "D" is suggested. It stands for "Debugging"[10]
- A line needs to be added to the calling panel's **&ZSEL=TRANS...** command (see Figure 2 below) to invoke either the XDCPANEL panel or the XDCLBDEF clist, depending upon the considerations discussed below.

#### Choosing Between Invoking XDCPANEL Directly or Indirectly (via XDCLBDEF)

If you invoke XDCPANEL directly via "PANEL(XDCPANEL)" (recommended):

[10] "**X**" is not recommended because that would conflict with ISPF's **=X** command.

```
)BODY
 ...
% D +XDC - Interactive Debugging with XDC
 ...


 )PROC
 ...
 &ZSEL = TRANS( TRUNC (&ZCMD,'.')
   ...
    D,'PANEL(XDCPANEL)' ***or*** D,'CMD(%XDCLBDEF)'
```

Figure 2: ISPF Panel Mods for Invoking the XDCPANEL Panel

- Z/XDC's ***hlq***.XDCV31.XDCMLIB, ***hlq***.XDCV31.XDCPLIB, ***hlq***.XDCV31.XDCTLIB and ***hlq***.XDCV31.XDCCLIST libraries must first be copied to or concatenated to your TSO logon PROC's //ISPPLIB, //ISPMLIB, //ISPTLIB and //SYSPROC ddnames, as discussed above in **Adding XDCMLIB, XDCPLIB and XDCTLIB to ISPF** on page 10 and in **Adding XDCCLIST** on page 8.
- Z/XDC's ***hlq***.XDCV31.XDCLINK, ***hlq***.XDCV31.XDCLINKE and ***hlq***.XDCV31.XDCPLPA libraries must be copied to or concatenated to your //STEPLIB, PLPA and/or linklist libraries, as discussed above in **Adding the XDCLINK and XDCLINKE Libraries to Your Linklist** on page 4 and **Adding XDCPLPA Load Modules to Common Storage** on page 3.

If you invoke XDCPANEL indirectly via "CMD(%XDCLBDEF)":

- You need not copy or add the ***hlq***.XDCV31.XDCMLIB, ***hlq***.XDCV31.XDCPLIB, ***hlq***.XDCV31.XDCTLIB and ***hlq***.XDCV31.XDCCLIST libraries to your ISPF libraries. They will be dynamically allocated by XDCLBDEF.
- But you still have to copy ***hlq***.XDCV31.XDCCLIST(XDCLBDEF) library to a //SYSPROC ddname in your logon PROCs.
- For non-authorized debugging, you need not add the ***hlq***.XDCV31.XDCLINK, ***hlq***.XDCV31.XDCLINKE and ***hlq***.XDCV31.XDCPLPA libraries to your linklist, PLPA or TSO //STEPLIB libraries. Instead, you can name those libraries in XDCLBDEF's LLIB operand, and they will be dynamically allocated as ISPLLIB type libraries.
- However, for authorized debugging, naming the ***hlq***.XDCV31.XDCLINK, ***hlq***.XDCV31.XDCLINKE and ***hlq***.XDCV31.XDCPL libraries via the LLIB operand will not work! This is because ISPF does not support authorized libraries via its ISPLLIB facility. Consequently, you will still have to copy or concatenate the ***hlq***.XDCV31.XDCLINK, ***hlq***.XDCV31.XDCLINKE and ***hlq***.XDCV31.XDCPLPA libraries into your //STEPLIB, PLPA and/or linklist libraries (as discussed in **Adding the XDCLINK and XDCLINKE Libraries to Your Linklist** on page 4 and **Adding XDCPLPA Load Modules to Common Storage** on page 3).

Users can also run the XDCLBDEF clist from ISPF's Command Shell (**=6**).

### Adding the XDCCMDS Scripts Library

The ***hlq***.XDCV31.XDCCMDS library is a partitioned dataset containing sample scripts of Z/XDC commands. End users can use Z/XDC's READ command to execute these scripts for various purposes. Accordingly, when you make Z/XDC available to your users, please also publicize this library so that they will be able to use it.

Also, when defining Z/XDC to your security system (see **Step 4: Defining Z/XDC to Your Computer's Security System** on page 14), be sure to give your users READ access to this library.

## Step 2: ReIPL (Maybe) - A Checklist

If you have done everything "right", then you should not have to reIPL. Please review the following checklist.

- When you added the hlq.XDCV31.XDCPLPA library (or its contents) to the LPA-list, did you also issue the following command?

  **SETPROG LPA,ADD,MOD=(XDC,XDC31),DSN=dsname**

  If not, then do it now or reIPL.

- If you added the hlq.XDCV31.XDCLINK and hlq.XDCV31.XDCLINKE libraries to the linklist, did you also issue the following commands?

  **SETPROG APF,ADD,DSN=hlq.XDCV31.XDCLINK,VOL=volser**
  **SETPROG APF,ADD,DSN=hlq.XDCV31.XDCLINKE,VOL=volser**
  **SETPROG LNKLST,DEFINE,NAME=TEMP,COPYFROM=CURRENT,NOCHECK**
  **SETPROG LNKLST,ADD,NAME=TEMP,DSN=hlq.XDCV31.XDCLINKE,ATTOP**
  **SETPROG LNKLST,ADD,NAME=TEMP,DSN=hlq.XDCV31.XDCLINK,ATTOP**
  **SETPROG LNKLST,ACTIVATE,NAME=TEMP**
  **F LLA,REFRESH**

  If not, then do it now or reIPL

- If you (instead) copied Z/XDC load modules into an existing linklist library, did you also issue the following commands?

  **F LLA,REFRESH**

  If not, then do it now.

- If copying Z/XDC load modules into an existing linklist library caused that library to create and expand into extra extents, did you also issue the following commands?

  **SETPROG LNKLST,DEFINE,NAME=TEMP,COPYFROM=CURRENT,NOCHECK**
  **SETPROG LNKLST,ACTIVATE,NAME=TEMP**

  If not, then do it now or reIPL.

- If you added the hlq.XDCV31.XDCLINK and/or ditto.XDCLINKE and/or ditto.XDCPLPA library to the APF section of a PROGxx member of a PARMLIB library, did you also issue the following commands?

  **SETPROG APF,ADD,DSN=hlq.XDCV31.XDCLINK,VOL=volser**
  **SETPROG APF,ADD,DSN=hlq.XDCV31.XDCLINKE,VOL=volser**
  **SETPROG APF,ADD,DSN=hlq.XDCV31.XDCPLPA,VOL=volser**

  If not, then do it now or reIPL.

- If you added the XDCCALLA and XDCCMDA names to the AUTHCMD list of an active IKJTSOxx member of the PARMLIB libraries, did you also issue the following TSO command?

  **PARMLIB UPDATE(xx)**

  If not, then do it now or reIPL.

Ok, so maybe you don't need to reIPL after all.

## Step 3: Defining Your License to Use Z/XDC

Before you can use Z/XDC, you must define to it your license to use it on your current system.

The License Definition check is invoked by Z/XDC automatically whenever a new debugging session is started. So nobody will be able to use Z/XDC until the license is applied.

For example, if you attempt to start server/XDC before applying the license, it will fail in miserable and ugly ways. So you need to apply the license first, before you try starting the **XDCSRVER** proc!

There is an interactive way to apply Z/XDC's license, but probably it is easier just to run a SUPERZAP job that you will have received from us. So do that now. I'll wait...

Done? Good. Let's continue...

If you are installing multiple copies of Z/XDC into multiple z/OS images, then after performing the licensing procedure for the first copy, you can IEBCOPY the modified XDCLCNSE load module over to the other copies of Z/XDC. This will work so long as all copies require the same Licensing Control Data.

Once you have successfully defined your license to Z/XDC, you will be able to run debugging sessions.

If at some future time you need to display the License Definition Panel, you can do so by starting a normal Z/XDC debugging session (such as "TSO XDCCALL IEFBR14") and then issuing Z/XDC's LICENSE command.

## Step 4: Defining Z/XDC to Your Computer's Security System

Z/XDC is a generalized debugging tool that can be a powerful aid to any assembler language programmer.

Z/XDC, in and of itself, is not "authorized"; accordingly, Z/XDC both can and should be made available to a computer center's general TSO user population. In a secured computer center, a general user cannot violate system security and integrity with Z/XDC.

There are, however, certain capabilities that an installation may wish to restrict the use of to only certain trusted individuals. Here is a partial list:

- When a debugging session is started via the XDCCALLA or XDCCMDA utility, or when a Z/XDC interface is installed into an authorized program[11], Z/XDC will receive control in an authorized mode. When this happens, the user of Z/XDC can (security permitting) zap arbitrary system storage. Again, when a debugging session is started via the XDCCALLA or XDCCMDA utility and a //TASKLIB[12] DD allocation exists containing one or more non-APF libraries, XDCCALLA/CMDA will take steps to make those libraries locally and temporarily APF authorized for use during the debugging session.[13]
- Whenever Z/XDC discovers it is running authorized, permission to continue is controlled by an [XDC.][14] AUTH security rule. This rule provides a final control over who is and is not given the power to break the System.
- When Z/XDC is running authorized, the user may be permitted to use Z/XDC commands (such as SET ASID or HOOK) to gain access to other address spaces. This can be controlled by the [XDC.]FASM.*jobname* security rules.
- When a user has such access to a foreign address space, Z/XDC may permit him to display and possibly to zap any data in that address space. Zapping can be controlled by the "[XDC.]ZAP.*description*" rules. [15]
- When a user logs onto Z/XDC's Cross Domain Facility (cdf/XDC), system security is called to verify that user's identity and again to determine which background jobs or system tasks he is permitted to connect to for debugging. This can be controlled by the [XDC.]FASM.*jobname* security rules.

[11] Note, in a properly secured computer center, only appropriate personnel will have the capability of installing a Z/XDC interface into an authorized program.

[12] By default, the XDCCALL/CMD[A] utilities will set up ddname TASKLIB as the debugging session's task library. There is, however, a syntax by which the user can choose a different ddname to be use for the task library.

[13] Outside of the debugging session and to the rest of the world, the affected libraries remain non-APF authorized.

[14] For RACF protected systems, the names of all Z/XDC's security rules start with **XDC.** followed by the rest or the rule name. But for ACF2 and TSS protected systems, the rule names do not include the leading **XDC.**.

[15] Where "description" more specifically describes the storage to be altered.

If you need to control these and other kinds of accesses, then you will need to define certain Z/XDC access control rules to your security system.

### For More Information About Z/XDC Security ...

The following pages describe the kinds of security services that can be set up for Z/XDC users and how to establish those services.

There also is available in Z/XDC's Built-In Help a comprehensive discussion of general security issues and questions raised by a product such as Z/XDC. Your Security Officer needs to read that discussion. To do so, he can either

- Read **Z/XDC v3.1 User Guide**
- Or start a debugging session (TSO XDCCALL IEFBR14) and then use Z/XDC's HELP SECURITY command to read the security related topics directly from Z/XDC's Built-in Help.
- Or use a browser to access colesoft.com/help (Z/XDC's **Help Online**), and then use the left-side navigation pane to navigate to the Security topic.

  HELP SECURITY has numerous subtopics describing in detail each of the security rules that Z/XDC checks and the circumstances under which it makes those checks. To see an index of these subtopics:
- If reading the **User Guide**, the **Table of Contents** will show all the subtopic names.
- If accessing the Built-In Help within Z/XDC, use LIST HELP SECURITY 2 to see a report of the subtopic names.
- If accessing the Help Online, the left-side navigation pane will show you the subtopic names.

### Z/XDC's Standard Security Method

Z/XDC issues calls to the installed security system (RACF, ACF2 or TSS) via IBM's "SAF interface"[16]. Z/XDC makes such calls whenever the user first attempts to use a Z/XDC command that may give rise to a security concern. Specifically, Z/XDC calls security under the following circumstances:

Upon the first use of Z/XDC within a debugging session in an authorized mode, Z/XDC checks for READ access to a Z/XDC defined resource named "[XDC.]AUTH" for permission to allow that. See Help topic named HELP SECURITY AUTH for details.

Upon the first time that a SET ASID command or an addressing function [such as ASID(...)] is used to gain read access to another address space, if that address space does not have an ownerid[17] that matches the ownerid of the address space from which the SET ASID command is being issued, then Z/XDC checks for READ access to a resource named "[XDC.]FASM.*jobname*" (where "*jobname*" is the name of the address space being accessed). This check is made for each different address space so accessed. See Help topic named HELP SECURITY FASM for details.

Upon the first time a storage altering command (ZAP, etc.) is used to alter the private area of another address space (and if that address space has a different ownerid than the home space), Z/XDC checks for UPDATE access to the "[XDC.]FASM.*jobname*" rule. Again, see HELP SECURITY FASM for details.

Any time a storage altering command is used to alter any storage anywhere, Z/XDC checks for UPDATE access to a rule named "[XDC.]ZAP.*description*" where "*description*" more specifically describes the storage to be altered. Note, this check will be made even for zaps that also require the check for UPDATE access to the "[XDC.]FASM.jobname" resource. See Help topic named HELP SECURITY ZAP for details.

[16] The "System Authorization Facility" (the "SAF interface") is a standard facility in all z/OS systems that provides a common interface to whatever security system is installed: RACF, ACF2 or Top Secret (TSS).

[17] Z/XDC finds an address space's "ownerid" by looking in the ACEEUSRI field of that address space's ACEE control block. If the ACEE does not exist, or if the ACEEUSRI field contains blanks or an asterisk, then Z/XDC concludes that the address space does not have an ownerid.

When a user logos on to a debugging session wither from a 3270 terminal or from a browser [planned], the logon process calls security to verify that user's TSO id and password.[18]

When a user who has signed on user selects a background job or task to debug, if that job's ownerid does not match the user's userid, then cdf/XDC calls security to see if that user is authorized to debug the selected job or task. It checks for UPDATE access to the appropriate "[XDC.]FASM.*jobname*" rule. Again, see Help topic named HELP SECURITY FASM for details.

### Using Z/XDC Prior to Creating Security Definitions

Z/XDC's security interface explicitly supports three security systems: RACF, ACF2 and Top Secret. This support requires various implementing definitions be made within the security systems as described in the following sections.

Until such definitions are made, the RACF and Top Secret security systems report the various Z/XDC resources as being "unprotected" (RC=4 from the RACROUTE macro), and for the most part, Z/XDC treats such responses as being equivalent to a "permit" response (RC=0)[19]. Accordingly, in RACF and Top Secret shops, Z/XDC can be installed and testing during a trial period without having to set up security definitions. But security concerned installations should make the necessary definitions prior to Z/XDC's being released to the user community for general use.

ACF2 is an exception. It "denies" (RC=8 from the RACROUTE macro) accesses to unprotected resources[20]; therefore, in ACF2 shops suitable security definitions for Z/XDC will have to be made before Z/XDC can be used at all.

[18] When a user signs onto cdf/XDC from a TSO session, no password check is necessary because the user's id and password have already been verified when the user first signed onto TSO.

[19] There is one exception. If you wish to test using Z/XDC as an FRR, then there is a security rule that Z/XDC calls for which a "not protected" response is treated as a "deny" response. For more information, see the HELP SECURITY LOSTLOCKS topic in either the **Z/XDC User Guide** or the Built-in Help or Help Online.

[20] ACF2 never makes an RC=4 return to a RACROUTE caller. Whenever a **not protected** situation arises, ACF2 makes an RC=8 (**denied**) return instead.

### Using Z/XDC in RACF Protected Systems

Figure 3 below shows the RACROUTE macro operands used by Z/XDC in RACF protected systems.

Important! Although this is not recommended, RACF shops do not need to configure their security rules prior to installing Z/XDC. When Z/XDC requests a ruling, and a pertaining rule does not exist, Z/XDC will proceed as if the ruling had been to allow the access. However, the safest thing of course is to get your rules set up ahead of time, and this topic discusses how to do that.

Note that Z/XDC uses the resource class named "FACILITY". This is an IBM defined class which many software vendors use as a convenient place in which to create their rules.[21] Z/XDC's use of this class avoids the necessity of requiring the customer to create or modify a "user" Class Descriptor Table.

For RACF protected systems, Z/XDC issues the RACROUTE macro with "REQUEST=AUTH", but with REQSTOR and SUBSYS not set. Example: An attempt by a user to access JES2's address space would cause Z/XDC to issue the following RACROUTE macro:

```
          RACROUTE REQUEST=AUTH,ATTR=READ,CLASS=CLASS-1,ENTITY=RESOURCE

          DC    AL1(L'CLASS)
CLASS     DC    C'FACILITY'
RESOURCE  DC    CL39'XDC.FASM.JES2'
```

| Call Purpuose | ATTR= | ENTITY Value[a] | Resource Class |
|---|---|---|---|
| Allow authorized mode | READ | CL39'XDC.AUTH' | FACILITY |
| Access a foreign address space | READ | CL39'XDC.FASM.*jobname*'[b] | FACILITY |
| Zap a foreign address space[c] | UPDATE | CL39'XDC.FASM.*jobname*'[b] | FACILITY |
| Connect to a debugging session started in a job or system task | UPDATE | CL39'XDC.FASM.*jobname*'[b] | FACILITY |
| Zap storage[c] | UPDATE | CL39'XDC.ZAP.*description*'[b] | FACILITY |
| Debug locked/disabled code[d] | READ | CL39'XDC.LOSTLOCKS' | FACILITY |

[a] All entity values start with "XDC" regardless of Z/XDC's name. Thus, the same rules that govern XDC proper also govern all clones of XDC. (See **"Step 10: Creating a Renamed Clone**" on page 35 for more information.)

[b] "*Jobname*" is replaced by the 8-character name of the address space that the user is trying to access.

[c] Zaps that require permission under the "XDC.FASM.*jobname*" rules, still also require permission under an appropriate "XDC.ZAP.*description*" rule.

[d] Z/XDC has limited support for debugging locked or disabled code. see HELP SECURITY LOSTLOCKS for details.

Figure 3: Z/XDC's RACROUTE Macro Usage in RACF Protected Systems

### RACF - Initial Setup: Deny All Accesses by Everyone

To implement Z/XDC security, a security officer should do the following:

[21] This hitchhiking usage does not interfere with the class's original purpose of controlling users of DFSMShsm commands.

Start by using RACF's RDEFINE commands to create, in the FACILITY class, background settings prohibiting access to all of Z/XDC's authorized facilities for all users:

**RDEFINE FACILITY XDC.APF.* UACC(NONE)**
**RDEFINE FACILITY XDC.AUTH UACC(NONE)**
**RDEFINE FACILITY XDC.ZAP.* UACC(NONE)**
**RDEFINE FACILITY XDC.FASM.* UACC(NONE)**
**RDEFINE FACILITY XDC.LOSTLOCKS UACC(NONE)**
**SETROPTS REFRESH RACLIST(FACILITY)**

Use the characters "XDC" in these definitions regardless of whether or not you have renamed Z/XDC. (See **Step 10: Creating a Renamed Clone** on page 35 for more information about creating/using clones of XDC.)

### RACF - Permitting Some Accesses to Some People

Use RACF's PERMIT command to give the XDCSRVER job READ access to the XDC.AUTH rule. (This is required.) Example:

**PERMIT XDC.AUTH CLASS(FACILITY) ID(ownerid[22]) ACCESS(READ)**

Use RACF's PERMIT commands to grant individual users and user groups access to Z/XDC's various authorized facilities.

### RACF - Permitting Accesses: Examples

The following examples presume the background defaults recommended above.

**PERMIT XDC.AUTH CLASS(FACILITY) ID(DBCOLE) ACCESS(READ)**
Allows DBCOLE to debug authorized programs.

**PERMIT XDC.FASM.* CLASS(FACILITY) ID(DBCOLE) ACCESS(READ)**
Allows DBCOLE to use Z/XDC's Foreign Address Space Mode to display (but not to zap) any address space in the system.

**PERMIT XDC.FASM.JES2 CLASS(FACILITY) ID(DBCOLE) ACCESS(UPDATE)**
Allows DBCOLE to zap storage in JES2's address space.

**PERMIT XDC.ZAP.* CLASS(FACILITY) ID(DBCOLE) ACCESS(UPDATE)**
Allows DBCOLE to step through program execution in his own address space and any address space that the XDC.FASM rules allow UPDATE access to. This also allows DBCOLE to zap any storage in the aforementioned address spaces.

**PERMIT XDC.LOSTLOCKS CLASS(FACILITY) ID(DBCOLE) ACCESS(READ)**
Allows DBCOLE to debug locked or disabled code.

**SETROPTS RACLIST(FACILITY) REFRESH**
**RLIST FACILITY ***
These commands harden the changes and display the results.

[22] "*Ownerid*" is the RACF ownerid under which the **XDCSRVER** job will run.

### RACF - Additional Information

For additional information about RACF, please refer to its **Security Administrator's Guide** and its **Command Language Refe**

Also, for a comprehensive discussion about creating the XDC.AUTH, XDC.ZAP.*description* and XDC.FASM.*jobname* rules and about how Z/XDC uses them, see HELP SECURITY and its subtopics.

### Using Z/XDC in ACF2 Protected Systems

Figure 6 below shows the RACROUTE macro operands used by Z/XDC in ACF2 protected systems. Important! ACF2 shops must configure their security rules prior to trying to use Z/XDC. This is because when security rules are absent, ACF2's ruling will always deny the access. So when Z/XDC's rules do not exist, Z/XDC will not start.

ACF2 Example: An attempt by a user to access JES2's address space would cause Z/XDC to issue the following RACROUTE macro:

```
          RACROUTE REQUEST=AUTH,ATTR=READ,REQSTOR[a]=XDC,SUBSYS[a]=XDC,
                CLASS[a]=CLASS-1,ENTITY=RESOURCE

          DC AL1(L'CLASS)
CLASS     DC    C'XDC'
RESOURCE  DC    CL44'FASM.JES2'
XDC       DC    CL8'XDC'
```

| Call Purpuose | ATTR= | ENTITY Value | Resource Class |
|---|---|---|---|
| Allow authorized mode | READ | CL44'XDC.AUTH' | XDC |
| Access a foreign address space | READ | CL44'FASM.*jobname*'[b] | XDC |
| Zap a foreign address space[c] | UPDATE | CL44'FASM.*jobname*'[b] | XDC |
| Connect to a debugging session started in a job or system task | UPDATE | CL44'FASM.*jobname*'[b] | XDC |
| Zap storage[c] | UPDATE | CL44'ZAP.*description*'[b] | XDC |
| Debug locked/disabled code[d] | READ | CL44'LOSTLOCKS' | XDC |

[a] The CLASS, REQSTOR and SUBSYS value of "XDC" must be used for security calls regardless of whether or not you have renamed Z/XDC. Thus, the same rules that govern XDC proper also govern all clones of XDC. (See **"Step 10: Creating a Renamed Clone**" on page 35 for more information.)

[b] "*Jobname*" is replaced by the 8-character name of the address space that the user is trying to access.

[c] Zaps that require permission under the "XDC.FASM.*jobname*" rules, still also require permission under an appropriate "XDC.ZAP.*description*" rule.

[d] Z/XDC has limited support for debugging locked or disabled code. see HELP SECURITY LOSTLOCKS for details.

Figure 4: Z/XDC's RACROUTE usage in ACF2 Protected Systems

### ACF2 - Initial Setup

To implement Z/XDC security, several initial definitions need to be made within ACF2[23]:

[23] The name of the class, the generalized resource rule type, and the INFODIR entry all must be "XDC" regardless of whether or not Z/XDC has been renamed (as discussed in **Step 10: Creating a Renamed Clone** on page 35).

- A GSO CLASMAP.XDC record is optional. This is because the resource name (**XDC**) is three characters long, so ACF2 will automatically use the name as the resource type. If you were to define a CLASMAP record, the commands would be:
  - **SET CONTROL(GSO)**
  - **INSERT CLASMAP.XDC RESOURCE(XDC) RSRCTYPE(XDC)**
- An R-XDC entry needs to be added to the INFODIR control record. The command would be:
  - **INFODIR TYPES(R-XDC)**
- Suitable "generalized resource rules" with TYPE(XDC) need to be created.
- An XDCSRVER logonid needs to be created for the Cross Domain Facility. This id must be assigned at least the following attributes: MUSASS, STC and RESTRICT.

More specifically, a security officer needs to modify the ACF2 control records as follows:

From TSO, type ACF, then issue the following commands:

```
SET CONTROL(GSO)
CHANGE INFODIR TYPES(D-RXDC)
INSERT CLASMAP.XDC RESOURCE(XDC) RSRCTYPE(XDC) ENTITYLN(13) INSERT
       ID(XDC) MODE(GLOBAL) RACROUTE(REQUEST=AUTH,CLASS=XDC)
```

You will probably want "to mask" (i.e., use wildcard characters in) some of the Z/XDC resource rule definitions you will be creating. Accordingly, a "directory" for the Z/XDC rules will have to be made resident in system storage. To do so, start by listing the INFODIR control record. Then add D-XDC to the TYPES field.

Finally, from an operator's console issue the ACF2 commands necessary to "refresh" the control records you have modified (CLASMAP and INFODIR). This will activate the changes you have made.

**ACF2 - Permitting Accesses**

Once "XDC" has been defined to ACF2, it is now necessary to define the actual generalized resource rules that make up the "XDC" class. An ACF2 security officer should do this as follows:

- From TSO, type "ACF", then type "SET RESOURCE(XDC)".
- Use ACF2's COMPILE and other commands to create the desired access control rules. The rules should all specify "TYPE(XDC)".
- The "$KEY" values for the various rules should be based on the following:

**$KEY(AUTH) TYPE(XDC)**
**UID(*uidstring*) SERVICE(READ) ALLOW**
This controls who can use Z/XDC to debug programs running APF authorized or running in supervisor state or running with a system key.

**$KEY(ZAP) TYPE(XDC)**
description UID(uidstring) SERVICE(READ) ALLOW This controls who can zap what storage. See the HELP SECURITY ZAP topics for details concerning "*description*".

**$KEY(FASM) TYPE(XDC)**
jobname UID(uidstring) SERVICE(READ) ALLOW This controls who can have READ access to the foreign address space named "jobname".

**$KEY(FASM) TYPE(XDC)**
***jobname* UID(*uidstring*) SERVICE(UPDATE) ALLOW**
This controls who can have ZAP access and debugging session access to the foreign address space named "*jobname*".

**$KEY(LOSTLOCKS) TYPE(XDC)**
**UID(*uidstring*) SERVICE(READ) ALLOW**
This controls who is allowed to debug locked code.

ACF2 permits $KEY values to contain "wildcard" characters (asterisks), thus it is possible, for example, to create one "FASM.*jobname*" rule to control accesses to multiple address spaces. (ACF2 refers to the use of wildcards as "masking"). Remember however, if such masked keys are to be used, then the "directory" for the "XDC" rules must be made "resident" in storage. This is done via the INFODIR control record (see above).

ACF2 has a "Restricted Commands List" that explicitly lists the commands (both authorized and non-authorized) that a user is allowed to use within TSO. If present, then that List needs to have entries added to it for XDCCALL, XDCCMD, XDCCALLA, XDCCMDA, XDCTFS and XDC.

Finally, in order to run the server task (server/XDC - See **Step 7: Installing Z/XDC's Server Job** on page 25.), you will need to create a logonid for it with at least the following attributes: MUSASS, STC and RESTRICT. The name of this logonid should be "XDCSRVER", and that name must match the name of the proc used to start server/XDC. You also must give the XDCSRVER logonid READ access to the AUTH rule. The following commands will accomplish the above:

- **SET LID**
- **INSERT XDCSRVER MUSASS STC RESTRICT**
- **SET R(XDC)**
- **RECKEY AUTH ADD( UID(*uidstringforxdcsrver*) SERVICE(READ) ALLOW)**

See ACF2's Administrator's Guide for more information about defining generalized resource rules, about logonids and about modifying the control records. Also, ***hlq***.XDCV31.XDCSAMP(ACF2SAMP) contains several examples of various Z/XDC type generalized resource rules.

Also, for a comprehensive discussion about creating the AUTH, ZAP.*description* and FASM.*jobname* rules and about how Z/XDC uses them, see the HELP SECURITY subtopics.

### Using Z/XDC in Top Secret Protected Systems (TSS)

Figure 5 below shows the RACROUTE macro operands used by Z/XDC in TSS protected systems.

Important! Although this is not recommended, TSS shops do not need to configure their security rules prior to installing Z/XDC. When Z/XDC requests a ruling, and a pertaining rule does not exist, Z/XDC will proceed as if the ruling had been to allow the access. However, the safest thing of course is to get your rules set up ahead of time, and this topic discusses how to do that.

ACF2 Example: An attempt by a user to access JES2's address space would cause Z/XDC to issue the following RACROUTE macro:

```
          RACROUTE REQUEST=AUTH,ATTR=READ,CLASS=CLASS-1,ENTITY=RESOURCE

          DC AL1(L'CLASS)
CLASS     DC    C'XDC'
RESOURCE  DC    CL44'FASM.JES2'
```

| Call Purpuose | ATTR= | ENTITY Value | Resource Class[b] |
|---|---|---|---|
| Allow authorized mode | READ | CL44'XDC.AUTH' | XDC |
| Access a foreign address space | READ | CL44'FASM.*jobname*'[b] | XDC |
| Zap a foreign address space[c] | UPDATE | CL44'FASM.*jobname*'[b] | XDC |
| Connect to a debugging session started in a job or system task | UPDATE | CL44'FASM.*jobname*'[b] | XDC |
| Zap storage[c] | UPDATE | CL44'ZAP.*description*'[b] | XDC |
| Debug locked/disabled code[d] | READ | CL44'LOSTLOCKS' | XDC |

[a] The Resource Class named "XDC" is used for security calls regardless of whether or not you have renamed Z/XDC. Thus, the same rules that govern XDC proper also govern all clones of XDC. (See **"Step 10: Creating a Renamed Clone**" on page 35 for more information.)

[b] "*Jobname*" is replaced by the 8-character name of the address space that the user is trying to access.

[c] Zaps that require permission under the "XDC.FASM.*jobname*" rules, still also require permission under an appropriate "XDC.ZAP.*description*" rule.

[d] Z/XDC has limited support for debugging locked or disabled code. see HELP SECURITY LOSTLOCKS for details.

Figure 5: Z/XDC's RACROUTE usage in Top Secret Protected Systems

### TSS - Initial Setup

To implement Z/XDC security, several initial definitions need to be made within TSS:

Z/XDC uses the resource class named "XDC" regardless of Z/XDC's actual name. Accordingly, a "system facility", an "ACID" ("access id") and a "resource" all must be defined in TSS as follows:

Start by renaming a spare facility to "XDCCDF". Then assign it the attributes necessary to permit Z/XDC's Cross Domain Facility to operate as a multi-user subsystem. (In order to make these changes permanent, you will have to update TSS's startup parms):

```
TSS MODIFY FAC(USERx=NAME=XDCCDF)
TSS MODIFY FAC(XDCCDF=PGM=XDCSERVE,NOABEND,MULTIUSER,NONPWR,NOTSOC)
```

Create an ACID ("access id") named "XDCCDF". This will be the ACID assigned to the Cross Domain Facility's startup proc:

**TSS CREATE(XDCCDF) NAME('name') DEPT(*deptacid*) PROTECTED FAC(STC,TSO) MASTFAC(XDCCDF)**

Assign XDCSRVER's startup proc to use the XDCCDF ACID:

**TSS ADD(STC) PROC(XDCSRVER) ACID(XDCCDF)**

Allow users to logon to XDCCDF:

**TSS ADD(userid) FAC(XDCCDF)**

Define a resource named "XDC" in the RDT ("Resource Definition Table"), and set its default access to NONE. "RESCODE" may specify any previously unused 2-digit code:

**TSS ADD(RDT) RESCLASS(XDC) RESCODE(*hexcode*) ATTR(DEFPROT,LONG)**
**ACLST(UPDATE,READ,NONE) DEFACC(NONE)27**

Assign ownership to the various entity values that Z/XDC uses in its security checks:

**TSS ADD(*ownerid*) XDC(APF.)**
**TSS ADD(*ownerid*) XDC(AUTH)**
**TSS ADD(*ownerid*) XDC(FASM.)**
**TSS ADD(*ownerid*) XDC(LOSTLOCKS)**
**TSS ADD(*ownerid*) XDC(ZAP.)**

**TSS - Permitting Some Accesses to Some People**

Allow users to access the various resources secured by Z/XDC. Example: The following commands permit user DBCOLE to use Z/XDC's Foreign Address Space Mode to have both display, alter, and cdf/XDC access to JES2's private areas:

**TSS PERMIT(DBCOLE) XDC(FASM.JES2) ACCESS(READ)**
**TSS PERMIT(DBCOLE) XDC(FASM.JES2) ACCESS(UPDATE)**

Note that with Top Secret (unlike other security systems) UPDATE access does not include or imply READ access. To permit both accesses, both accesses must be explicitly defined.

**TSS - Permitting Accesses: Examples**

The following examples presume the default access to all rules is NONE.

**TSS PERMIT(DBCOLE) XDC(AUTH) ACCESS(READ)** Allows DBCOLE to debug authorized programs.

**TSS PERMIT(DBCOLE) XDC(FASM.) ACCESS(READ)** Allows DBCOLE to use Z/XDC's Foreign Address Space Mode to display (but not to zap) any address space in the system.

**TSS PERMIT(DBCOLE) XDC(FASM.JES2) ACCESS(READ UPDATE)** Allows DBCOLE to zap storage in JES2's address space.

**TSS PERMIT(DBCOLE) XDC(ZAP.) ACCESS(UPDATE)** Allows DBCOLE to trace execution in (and to zap any storage in) his own address space and any address space that the XDC.FASM rules allow UPDATE access to.

**PERMIT XDC.LOSTLOCKS CLASS(FACILITY) ACCESS(READ)** Allows DBCOLE to debug locked or disabled code.

**TSS - Additional Information**

For additional information, please refer to TSS's MVS Control Options Guide, MVSImplementation, Concepts and Facilities and TSS Command Functions Guide.

Also, for a comprehensive discussion about creating the AUTH, ZAP.*description* and FASM.*jobname* rules and about how Z/XDC uses them, see the HELP SECURITY topic in Z/XDC v3.1 User Guide or in Z/XDC's Built-in Help.

## Step 5: c/XDC Requires an OMVS Segment

c/XDC processing requires Unix System Services (USS, also referred to as OMVS); therefore...

- The XDC Server Space (XDCSRVER) must be started under a userid that has a OMVS segment defined in it's RACF profile,
- And any users that uses c/XDC must also have an OMVS segment defined.

The OMVS segment can be manually defined for each user or you can create a default OMVS Segment that is automatically propagated on an as needed basis. Any TSO user or batch job that does not have an OMVS segment will be assigned a unique UnixID# and z/OS Unix (OMVS/USS) Segment the first time the user uses the c/XDC Licensed feature.

**Example:** If you already know z/OS Unix and are comfortable with the whole concept of **z/OS Unix Segments**, then just skip this example. You already know what you need to do.

On the other hand, if you (like me) are comfortable with TSO but don't know Unix from Tunix, then here are the bare bones things you need to do to create a model OMVS segment and then get on with your life:

First, create a model RACF username (**OEMODEL** in this example) and its OMVS Segment:

```
AU oemodel OMVS(HOME('/u/') PROGRAM('/bin/sh'))
```

Next, create a RACF rule with the magic name: **BPX.UNIQUE.USER**:

```
RDELETE FACILITY BPX.UNIQUE.USER
RDEFINE FACILITY BPX.UNIQUE.USER APPLDATA('oemodel') UACC(READ) OWNER(SYS1)
SETR REFRESH RACLIST(FACILITY)
```

This notifies RACF to enable default OMVS processing.

Done. You can now move on.

## Step 6: Installing Z/XDC's Service SVC, Legacy Hook SVC and System Interface

In order to make the full range of Z/XDC's facilities available for use, two special SVCs and several other system interface routines must be installed for Z/XDC. These SVCs and interfaces provide various services that help improve the power of Z/XDC.

The routines do **not**, however, in any way enable Z/XDC to run authorized. (Z/XDC will run authorized only when it is invoked by a program that itself is already authorized). Considerable care has been exercised in the design of Z/XDC's SVCs and system interfaces. They cannot be misused by **un**authorized callers to subvert system security or integrity.

Z/XDC's System Interface need not, and in fact cannot, be installed by normal methods. Instead, it must be dynamically installed by the server/XDC job. (See **Step 7: Installing Z/XDC's Server Job** on page 25.)

server/XDC must be run after every IPL. It also must be bounced every time Z/XDC is updated by maintenance.

During the System Interface Installation process, Z/XDC does the following:

- It installs (or maybe reinstalls, if necessary) its hook SVC and its service SVC.
- The SVC numbers usually will be 199 and 198, respectively, but that depends upon whether or not those slots in the system's SVC table are empty. If they are not, then other numbers will be chosen.
- The installation process builds for Z/XDC a subsystem control table (SSCT) named "XDC*c*" and uses it, among other things, as an anchor for Z/XDC's System Interface. It also is a place to save Z/XDC's SVC numbers.
- Note, the "*c*" in "XDCc" is a release specific special character. For release v3.1, "*c*" is a period ("."), so v3.1's SSCT name is "XDC.". Notes:
- This intentionally makes it impossible for Z/XDC's SSCTs to ever conflict with SSCTs that are installed by z/OS, system products and other vendor products.
- It also makes it possible for Z/XDC itself to recognize all its SSCTs, regardless of clone name (see next bullet).
- When Z/XDC is installed as a renamed clone, the **XDC** part of the SSCT's name will be changed to match the clone's name. This allows multiple XDCs to have wholly independent System Interfaces. See **Step 10: Creating a Renamed Clone** on page 35 for more information.
- The installation process also installs various other system routines to support:
- Deferred breakpoints (see the HELP BREAKPOINTS DEFERRED topic in **Z/XDC v3.1 User Guide** or in Z/XDC's Built-in Help)
- Dynamic hooks (HELP HOOKS DYNAMIC)
- And certain end-of-task and end-of-memory cleanup functions.

When server/XDC completes the(re)installation of Z/XDC's System Interface, a detailed System Interface Initialization Report (messages DBC514I) is sent to SYSLOG showing exactly what the (re)installation process did and did not do, and why.

#### About Z/XDC's SVCs

In the case of the two SVCs, Z/XDC automatically selects two available SVC numbers (199 and 198 or lower [24]), and it assigns them to the SVCs. Usually:

- The legacy hook SVC will be assigned as SVC 199,
- And the service SVC will be assigned as SVC 198.
- **But this is not guaranteed!** If another product has already taken either 199 or 198, then Z/XDC will automatically select different numbers. This is rare, but it does happen.
- Applying Z/XDC maintenance may also change the SVC numbers.

## Step 7: Installing Z/XDC's Server Job

server/XDC provides several global services that are required for normal Z/XDC operation. These include:

- The ability (known as cdf/XDC) to connect debugging sessions to batch jobs
- The ability to set hooks for creating on-the-fly debugging sessions
- The ability to connect a debugging session across a sysplex to a job running on another system
- The ability to use c/XDC to debug C, C++, Metal C and other C languages.[25]

[24] SVC numbers lower than 200 are chosen so as not to interfere with other user SVCs.
[25] c/XDC supports debugging a wide variety of C languages, including XL C and C++, SPC C, Metal C, Dignus Systems/C and C++, XL-clang C and C++ and others.

The various capabilities within server/XDC are known as **subservers**.

### Bouncing server/XDC - Is it Safe?

Yes. It is safe for two reasons:

- Whenever server/XDC restarts, it does a thorough rebuild of it runtime environment. So if the prior run was stopped, was cancelled, or simply failed, the restart will always succeed.
- When customers are engaged in active debugging sessions, they are connected directly to those sessions, not to the server. So a server/XDC bounce won't ever be noticed by the users.

*However* issues do arise when the server is bounced during maintenance update runs, which are described in detail by a $README file that is provided whenever you download new maintenance from colesoft.com. See **The $README File** on page 1 for more information about that.

The server job needs to be up and running for the life of the IPL. It may be bounced without undue consequences. Specifically, when server/XDC is down:

- New batch debugging sessions cannot be created.
- But existing batch debugging sessions will not be affected.
- The HOOK command and the MAP command for C programs will be unavailable.

So such bounces should be as brief as reasonable. In any case, all services will be resumed when server/XDC is brought back up.

When server/XDC starts or restarts, it always builds or rebuilds its infrastructure from scratch. So when bringing it down, it really doesn't matter much whether you do so via a STOP (P) command or a CANCEL (C) command.

If for some reason server/XDC is hosed, a STOP may not work, a CANCEL will usually work and a FORCE will always work. In all cases, the restart will go smoothly.

### Setting up cdf/XDC

cdf/XDC runs as a subserver of server/XDC. Its purpose is to connect users to debugging sessions that may occur when a batch job or started task (or sometimes even a TSO session) has abended and passed control to Z/XDC. At that point, Z/XDC has to wait for someone to talk to. It does so using VTAM requests. Accordingly, you must create several VTAM resources that cdf/XDC will require for communications between multiple debugging sessions and the multiple developers engaging with those session.

To create those resources, follow these steps:

1. ***hlq***.XDCV31.XDCSRVER(XSRVVTAM) contains a a factory default set of VTAM statements that define a major node to be named **XDCCDF** (See Figure 6 above). This major node is needed by the cdf/XDC subserver. These statements need to be copied to member XDCCDF of one of your system's VTAMLST libraries.
2. Note, XSRVVTAM is good "right out of the box". No changes are necessary. However, if you do want to change the names of the APPLID's defined to cdf/XDC, then be sure to follow carefully the instructions given in the commentary contained within XSRVVTAM.
3. If you want the XDCCDF node to come up at VTAM startup time, then add XDCCDF's name to the list of major node names found in an active ATCCONxx member of the VTAMLST libraries. Otherwise, the node will have to be started manually by an operator command ("V NET,ID=XDCCDF,ACT").
4. ***hlq***.XDCV31.XDCSRVER(XSRVPARM) contains control parameters required by the cdf/XDC subserver (see Figure 9 above). They tell cdf/XDC what the names are of the various APPLID groups that are defined in your VTAMLST(XDCCDF) file. Notes:

```
XDCCDF VBUILD TYPE=APPL
XDC APPL EAS=1,VPACING=0,AUTH=(ACQ,PASS)
XDCSIDE APPL EAS=1,VPACING=0,AUTH=(ACQ)

XDCRTE01 APPL EAS=1,VPACING=0,AUTH=(ACQ,PASS)
XDCRTE02 APPL EAS=1,VPACING=0,AUTH=(ACQ,PASS)
..

XDCTFS01 APPL EAS=1,VPACING=0,AUTH=(ACQ,PASS)
XDCTFS02 APPL EAS=1,VPACING=0,AUTH=(ACQ,PASS)
...
XDCTSO01 APPL EAS=1,VPACING=0,AUTH=(ACQ)
XDCTSO02 APPL EAS=1,VPACING=0,AUTH=(ACQ)
...
```

Figure 6: Typical XDCCDF member to be added to a VTAMLST library for cdf/XDC

```
<CDF>
VBUILD  XDCCDF
APPLID1 XDC
APPLID2 XDCSIDE
APPLID3 XDCRT
APPLID4 XDCTFS
APPLID5 XDCTSO
```

Figure 7: Typical SYSIN File Parameters for cdf/XDC

- If you have changed the names of the APPLIDs from what they were in XSRVVTAM, then you will have to make corresponding changes to XSRVPARM so that cdf/XDC will know what its APPLID names are. Follow carefully the instructions given in the commentary contained within XSRVPARM.
- The server/XDC's startup proc's //SYSIN ddname needs to point to wherever you put this file. Any FB-80 library would be suitable.

```
//XDCSRVER EXEC PGM=XDCSERVE,PARM='SERVER',REGION=0M
//STEPLIB  DD   DISP=SHR,DSN=APF-authorized library containing XDCSERVE [optional]
//SYSIN    DD   DISP=SHR,FREE=CLOSE,DSN=an.fb80.library(XSRVPARM)
//SYSPRINT DD   SYSOUT=* [optional]
```

Figure 8: Model Startup JCL for server/XDC

5. A startup proc for the XDCSRVER started task, such as is shown in Figure 10 above, needs to be added to one of your system's PROCLIBs. The proc's name typically is "XDCSRVER". A model for the proc can be found in ***hlq***.XDCV31.XDCSRVER(XSRVPROC). The JCL should be edited as indicated in Figure 8
6. server/XDC needs to have READ access to the XDC.AUTH[26] or AUTH[27] security rule. Please read the HELP SECURITY AUTH topic in **Z/XDC v3.1 User Guide** or in Z/XDC's Built-in Help for more information.

[26] For RACF
[27] For ACF2 or Top Secret

7. There are two ways that users can connect to batch job debugging sessions:
   - One is via a Z/XDC Startup Panel in ISPF.
   - The other is via a direct logon from an idle 3270 type terminal.

   If you chose to support direct logons, then you need to create RACF, ACF2 or Top Secret security definitions that support such logons. XDCSRVER needs to be defined to your security system such that it will permit the cdf/XDC subserver to make security calls to it in behalf of other users. Please refer back to **Step 4: Defining Z/XDC to Your Computer's Security System** on page 14 for details.
8. In addition, when a user connects into cdf/XDC looking for a debugging session to connect to, cdf/XDC has to examine all pending sessions, to see which one(s) the user has the right to connect to. For this purpose, cdf/XDC issues "RACHECK" type security calls to make this determination. This check is done before cdf/XDC displays to the user the list of jobs pending debugging. Therefore, if a pending job is not displayed that should be displayed, then this is an indication that the appropriate security system rules are not set up correctly.

   A consequence of this is that security logs may show rejected access attempts for the cdf/XDC user. These violations can be disregarded since the user has no control over these access attempts.

   The rule that cdf/XDC checks is:
   - "**FASM.*jobname***" for ACF2 and Top Secret systems,
   - "**XDC.FASM.*jobname***" for RACF systems.

   Refer back to **Step 4: Defining Z/XDC to Your Computer's Security System** on page 14 for details.
9. The following operator command need to be added to an active COMMNDxx member of your PARMLIB libraries:

   **COM='S XDCSRVER'**

   This command starts server/XDC. If the server is started too soon (i.e. before the XDCCDF VTAM node comes up), then the cdf/XDC subserver will just wait for VTAM to come up.
10. When server/XDC successfully completes its initialization, it will issue the following two messages and then wait for work to do:

    **DBC213I SUBSERVER MODIFY INTERFACE ACCEPTING COMMANDS.**
    **DBC655I ENTER "F XDCSRVER,HELP" FOR A LIST OF VALID COMMANDS.**

## Step 8: The Installation Verification Test

The Installation Verification Test should now be done in TSO. First, logon to TSO, and then start up ISPF. Then use one of the following methods to start Z/XDC, depending upon how you have installed the product:

**Method 1: Navigate to Z/XDC's Startup Panel, fill it in as shown in Figure 9 on page 29, and press ENTER.** This will work if you have installed Z/XDC's Startup Panel into ISPF. Note, if you have renamed Z/XDC (see **Step 10: Creating a Renamed Clone** on page 35), then the first thing you have to do is type Z/XDC's new name (V31, for example) on the command line and press ENTER. This will automagically reconfigure the panel to use the renamed Z/XDC.

```
------------------------- Z/XDC STARTUP PANEL ----------------------------(v6)
OPTION ===> 2                                            XDC's name ===> XDC
1 XDCCMD  - Debug TSO CMD processor - will be passed a CPPL
2 XDCCALL - Debug other PGMs in TSO - will be passed an OS PARM field
3 XDCCDF  - Debug background jobs or tasks via cdf/XDC

PARAMETERS FOR OPTION 1 OR 2 (FOREGROUND DEBUGGING IN TSO):
  Does CMD/PGM use ISPF services? (YES/NO) ===> NO     Dialog APPLID ===> TFS
  Should CMD/PGM start APF auth?  (YES/NO) ===> NO     Quick Start?  ===> NO
  Should AUTOSTEP be allowed?     (YES/NO) ===> YES
  Should RENT and REFR pgms be loaded into key-8 Storage? (YES/NO)   ===> YES
  Script DSN ===>
    CMD/PGM Name ===> IEFBR14          Upcase the PARM? ===> YES
    CMD/PGM PARM ===>

    Tasklib DSNs ===>
                 ===>
                 ===>
                 ===>
                 ===>
                 ===>
    or DDNAME    ===>
PARAMETER FOR OPTION 3 (BACKGROUND DEBUGGING VIA cdf/XDC)
Connect to cdf/XDC using your current TSO Userid? (YES/NO) ===> YES
```

Figure 9: Z/XDC's Startup Panel in ISPF

**Method 2: Navigate to ISPF's "Command Shell" (=6) and type XDCLBDEF:** This will work if you have suitably modified and installed the XDCLBDEF clist. If it does work, then you will see the Startup Panel shown in Figure reffig:zxdc-start-panel-ispf above.

**Method 3: Navigate to ISPF's "Command Shell" or to TSO's READY prompt and type: XDCCALL IEFBR14** This will work if you have installed Z/XDC's load libraries (***hlq***.XDCV31.XDCLINK, ***hlq***.XDCV31.XDCLINKE and ***hlq***.XDCV31.XDCPLPA) or load modules into the system's search order for your TSO session.

Doing any of these things should cause the fullscreen display shown in Figure 10 on page 30 or Figure 11 on page 31 to appear at your terminal.

```
TCB#9 RB#1 ----------------- Z/XDC ISPF INTERFACE ------------------------
XDC ===>
. DBC854I Z/XDC for z/OS
. DBC855I COPYRIGHT (C) COLESOFT PARTNERS, INC. - 2010-2016. ALL RIGHTS
. DBC855I RESERVED
. DBC856I FOR PRODUCT SUPPORT, PLEASE CALL COLESOFT AT 540-456-8210

. DBC575I Type ? on the command line to see the Helper Dialogs.
. DBC993I Type ? at left of any line for a list of its permitted Line Cmds.
. DBC992I For Built-in Help, type HELP, then press ENTER. (Not pf-key 1)
. DBC994I Type LIST HELP to display the Built-in Help Index.
. DBC996I Type HELP HELP for how to use Z/XDC's Built-in Help.

. DBC830I XDC v3.1 ENTERED NONAUTHORIZED, AMODE(24), UNDER RB#3 FROM TCB#9 IN
. DBC830I DBCOLEB (ASN=0095.17)
. DBC891I XDC v3.1 ENTERED AS AN ESTAI OWNED BY TCB#8
. DBC831I THE ERROR LEVEL AND RETRY LEVEL ENVIRONMENTS ARE THE SAME
_ DBC841I DEAD MSG: 2711 *** DBC901I SESSION IS READY FOR DEBUGGING, TYPE H AT
                    THE LEFT FOR MORE INFORMATION ***



XDC ===> L PSW;L EPSW;L REGS
_   PSW  078D1000 00E55000 (cc-LO) (24) - IEFBR14.IEFBR14+0
_  EPSW  078D1000 000689AA (cc-LO) (24) - DBC810W NOT WITHIN ANY IDENTIFIABLE M
_      R0  0004E428 0006BF90 E7C4C3C3 C1D3D340  *..U.....XDCCALL *
_      R4  C9C5C6C2 D9F1F440 00000000 00034E80  *IEFBR14 ......+.*
_      R8  00000000 00034018 000166E8 057F3818  *...... ....Y."..*
_      R12 857F2818 00068610 80068A26 00E55000  *e"....f......V&.*
   ONE OR MORE RHn CONTAINS NON-ZERO DATA
```

Figure 10: Initial Screen Displayed by Z/XDC on a Classic 43x80 3270 Terminal

```
TCB#10 RB#1 ------------------------------------- Z/XDC ISPF INTERFACE ------------------------------------------
XDC ===>
. DBC854I Z/XDC for z/OS
. DBC855I COPYRIGHT (C) COLESOFT PARTNERS, INC. - 2010-2016. ALL RIGHTS RESERVED
. DBC856I FOR PRODUCT SUPPORT, PLEASE CALL COLESOFT AT 540-456-8210

. DBC575I Type ? on the command line to see the Helper Dialogs.
. DBC993I Type ? at left of any line for a list of its permitted Line Cmds.
. DBC992I For Built-in Help, type HELP, then press ENTER. (Not pf-key 1)
. DBC994I Type LIST HELP to display the Built-in Help Index.
. DBC996I Type HELP HELP for how to use Z/XDC's Built-in Help.

. DBC830I XDC v3.1 ENTERED NONAUTHORIZED, AMODE(31), UNDER RB#3 FROM TCB#10 IN
. DBC830I DBCOLE7 (ASN=008B.23)
. DBC891I XDC v3.1 ENTERED AS AN ESTAI OWNED BY TCB#9
. DBC831I THE ERROR LEVEL AND RETRY LEVEL ENVIRONMENTS ARE THE SAME
_ DBC841I DEAD MSG: 2711 *** DBC901I SESSION IS READY FOR DEBUGGING, TYPE H AT THE LEFT FOR MORE
                    INFORMATION ***



 XDC ===> L PSWE;L EPSWE;L BEA;;L RWREGS;;L AREGS
_ PSWE  07851000 80000000 00000000_000AAE90 (cc-LO) (31) - DBCCALL.XDCCALL.XDCCALL
_ EPSWE 07851000 80000000 00000000_0009C9AA (cc-LO) (31) - DBC810W NOT WITHIN ANY IDENTIFIABLE MODULE OR ANY OTHER OBJECT
_ Breaking Event Address (BEA): XDCCALL+273E

_    RW0  00000000_00055428 00000000_0009FF90  FFFFFFFF_E7C4C3C3 FFFFFFFF_C1D3D340  *...................XDCC....ALL *
_    RW4  FFFFFFFF_C4C2C3C3 FFFFFFFF_C1D3D340  FFFFFFFF_00000000 FFFFFFFF_0003AE80  *....DBCC....ALL ................*
_    RW8  FFFFFFFF_00000000 FFFFFFFF_0003A018  FFFFFFFF_000166E8 FFFFFFFF_057F3818  *.....................Y....."..*
_    RW12 FFFFFFFF_857F2818 00000000_0009C610  00000000_8009CA26 00000000_800AAE90  *....e".....F.......-.........*

_    AR0  00000000 00000000 FFFFFFFF FFFFFFFF  FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF  *................................*
_    AR8  FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF  FFFFFFFF FFFFFFFF 00000000 00000000  *................................*
```

Figure 11: Initial Screen Displayed by Z/XDC on a Terminal Set to Large Dimensions

If, however, you see a display such as that shown in Figure 14 below, then most likely Z/XDC's various ISPF elements (panels, tables, etc.) have not been properly made available to ISPF. Go back and review the **Setting Up the XDC-PANEL Panel or the XDCLBDEF Clist in ISR@PRIM (or ISRUTIL)** on page 11 for for more information.

Meanwhile, if you just press the **ENTER** key, Z/XDC will proceed with a normal debugging session but will fall back to using fullscreen TPUTs (instead of ISPF's Display Services) to paint its displays. The main consequence of this is that you will not be able to use ISPF's SPLIT, SWAP and =jump commands while using Z/XDC.

```
TCB#9 RB#1 ------------------ Z/XDC TPUT INTERFACE --------------------------
XDC ===>
   ISPP100 Panel 'XDCV31A' error
   ISPP100 PANEL NOT FOUND.
.  DBC750I PRESS ENTER TO CONTINUE DEBUGGING USING Z/XDC'S FULLSCREEN TPUT DISPLAY METHOD
```

Figure 12: Error When Z/XDC is Not Properly Installed into ISPF

In any case, once you see the display in Figure 10 on page 30 or Figure 11 on page 31, it is verified that Z/XDC is installed in the proper system libraries and that it is operational. You may proceed, if you wish, to try out the various Z/XDC commands.

When you are done you can terminate Z/XDC by pressing PF3 or PF4 or by typing END. This will cause the IEFBR14 program to abend with an s0C1. This is a normal consequence of the END command which (unlike the GO command) requests Z/XDC to allow the intercepted abend to percolate (i.e. to continue abending).

### Testing Authorized Debugging

If you have installed the authorized commands XDCCALLA and XDCCMDA, then you should test them now. You may do either of the following:

- **Go to Z/XDC's Startup Panel, fill it in as shown in Figure 13 on page 32, and press ENTER.** (In particular, **before** pressing ENTER, be sure to respond YES to the question: "**Should CMD/PGM start APF auth?"**) This will work if you have installed Z/XDC's Startup Panel into properly.
- **Or go to ISPF's "Command Shell" or to TSO's READY prompt and type: XDCCALLA IEFBR14** This will work if you have installed Z/XDC's load libraries (***hlq***.XDCV31.XDCLINK, ***hlq***.XDCV31.XDCLINKE and ***hlq***.XDCV31.XDCPLPA) or load modules into the system's search order for your TSO session, and also added XDCCALLA (and XDCCMDA) to an active IKJTSOxx member of the PARMLIB libraries.

```
-------------------------- Z/XDC STARTUP PANEL ----------------------------(v6)
OPTION ===> 2                                            XDC's name ===> XDC
1 XDCCMD  - Debug TSO CMD processor - will be passed a CPPL
2 XDCCALL - Debug other PGMs in TSO - will be passed an OS PARM field
3 XDCCDF  - Debug background jobs or tasks via cdf/XDC

PARAMETERS FOR OPTION 1 OR 2 (FOREGROUND DEBUGGING IN TSO):
  Does CMD/PGM use ISPF services? (YES/NO) ===> NO     Dialog APPLID ===> TFS
  Should CMD/PGM start APF auth?  (YES/NO) ===> YES     Quick Start?  ===> NO
  Should AUTOSTEP be allowed?     (YES/NO) ===> YES
  Should RENT and REFR pgms be loaded into key-8 Storage? (YES/NO)   ===> YES
  Script DSN ===>
    CMD/PGM Name ===> IEFBR14          Upcase the PARM? ===> YES
    CMD/PGM PARM ===>

    Tasklib DSNs ===>
                 ===>
                 ===>
                 ===>
                 ===>
                 ===>
    or DDNAME    ===>
PARAMETER FOR OPTION 3 (BACKGROUND DEBUGGING VIA cdf/XDC)
Connect to cdf/XDC using your current TSO Userid? (YES/NO) ===> YES
```

Figure 13: Starting Z/XDC Authorized from its Startup Panel in ISPF

## Step 9: cdf/XDC Installation Verification

In order to test cdf/XDC, you first must have a background job that has abended and is asking for cdf/XDC services. The job shown in Figure 14 below will serve this purpose.

```
//      --- JOB ---
//IVP      EXEC PGM=XDCIVP
//STEPLIB  DD   DISP=SHR,DSN=hlq.XDCV31.XDCLINK
//         DD   DISP=SHR,DSN=hlq.XDCV31.XDCLINKE
//         DD   DISP=SHR,DSN=hlq.XDCV31.XDCPLPA
//XDCPROF  DD   DISP=SHR,DSN=Your ISPF profile library [optional]
```

Figure 14: Model JCL for Testing cdf/XDC

The //XDCPROF DD card, if present, is used by Z/XDC to find the user's profile data (member XDCDPV31, assuming it exists). From the profile, Z/XDC determines such things as how long it should wait for a user to log onto the debugging session and whether or not it should display a WTOR to the operators permitting them to abort the logon wait prior to its timing out.

If //XDCPROF is omitted, then Z/XDC uses the "factory defaults" documented in HELP PROFILES FACTORYDEFAULTS.

There are two ways users can log onto cdf/XDC:

- TSO users with ISPF can use Z/XDC's Startup Panel ("XDCPANEL", see **Adding XDCMLIB, XDCPLIB and XDCTLIB to ISPF** on page 10). From that panel, select option **3** to connect to cdf/XDC.
- Other users can simply logon directly to cdf/XDC from an idle VTAM terminal. By default, the necessary command is "LOGON APPLID=name".[28][29] cdf/XDC will then present the user with a screen requesting his TSO userid and password (Figure 15 on page 34).

[28] Where "name" matches the **APPLID1** definition found in server/XDC's **//SYSIN** parameters. If cdf/XDC has been installed as directed, then "*name*" will match Z/XDC's name (usually **XDC**).

[29] At some data centers, the correct command would be "**LOGON APPLID(name)**".

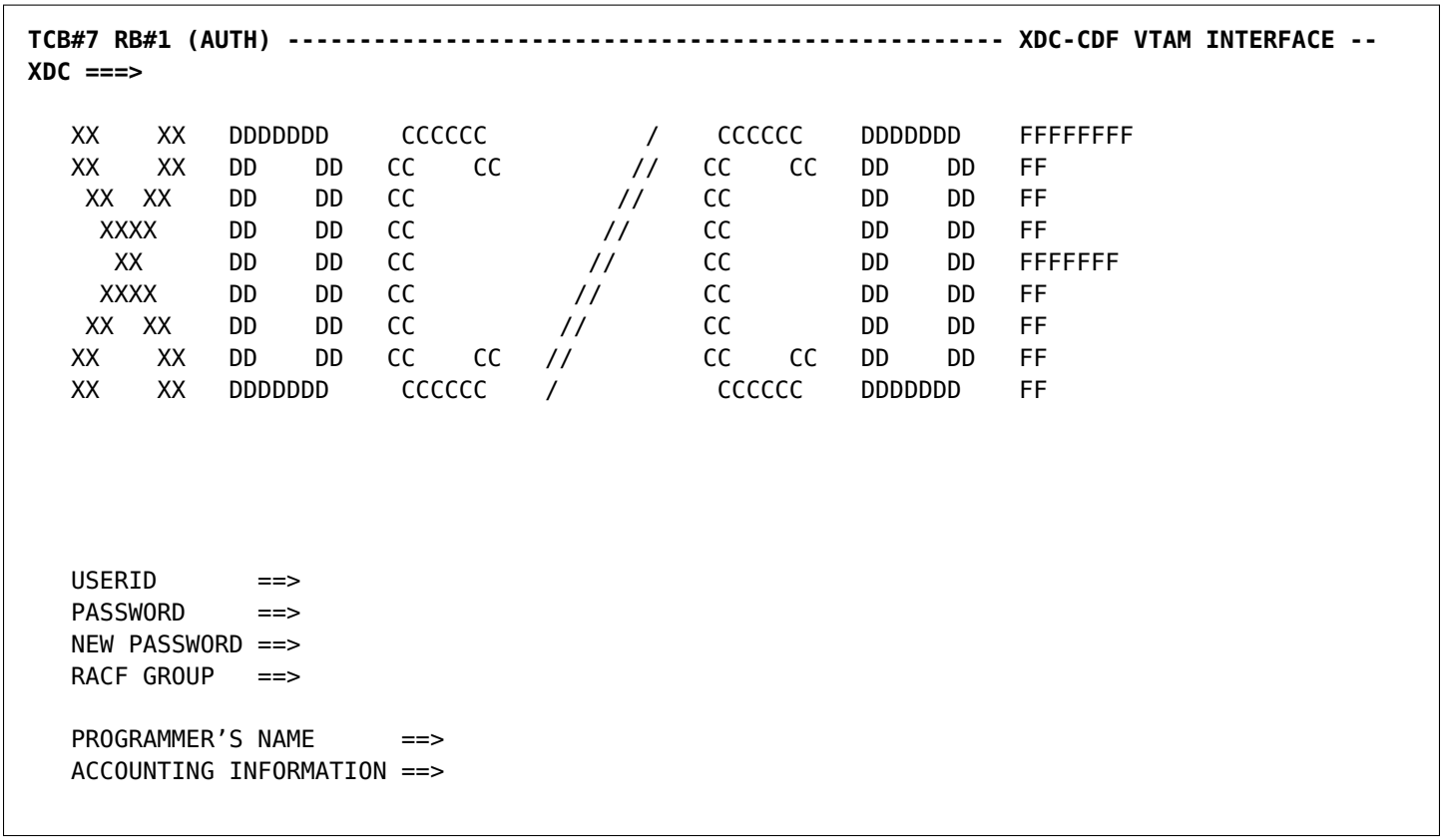


Figure 15: Logon Screen for VTAM Connections to Z/XDC's Cross Domain Facility

Once within cdf/XDC, the user will be presented with a list of abended background jobs and system tasks that he is permitted (according to system security) to debug (Figure 16 below). The user should merely select the desired job (by typing an S next to the job's name). He will then be connected to the job's debugging session. The will see a normal Z/XDC debugging screen, and he will be able to issue normal Z/XDC commands.

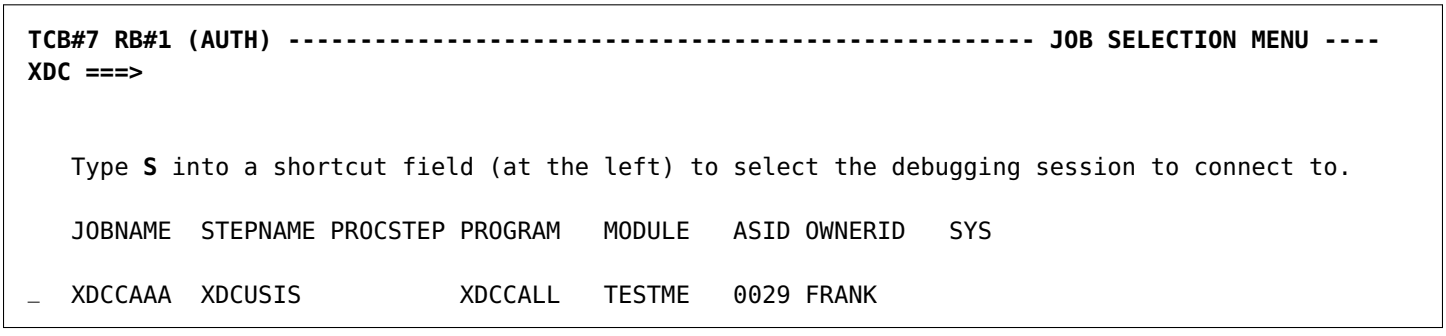


Figure 16: cdf/XDC Job Selection menu

For further information, though, the user should, before proceeding, read the HELP XDCSRVER CDF topic (at cole-soft.com/help).

## Step 10: Creating a Renamed Clone

If you want to install Z/XDC v3.1 into your system and have it coexist with a different version or release of XDC, you can do so, but in order to do so, you will have to rename one or the other of the XDCs. (Usually, one would decide to rename the newer release of Z/XDC or a beta version of Z/XDC, perhaps for a **try out** period.) You may rename Z/XDC to any other 3-character name you choose. One such name might be V31. (But my personal favorite is DBC.)

The names of all Z/XDC load modules in ***hlq*.XDCV31.XDCLINK**, ***hlq*.XDCV31.XDCLINKE**, ***hlq*.XDCV31.XDCPLPA** and ***hlq*.XDCV31.XDCGUI** start with the characters XDC. These first three characters can be changed to V31, to your initials or to any other legal three characters you choose. The remaining characters in each name **must not** be altered.

A renamed Z/XDC is referred to as a Z/XDC **clone**.

Before deciding to rename Z/XDC, consider the following:

- If you change the names of any Z/XDC load modules, then you must change the names of all of them in all three load libraries: ***hlq*.XDCV31.XDCLINK**, ***hlq*.XDCV31.XDCLINKE**, ***hlq*.XDCV31.XDCPLPA** and ***hlq*.XDCV31.XDCGUI**.
- The Z/XDC load modules may be renamed directly within the Permanent Product Libraries; but remember, in order to apply maintenance, you will have to rename them all back to **XDC***name*, then apply the maintenance, and then rename them all back to their clone names.
- You will have to repeat this process each time that you apply Z/XDC maintenance.
- If user programs are written to invoke Z/XDC internally, then they may become dependent upon the name that you choose. Thus, your name choice may become "locked in" over time.

### Considerations for Using a Renamed Z/XDC

The ability to create renamed clones of Z/XDC makes it possible to simultaneously run any combination of Z/XDC versions and releases that you wish. However, there are several considerations that need to be taken into account. (For the purposes of these illustration, let me presume that you have chosen to create a clone named of **V31**.)

- Users will have to invoke Z/XDC using clone names (**V31***name* names) instead of XDCname names.

  Example:

  ```
  //A EXEC PGM=V31CALLA,PARM='IEFBR14'
  ```

- If your programs have hardcoded LOADs for the XDC load module, they will have to be changed to use the clone name instead.

  Example:

  ```
  LOAD EPLOC==CL8'V31'
  ```

- Suggestion: Code your program to obtain Z/XDC's name via a parameter, a command, a keyword ddname or some other mechanism.

  My personal favorite way to do this would be to scan the TIOT for a ddname of the form //XDCISxxx, where xxx would be the desired clone name. Then to create such a TIOT entry, all you'd have to do is add a JCL card.

  Example:

  ```
  //XDCISV31 DD DUMMY.
  ```

  For more information, see HELP XDCCLONES AXDCIS.

- Any //XDC*whatever* DD cards or dynamic allocations you might have for your debugging sessions (profiles, traces, other controls) will have to be changed.

  Examples:

//**V31**QUICK DD DUMMY
//**V31**PROF DD DISP=SHR,DSN=*userid*.ISP.ISPPROF

- At IPL time, you will need to run a V31SRVER job similar to the XDCSRVER job described in **Step 7: Installing Z/XDC's Server Job** on page 25.
- In ISPF, to use the V31 name, on the XDC Startup Panel, you simply have to type **V31** on the command line and then press ENTER. The panel will automatically adjust itself to invoked V31 instead of XDC.
- You would have to create a V31CDF VTAM node similar to that described **Setting up cdf/XDC** on page 26. Briefly, you would need to do the following:
  - Make a copy of your XDCSRVER startup proc, and rename both it and everything within it from XDCwhatever to **V31**whatever.
  - Make a copy of the //SYSIN file referenced by your **V31**SRVER startup proc, and perform similar renames within it.
  - Make a copy of your XDCCDF VTAMLST member, rename it to **V31**CDF and perform similar renames within it.
  - For auto-start of V31CDF's nodes at IPL time, add **V31**CDF to an active ATCCON*xx* member of the VTAMLST libraries.
  - Also, add a COM='S **V31SRVER**' command to an active COMMND*xx* member of the PARMLIB libraries.
- For TSO, you will have to add **V31**CALLA and **V31**CMDA to the AUTHCMD section of an active IKJTSO*xx* member of the PARMLIB libraries.
- Note, security system rules do not have to be specifically created or changed for any clone. All clones of Z/XDC will honor security rules based upon the name XDC (not the clone's name).
- However, for sites that use the ACF2 security system, the following actions have to be taken:
  - ACF2 has an optional table of permitted TSO commands (both authorized and nonauthorized). If present, then that table needs to have entries added to it for **V31**CALL, **V31**CMD, **V31**CALLA, **V31**CMDA, **V31**TFS and **V31** itself.
  - A **V31**SRVER logonid needs to be created for the Cross Domain Facility. This id, like the XDCSRVER id, must be assigned at least the following attributes: MUSASS, STC and RESTRICT.
- Z/XDC Profiles: When a user first starts up V31, if he had a saved profile in the older XDC, then V31 will not be able to find it automatically. So to access his old profile, a user will have to issue the following commands:

  PROFILE READ DPV31 XDC
  PROFILE SAVE

The PROFILE SAVE command will cause the profile to be rewritten using the name V31DPV31 (Z/XDC's default profile name when running under the clone name of V31).

The remainder of this **Install Guide** assumes that Z/XDC has not been renamed.

### Step 11: Making hlq.XDCV31.XDCADATA the Default MAPLIB Library

The ***hlq***.XDCV31.XDCADATA library contains ADATA[30] for a few of Z/XDC's load modules, the most important of which is the XDCMAPS load module. XDCMAPS contains mapping information (SYM data) for a very large number of z/OS system control blocks. The ***hlq***.XDCV31.XDCADATA library contains corresponding source image information (ADATA) for those same control blocks. Consequently, it is useful to set up ***hlq***.XDCV31.XDCADATA as a default place for Z/XDC to look when it needs to build dsect maps in response to a DMAP command.

[30] **ADATA** is a file of data produced during the assembly of a program. It contains a large amount of information about the source code of that program and the assembly process for that program. ADATA can be produced both by IBM's High Level Assembler and by Tachyon Software's Cross Assembler and Dignus' Systems/ASM assembler.

The place where Z/XDC looks for ADATA is called a "MAPLIB library", and Z/XDC has a SET MAPLIBS command whereby users can build and save one or more lists of MAPLIB libraries.

Z/XDC also has a facility whereby you can create a default MAPLIBS list that will be automatically available to a user's debugging session without requiring him to issue the SET MAPLIBS command. This default MAPLIB information is kept in the user's session profile.

The ***hlq***.XDCV31.XDCADATA library (or whatever you have renamed it to) is a good candidate for being a default MAPLIB library. To accomplish this, you will have to start a Z/XDC debugging session and then issue the following commands:

**SET MAPLIBS RESET *hlq*.XDCV31.XDCADATA**[31] **SAVE**
**PROFILE SAVE**

But don't actually issue these commands yet. Read the next section ("Creating a Default Profile: TFSDPV31") first.

#### Step 12: Creating a Default Profile: TFSDPV31

TFSDPV31, when it exists, is a system-wide default profile for Z/XDC (see **Adding XDCMLIB, XDCPLIB and XDCTLIB to ISPF** on page 10). When a user invokes Z/XDC for the first time, if a TFSDPV31 member exists in an ISPF Table Library (//ISPTLIB), then that is loaded for his use.[32] It is loaded by Z/XDC regardless of whether Z/XDC is running within or outside of an ISPF environment.

If a user changes his profile data and then issues a PROFILE SAVE XDC command, then Z/XDC stores the changed profile into the user's own profile library in a member named ***xxx***DPV31, where "***xxx***" matches Z/XDC's name (see **Step 10: Creating a Renamed Clone** on page 35).

Subsequently, the next time this user starts Z/XDC, his *xxx*DPV31 will be loaded instead of the System-wide TFSDPV31.

### Reasons to Create or not Create a System-Wide Default Profile

If a user already has his own default profile, either from the current Z/XDC release, or from any older release, then Z/XDC will ignore a Data Center's System-wide default profile. In such cases, the only way a user could load the System-wide default would be to explicitly issue a PROFILE READ XDC TFS command. (See HELP COMMANDS PROFILE READ for details.)

When a System-wide default profile (TFSDPV31) does exist, it will be used automatically only when the user has no current or older default profile of his own.

When a System-wide default profile (TFSDPV31) does not exist, then Z/XDC will automatically use one of its own built-in Factory Default profiles. (See HELP PROFILES FACTORYDEFAULTS for details.)

So if the factory default profile settings are OK as they are, then there is no reason for you to go to the trouble of creating a local version. However, it is likely that there are several factory default settings your programmers will not like. For detailed suggestions about what you should consider changing, see **Which System-Wide Default Profile Values You Might Consider Changing** on page 39.

To find out what the factory default settings are and what they mean, refer to HELP PROFILES FACTORYDEFAULTS. This topic can be found both in Z/XDC's Built-in Help and in the Z/XDC User Guide.

[31] Or whatever you have renamed this library to.

[32] If a user has previously saved his own Z/XDC profiles, then Z/XDC will not automatically load TFSDPV31 Instead, it will load the user's profile and then automatically convert it, if necessary, to v3.1 format. But Z/XDC will not write the converted profile back to disk unless and until the user issues a PROFILE SAVE command.

How to Create Z/XDC's System-Wide Default Profile If you wish to change any of the factory default settings, then you will have to create a System-wide default profile for use at your Data Center. See **Which System-Wide Default Profile Values You Might Consider Changing** (on page 39) for suggestions about which profile settings you might consider changing first.

### How to Create Z/XDC's System-Wide Default Profile

If you wish to change any of the factory default settings, then you will have to create a System-wide default profile for use at your Data Center. See **Which System-Wide Default Profile Values You Might Consider Changing** (on page 39) for suggestions about which profile settings you might consider changing first.

- Start up Z/XDC. (From ISPF's Command Shell [=6], issue XDCCALL IEFBR14.)
- Issue PROFILE RESET *[name]* to insure the desired set of Factory Defaults is loaded. **[name]** can be one of AWIDE, A80, CWIDE, C80, DWIDE or D80. (See HELP COMMANDS PROFILE RESET for more information.)
- Issue: **SET MAPLIBS RESET *hlq*.XDCV31.XDCADATA**[33] **SAVE** to make the ***hlq***.XDCV31.XDCADATA library the default MAPLIB library for future debugging sessions. (See the preceding section, **Step 11: Making *hlq*.XDCV31.XDCADATA the Default MAPLIB Library**, page 36, for more information.)
- Issue **PROFILE** to start the Profile Menuing System (see Figure 20 on page 49).
- One by one, select (with an **S** om the shortcut field at the left) each of the menu's panels and make those changes you deem necessary.
- Use **PF3** to close each menu panel and move on to the next.
- Use **PF3** again to exit the Profile menuing System altogether.
- Type **PROFILE SAVE XDC TFS**s to save the changed profile back into your profile library under the member named TFSDPV31.
- Leave Z/XDC and invoke ISPF's "Move/Copy Utility" (=3.3).
- Move (not just copy) your changed profile (member name TFSDPV31) from your profile library to your system's ISPF table library.

[33] Or whatever you have renamed this library to.

```
TCB#5 RB#1 ------------------------------------------- Profile Home Menu (Select submenu by number) --
XDC ===>
.  DBC722I Press PF3, PF4 or PF5 to exit the menus.

   Current profile's name (version):  XDC (v3.1A)
   Loaded from:                       //XDCPROF DSN=FRANK.ISPF.ISPPROF(XDCDPV31)
   Save Status:                       Save pending (Settings have changed)
   Saved when:                        Upon explicit request
   Should this profile be saved now?  NO
   Profile Description:               [none]

   Select one or more of the following menu options:
    _  01 Display/Change TSO/ISPF/CDF/CICS Related Settings
    _  02 Display/Change Terminal Settings
    _  03 Display/Change Color and Hilighting
    _  04 Display/Change PF Key to Function Key Assignments
    S  05 Display/Change Function Keys
    _  06 Display/Change Function Keys for Help Topics
    _  07 Display/Change Session Logging Parameters
    _  08 Display/Change Window Controls
    _  09 Display/Change READ ZAP and Parse Order Settings
    S  10 Display/Change FORMAT TRACE and FIND Settings
    _  11 Display/Change AUTOMAP PRINT and Other Settings
    _  12 Display/Change REXX settings
    _  13 Display/Change c/XDC Settings
    S  14 Display/Change USS Settings
```

Figure 17: Z/XDC's Profile Menu System

### Which System-Wide Default Profile Values You Might Consider Changing

As discussed above, it is important to set up the ***hlq*XDCV31.XDCADATA** library as a default MAPLIB library for use by your user community. If you haven't done so already, then please see **Step 11: Making *hlq*.XDCV31.XDCADATA the Default MAPLIB Library** on page 36 for more information.

Initially, there should be very few other profile items whose defaults should be changed on a system-wide basis. The initial default values have received considerable thought, and I feel that they are good for most users. Of course, as users gain experience with Z/XDC, they quite likely will want to make specific changes on an individual basis. In any case, the following default profile settings might be worth changing on a system-wide basis:

**Individual PF-key Assignments:** Many of Z/XDC's factory default PF-key assignments are "real nonstandard". (See Figure 18 on page 40.) Accordingly, the temptation, for many people, is high to change them, but please think twice before doing so.

The factory default definitions have been well thought out. For example, SET WINDOW CREATE is a very powerful and useful command, but is relatively hard to type. HELP, on the other hand, also is powerful and useful, but is relatively easy to type. Accordingly, I have chosen to make SET WINDOW CREATE the default setting for PF1 (not HELP).

```
TCB#5 RB#1 ------------------------------------------------------------ 05 Function Keys---
XDC ===>
.  DBC722I Press PF3 or PF5 to accept changes and return.
.  DBC722I Press ENTER to see next panel.

   FUNCTION  PF KEYS
   KEYS      SET A     CURRENT SETTING
    1         1st      SET WINDOW DELETE
    2         2nd      SPLIT
    3         3rd      END
    4         4th      END COMPLETELY
    5         5th      FIND
    6         6th      TSO -
    7         7th      UP -
    8         8th      DOWN -
    9         9th      T
   10        10th      T BY
   11        11th      T NXSEQ() NXSEQ(2) NXSEQ(3) NXSEQ(4) NXSEQ(5) NXSEQ(6);G
   12        12th      RETRIEVE

    _ Select here for 06 Function Keys for Help Topics
    _ Select here for 04 PF Key to Function Key Assignments



TCB#5 RB#1 ------------------------------------------------------------ 05 Function Keys---
XDC ===>
.  DBC722I Press PF3 or PF5 to accept changes and return.
.  DBC722I Press ENTER to see next panel.

   FUNCTION  PF KEYS
   KEYS      SET B     CURRENT SETTING
   13         1st      SET WINDOW CREATE
   14         2nd      SPLIT NEW
   15         3rd      END
   16         4th      END COMPLETELY
   17         5th      SCANLOG -
   18         6th      TSO -
   19         7th      UP M
   20         8th      DOWN M
   21         9th      SWAP NEXT
   22        10th      LEFT -
   23        11th      RIGHT -
   24        12th      RETRIEVE +1

    _ Select here for 06 Function Keys for Help Topics
    _ Select here for 04 PF Key to Function Key Assignments
```

Figure 18: Default PF-key Definitions

Also notice that several of the PF-key definitions end with a hyphen. This causes the command line contents (if any) to be appended to the command string (as command operands) before it is processed. This turns out to be a very useful feature.

As I said, some of the defaults are real nonstandard, but there are good reasons for this. For detailed discussions about the default settings (what they are, what they do and why), please read the HELP FULLSCREEN PFKEYS DEFAULTKEYS topic in the Built-in Help.

**Session Log File Allocation:** Z/XDC supports automatic or manual logging of debugging sessions either to disk or SYSOUT spool files. The initial default is for Z/XDC to automatically log all debugging sessions to "HOLD" class X on spool. If "X" is not your standard held output class, then you may change the default to an any class you prefer. (See Figure 19 below.)

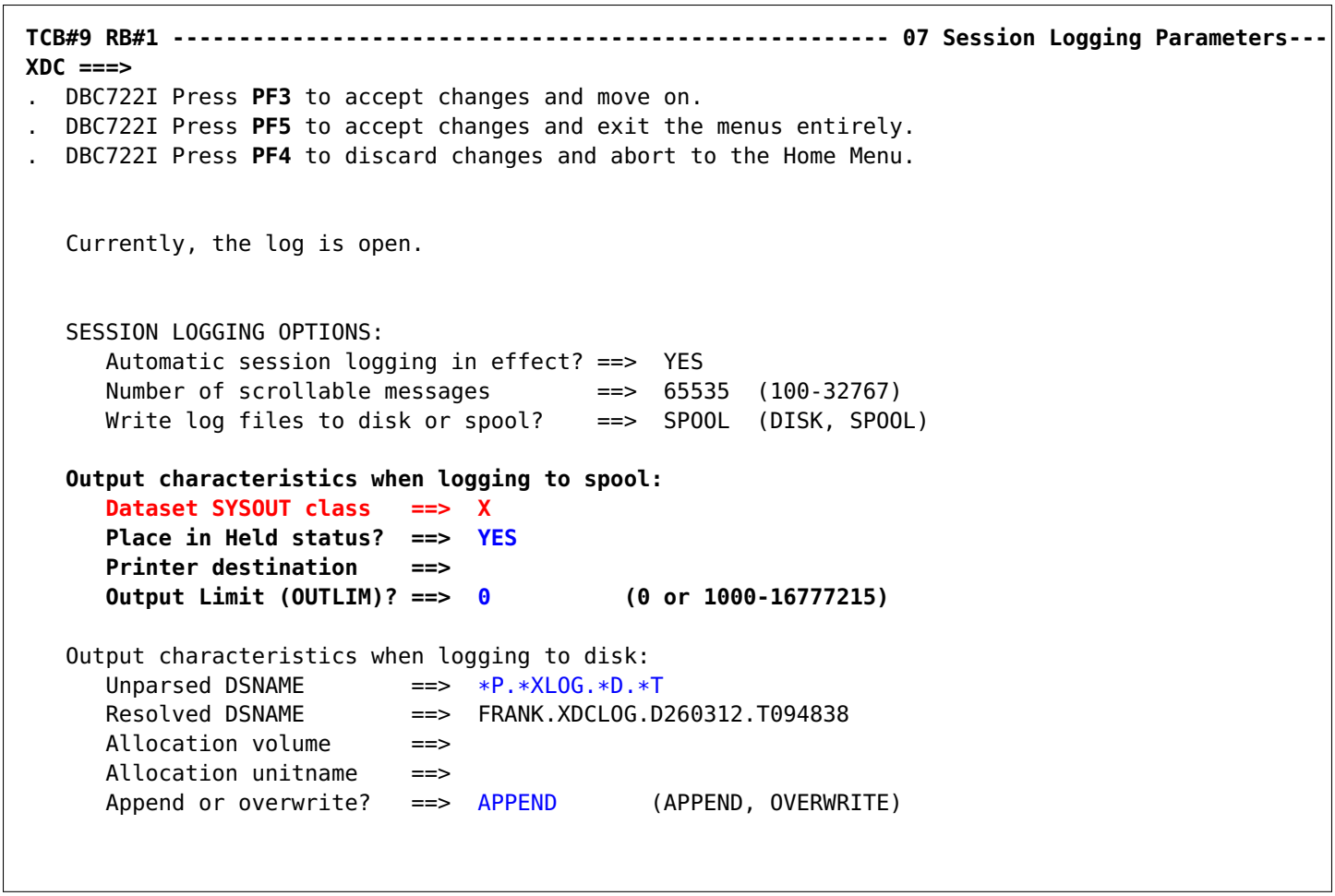


Figure 19: Profile Settings for Z/XDC's Session Log

I would not, however, recommend turning off logging since a user cannot predict when it might become useful.

I also would not recommend rerouting the log from spool to disk because that would lead to an accumulation of disk datasets that someone, from time to time, would have to go to the trouble of deleting.

A held class on spool is ideal because if it is needed, it can be very easily printed or offloaded to disk, and if it's not needed, then Operations at many computer centers typically runs a periodic command that automatically purges all held output that has remained too long on spool.

**Changing the Default Scripts Library (used by Z/XDC's READ command):** The default profiles all set the default scripts library to ***hlq***.XDCV31.XDCCMDS. But if you have installed Z/XDC's Permanent Product Libraries using a different HLQ (High Level Qualifier), then you will want to change the scripts library name in your system-wide default profiles. The Profile Menu panel for doing this is shown in Figure 20 below. Simply overtype ***hlq***.XDCV31.XDCCMDS with the actual name of the scripts library you want to use.

```
TCB#9 RB#1 ------------------------------------------------ 09 READ ZAP and Parse Order Settings---
XDC ===>
.  DBC722I Press PF3 to accept changes and move on.
.  DBC722I Press PF5 to accept changes and exit the menus entirely.
.  DBC722I Press PF4 to discard changes and abort to the Home Menu.

           09 READ ZAP and Parse Order Settings              CHOICES
    NO     Allow Zapping of Store Protected Storage?         YES    NO
    YES    Limit Mnemonic Zaps by Instruction Length?        YES    NO
    YES    Echo Scripted Cmds and Displays as They Happen?   YES    NO
           Default Scripts Library Name hlq.XDCV31.XDCCMDS
    notset Script File Sequence# Field?                      PRESENT ABSENT DETECT

    ASM    1st Language in the Parse Order                   ASM  CEE  NONE
    CEE    2nd Language in the Parse Order                   ASM  CEE  NONE
           3rd Language in the Parse Order                   ASM  CEE  NONE
           4th Language in the Parse Order                   ASM  CEE  NONE
           5th Language in the Parse Order                   ASM  CEE  NONE
           6th Language in the Parse Order                   ASM  CEE  NONE
           7th Language in the Parse Order                   ASM  CEE  NONE
           8th Language in the Parse Order                   ASM  CEE  NONE
```

Figure 20: Profile Settings for READ, ZAP and PARSEORDER

**//XDCPRINT File Characteristics:** Z/XDC supports a PRINT command that users can use to print various things such as selected portions of the Built-in Help and even entire Z/XDC manuals. Users can use the Profile Menu System to customize the characteristics of the printed output file. (See Figure 21 below.) You may want to check out these settings and perhaps change them if necessary to match the characteristics of your most typical printers.

```
TCB#9 RB#1 ------------------------------------------------- 11 AUTOMAP PRINT and Other Settings---
XDC ===>
.  DBC722I Press PF3 to accept changes and move on.
.  DBC722I Press PF5 to accept changes and exit the menus entirely.
.  DBC722I Press PF4 to discard changes and abort to the Home Menu.

                11 AUTOMAP PRINT and Other Settings      CHOICES
    YES         AUTOMAP Programs Upon First Execution?   YES    NO

    108         XDCPRINT Lines Per Page                  10-255 or 0 (no pagination)
    X           XDCPRINT and SYSUDUMP output class       A-Z, 0-9, or *
    DEFAULT     XDCPRINT Output Status                   HOLD   NOHOLD  DEFAULT

    NO          Intercept/Debug CANCEL/DETACH ABENDs?    YES    NO
    64BIT       Meaning of ! Indirect Operator           AMODE  64BIT
    5           Multiconversation Timeout Control        1 to 3600 seconds.
    ~           Built-in Function Leader Character       ¢ ‘ ~
```

Figure 21: Profile Settings for AUTOMAP, PRINT and Other Settings

Refer to the HELP PROFILES topic (in **Z/XDC v3.1 User Guide** or in Z/XDC's Built-in Help) for more detailed information about user profiles.

## Step 13: Exit Routines

If you or your users have exit routines that have been written for prior versions or releases of Z/XDC, then you probably should reassemble them. There may have been some changes in the programming interface that the exits use. See **Z/XDC v3.1 Release Guide** for more information. Also, check out the commentary located in the #DBCPARM macro found in the ***hlq*.XDCV31.XDCMACS** library.

Z/XDC contains interfaces for an "Installation exit" and for several "User exits":

- **User exits** generally are provided by the individual users of Z/XDC to meet specific needs that may arise during the debugging of specific programs or subsystems. They can be implemented on a user by user basis.
- **Installation exits** are more appropriately implemented on a system-wide basis.

For more information about User Exits, please read the HELP EXITS topic in **Z/XDC v3.1 User Guide** or in Z/XDC's Built-in Help.

The supported User Exits are:

### A User Communications Interface (UCI) Exit

The User Communications Interface Exit can be used to replace Z/XDC's own user communications interfaces. Normally, Z/XDC will communicate with the user via either TPUT/TGET's, VTAM SEND/RECEIVE's or WTO/WTOR's. But with a UCI exit, user communications can be directed towards any device or path desired. For more information see the HELP EXITS UCI topic in **Z/XDC v3.1 User Guide** or in Z/XDC's Built-in Help.

Source code for a sample UCI exit can be found in ***hlq***.XDCV31.XDCASM(SRCONLHG).

#### Consolidated Exits

Z/XDC currently supports the following five consolidated exits:

**An "Initialization" Exit** The Initialization Exit is called every time Z/XDC is entered in a non-abortive environment, i.e., whenever Z/XDC is called with an SDWA provided. (Z/XDC aborts when it is called without an SDWA).

**A "Resumption" Exit** The Resumption Exit is called every time Z/XDC is about to return to its caller (usually the RTM, the system's "recovery/termination manager") with "retry" signaled.

**A "Termination" Exit** The Termination Exit is called every time Z/XDC is about to return to its caller with "percolate" signaled

**A "User SVC" Exit** The User SVC Exit is called by Z/XDC when it is processing a TRACE command and the current instruction is a user SVC (or an **EX** or **EXRL** of a user SVC). Sometimes, user SVC routines make non-standard returns to locations other than the immediately following instruction. This exit can be used to predict for Z/XDC when a user SVC will make such a non-standard return and where that return will be made to. This exit is necessary during Z/XDC tracing in order for Z/XDC not to lose control of the trace.

**A "User Commands" Exit** The User Commands Exit is called by Z/XDC when it has been given a command that it does not recognize. The exit may process or reject the command. Interfaces to several internal Z/XDC service routines are made available to the exit.

Source code for a sample User Commands Exit can be found in ***hlq***.XDCV31.XDCASM(SRCXUCMD). (SRCXUCMD implements a LIST TIOT command for Z/XDC.)

All consolidated exits must be packaged together into a single module and front-ended by a user written router routine. Thus, if any consolidated exit is written, then at least null versions of all consolidated exits must be written. Use of consolidated exits is documented in the HELP EXITS MISCXITS topic in **Z/XDC v3.1 User Guide** or in Z/XDC's Built-in Help.

Source for a sample router module can be found in ***hlq***.XDCV31.XDCASM(SRCXITSR).

#### A "Front-end/Back-end" Routine

Every time Z/XDC receives control, it GETMAIN's and builds a multi-page temporary data area. Every time it releases control, it FREEMAIN's that area. Z/XDC has support that allows a front-end routine to provide the pages of storage that Z/XDC uses for its temporary data. This has been useful for making Z/XDC operate in certain highly constrained environments found at one or two customer sites. Please see the HELP EXITS topic in **Z/XDC v3.1 User Guide** or in Z/XDC's Built-in Help.

## Step 14: Setting up the gui/XDC Server

**!!!IMPORTANT!!!** There are several prerequisites to sort out before setting up the Z/XDC GUI Server.

- zOS 3.1 or higher. [34]
- The VS Code extension communicates with z/OS HTTP Server, therefore z/OS HTTP Server is required.
  - Starting with z/OS 2.2, z/OS HTTP Server is part of the base z/OS.
  - You may still need to perform some initial setup and configuration on your system.
  - On our z/OS 3.1 ADCD system, z/OS HTTP Server is installed in z/OS UNIX under the **/web/httpd1** directory. This directory will be referenced throughout this document. Substitute with your installation directory.
  - You need to know the SAF user id and group for z/OS HTTP Server. On our system, they are ***WEBSRV*** and ***WEBGRP***. You should have something similar on your system.
  - The z/OS HTTP Server is typically started via a PROC on the z/OS (MVS) side of things. You will need to know the name of this PROC and have the authority to stop and start this PROC.
  - You will need to have access to a z/OS UNIX ID with UID(0) and GID(0) or sudo in order to edit the z/OS HTTP Server config file.
  - A quick test to see if the z/OS HTTP Server is up and running is to point your browser to your z/OS instance's IP address. If the page loads, then the z/OS HTTP Server is up and running. If not, please make sure it is not blocked by a firewall rule somewhere.
- You will also need to be able to define a DNS entry for the url used by the VS Code Extension.
- Our VS Code extension uses HTTPS communications exclusively. A SSL certificate must be setup for us by the z/OS HTTP Server
  - Instructions for the creation of the SSL certificate are beyond the scope of this document. We may be able to help with setting up self-signed certificates using RACF. Please reach out to us if you need help. For other ESM, please refer to their respective manuals on how to manage certificates.
  - A self-signed certificate may be used be but the signing CA must be made available and installed to the local machine running the extension.
  - The SSL certificate must reside in a keyring that is owned by z/OS HTTP Server e.g. owned by *WEBSRV*.
- The gui/XDC Server requires 2 TCPIP ports in z/OS.
  - The first port is used for communications between the z/OS HTTP Server and the VS Code Extension.
  - The second port is used for internal communications between the z/OS HTTP Server and the gui/XDC Server.
  - These 2 ports should be added to the TCPIP PROFILE so that they are reserved and will be not allocated by TCPIP for use by any job requesting a default port.
- The VS Code extension is built with VS Code version ***1.110.0***. You must be at this level or higher in order to install and use the extension. If you kept up to date with VS Code updates, then you are good to go. (At the time of writing, VS Code is at version ***1.130.0***)

With that out of the way, let's get started.

[34] Z/XDC itself supports zOS 2.5.

**Add the gui/XDC Server proc** A startup proc for the XDCGUI started task, such as is shown in Figure 22 below, needs to be added to one of your system's PROCLIBs. The proc's name typically is "XDCGUI". A model for the proc can be found in hlq.XDCV31.XDCSRVER(XGUIPROC).

```
//* caps off 1                                                          XDCLMGUI
//XDCGUI   EXEC PGM=XDCLMGUI,PARM='--cold --fcgi-port ####', [fcgi-port must match z/OS HTTP Server config]
//             REGION=0M
//*
//STEPLIB  DD  DISP=SHR,DSN=<hlq>.XDCV31.XDCGUI  [this library needs to be APF authorized]
//*
//CEEOPTS  DD  *
  POSIX(ON)
  TRAP(ON,NOSPIE)
//*
//SYSPRINT DD  SYSOUT=*
//STDOUT   DD  SYSOUT=*
//STDERR   DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//CEEMSG   DD  SYSOUT=*
//CEEDUMP  DD  SYSOUT=*
//*
```

Figure 22: Sample XGUIPROC proc

The gui/XDC Server can only be stopped with the the **STOP** (*P XDCGUI*) or **FORCE** (*FORCE XDCGUI*) operator commands. The **CANCEL** operator command will be rejected.

**NOTE: <hlq>.XDCV31.XDCGUI must be APF authorized.**

**Adding a VHOST to z/OS HTTP Server** Editing */web/httpd1/httpd.conf* requires a user with superuser access or with the user ID that is the owner of the file, as well as a text editor that supports EBCDIC. You can also download *httpd.conf* to your desktop via FTP in ASCII mode, editting it there and transfer it back up to your system. It is highly recommended to create a backup of *httpd.conf* before any edits.

Before editing *httpd.conf*, it is recommended to creating a directory under USS that is on one of your own ZFS volumes. Once created, use the *chown* USS command to change the group and user to the same group and user used by z/OS HTTP Server, eg. WEBGRP and WEBSRV.

```
# This is the port number for HTTPS communications with the VS Code extension.
# Add this line with the rest of the "Listen" statements.
Listen 8443

# In the "Dynamic Shared Object (DSO) Support" section, enable the
# following 2 modules by removing the leading hash mark (#) or adding
# the following to lines to the end of the list.
LoadModule rewrite_module modules/mod_rewrite.so
LoadModule proxy_fcgi_module modules/mod_proxy_fcgi.so


######################################################################
#                 Start of Virtual Host definition.                  #
######################################################################

# XDC Clone name. Will be used as part of the GUI Server URL in the VS Code extension.
Define xdcclone "xdc"

# TCPIP port assigned to the XDC gui/Server
# This port number must match what is specified in the XDCGUI proc.
Define fcgiport "15003"

# A USS directory for use by this VHOST.
# Needs to be accessible by the user and group associated with z/OS HTTP Server, eg. WEBSRV and WEBGRP.
Define documentroot "/u/csw/www/htdoc/"

# A log directory for this VHOST. Needs to be accessible by WEBSRV and WEBGRP
Define logdir "/u/csw/www/logs/"

# The VHOST will use the same HTTPS port used earlier in the the Listen statement.
<VirtualHost *:8443>

  # A URL matching your installations naming rules, prefixed by the XDC clone name.
  ServerName $xdcclone.some.url

  # Enable SSL
  SSLProxyEngine on
  SSLEnable
  SSLClientAuth none

  # Points to a key ring containing a certificate for https.
  # The keyring must be owned by the IBM HTTP Server, eg. WEBSRV.
  KeyFile /saf WEBSRV/WEBSRVring

  # The certifcate name in the keyring to use.  A self-signed certificate can be used.
  SSLServerCert xdccsmlocalcert1

  # Disable older and less secure SSL protocols.
  SSLProtocolDisable SSLv2 SSLv3
```

Figure 23: Sample httpd.conf configuration statements

```
  DirectoryIndex index.html index.html.var

  # This instructs the IBM HTTP Server to pass any incoming connection to the /api route to be passed
  # to gui/XDC Server.
  ProxyPass /api fcgi://127.0.0.1:$fcgiport/

  # This defines the api route that is used by the VS Code Extension
  # to communicate with Z/XDC.  The route must me /api, do not change it.
  <Location "/api">

    # This allows bearer tokens to be passed by the Fcgi Proxy.
    CGIPassAuth on

    # Enable mod_rewrite to capture the Origin header
    RewriteEngine On

    # Store the Origin header in an environment variable
    RewriteCond %HTTP:Origin ^(https?://.+)$ [NC]
    RewriteRule .* - [E=ORIGIN:%1]

    # Set CORS headers dynamically to ALLOW all incoming connection. You may want to tighten these
    # rules. Please refer to the Apache HTTPD documentation for details.
    Header always set Access-Control-Allow-Origin "%ORIGINe" env=ORIGIN
    Header always set Access-Control-Allow-Credentials "true"
    Header always set Access-Control-Allow-Methods "GET, POST, OPTIONS, PUT, PATCH, DELETE"
    Header always set Access-Control-Allow-Headers "Content-Type, Authorization"

    # Handle CORS preflight requests (OPTIONS) directly by mod_rewrite and z/OS HTTP Server.
    RewriteCond %REQUEST_METHOD OPTIONS
    RewriteRule .* - [R=204,END]

  </Location>

  # Sets up permissions and features for the document root. This folder is unused but is required.
  DocumentRoot $documentroot
  <Directory $documentroot>
    Options Indexes FollowSymLinks
    IndexOptions NameWidth=*
    AllowOverride All
    Require all granted

    CharsetSourceEnc  UTF-8
    CharsetDefault    UTF-8
    CharsetOptions    DGWCompat
    AddDefaultCharset UTF-8

  </Directory>

  #LogLevel debug
  LogLevel info rewrite:trace5
  ErrorLog  $logdir/xdc_error_log
  CustomLog $logdir/xdc_access_log common
</VirtualHost>
```

Figure 24: Sample httpd.conf configuration statements

When you feel comfortable with the edits to *httpd.conf*, restart the z/OS HTTP Server. Once z/OS HTTP Server comes back up, open a browser and visit ***https://{${xdcclone}.{some.url}:{port}/api/***. You should see a JSON string displaying the Z/XDC clone name and gui/XDC Server's version number.

**"name":"XDC","version":"0.1.0"**

If the z/OS HTTP Server does not come back up, look in **/web/httpd1/logs/error.log** and **${logdir}/xdc_error_log** for details.

## Step 15: Installing the VS Code Extension

To installl the XDC VS Code Extension, you will need access to the extension package named **code-xdc-0.x.y.vsix**.

The quickest and simplest way to install our extension is to search for **XDC** in the Microsoft Marketplace directly in VS Code. The extension package can also be downloaded directly from Microsoft Marketplace or from our website .

With the Z/XDZC VS Code Extension package saved on your desktop, open up VS Code and then...

- Step 1: Open the VS Code Extensions panel.
- Step 2: Click on the hamburger menu...
- Step 3: Select "*Install from VSIX...*"
- Step 4: Navigate to where **code-xdc-0.x.y.vsix** is saved on your desktop and select it.

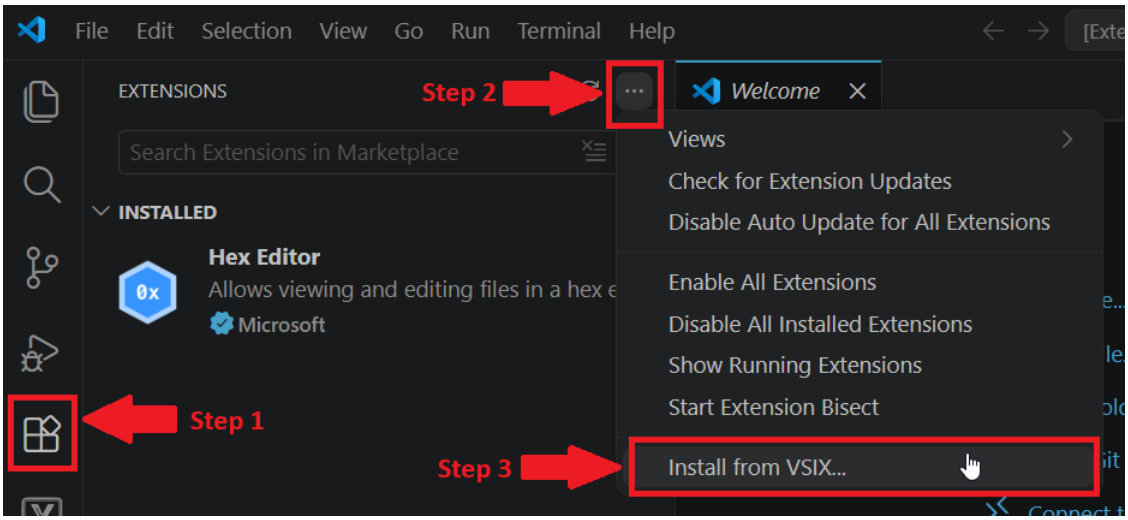


Figure 25: Steps to install VS Code extension

The XDC VS Code Extension can also be installed from the command line. Open a command prompt/shell/Powershell/etc and change to the directory containg the XDC VS Code Extension package and issue:

```
code --install-extension code-xdc-0.x.y.vsix
```

And, that's it. Once installed, the XDC Extension will appear in the side panel. When a new code drop occurs, just install the new package. VS Code will automatically replace the older version with the new one.

Refer to the Z/XDC VS Code Extension Quick Start on how to get started using the extension.